TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2013

32

Dorian Gálvez López

Place and Object Recognition for Real-time Visual Mapping

Departamento
Informática e Ingeniería de Sistemas

Director/es
Tardós Solano, Juan Domingo

ISSN 2254-7606

Prensas de la Universidad Universidad Zaragoza

Universidad Zaragoza
1542

Tesis Doctoral

PLACE AND OBJECT RECOGNITION FOR REAL-TIME VISUAL MAPPING

Autor

Dorian Gálvez López

Director/es

Tardós Solano, Juan Domingo

UNIVERSIDAD DE ZARAGOZA

Informática e Ingeniería de Sistemas

2013

Departamento de
Informática e
Ingeniería de Sistemas
Universidad Zaragoza

Ph.D Thesis

Place and Object Recognition for Real-time Visual Mapping

Dorian Gálvez-López

Supervised by Juan D. Tardós

Computer Science and Systems Engineering Department
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
January, 2013

Place and Object Recognition for Real-time Visual Mapping

Dorian Gálvez-López

Supervisor

Juan Domingo Tardós	Universidad de Zaragoza, Spain

Dissertation Committee

Patric Jensfelt	Kungliga Tekniska Högskolan, Sweden
José Neira Parra	Universidad de Zaragoza, Spain
Juan Andrade Cetto	Universitat Politècnica de Catalunya, Spain
José Luis Blanco Claraco	Universidad de Almería, Spain
Andreas Nüchter	Jacobs University, Germany
Pedro Piniés Rodríguez	Universidad de Zaragoza, Spain
Ricardo Vázquez Martín	Centro Andaluz de Innovación y Tecnologías de la Información y las Comunicaciones, Spain

International Reviewers

Margarita Chli	Eidgenössische Technische Hochschule Zürich, Switzerland
Giorgio Grisetti	Sapienza Università di Roma, Italy
Brian Williams	Google, USA

Submitted in fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science and Systems Engineering with "International Doctor" Mention in the Computer Science and Systems Engineering Department, Escuela de Ingeniería y Arquitectura, Universidad de Zaragoza. January, 2013.

Project framework

The work in this thesis has been funded by the following institutions:

- Pre-doctoral grants:
  - Ministerio de Educación, Cultura y Deporte of Spain: FPU-AP2008-02272;
  - Diputación General de Aragón, Spain: DGA-2008-B013/09.
- Other grants:
  - European Union,
  - Comisión Interministerial de Ciencia y Tecnología (Spain),
  - Diputación General de Aragón – Fondo Social Europeo, group T04 (Spain).

The work in this thesis has been developed with the Robotics, Perception and Real Time Group of the University of Zaragoza in the framework of the following projects:

- RoboEarth: EU Seventh Framework Programme, FP7-2009-248942;
- Rawseeds: EU Sixth Framework Programme, FP6-2005-IST-045144;
- nSPLAM: CICYT, DPI2009-13710;
- SLAM6DOF: CICYT, DPI2006-13578.

Resumen

Este trabajo aborda dos de las principales dificultades presentes en los sistemas actuales de localización y creación de mapas de forma simultánea (del inglés *Simultaneous Localization And Mapping*, SLAM): el reconocimiento de lugares ya visitados para cerrar bucles en la trajectoria y crear mapas precisos, y el reconocimiento de objetos para enriquecer los mapas con estructuras de alto nivel y mejorar la interación entre robots y personas.

En SLAM visual, las características que se extraen de las imágenes de una secuencia de vídeo se van acumulando con el tiempo, haciendo más laboriosos dos de los aspectos de la detección de bucles: la eliminación de los bucles incorrectos que se detectan entre lugares que tienen una apariencia muy similar, y conseguir un tiempo de ejecución bajo y factible en trayectorias largas. En este trabajo proponemos una técnica basada en vocabularios visuales y en bolsas de palabras para detectar bucles de manera robusta y eficiente, centrándonos en dos ideas principales: 1) aprovechar el origen secuencial de las imágenes de vídeo, y 2) hacer que todo el proceso pueda funcionar a frecuencia de vídeo.

Para obtener beneficio del origen secuencial de las imágenes, presentamos una métrica de similaridad normalizada para medir el parecido entre imágenes e incrementar la distintividad de las detecciones correctas. A su vez, agrupamos los emparejamientos de imágenes candidatas a ser bucle para evitar que éstas compitan cuando realmente fueron tomadas desde el mismo lugar. Finalmente, incorporamos una restricción temporal para comprobar la coherencia entre detecciones consecutivas.

La eficiencia se logra utilizando índices inversos y directos y características binarias. Un índice inverso acelera la comparación entre imágenes de lugares, y un índice directo, el cálculo de correspondencias de puntos entre éstas. Por primera vez, en este trabajo se han utilizado características binarias para detectar bucles, dando lugar a una solución viable incluso hasta para decenas de miles de imágenes.

Los bucles se verifican comprobando la coherencia de la geometría de las escenas emparejadas. Para ello utilizamos varios métodos robustos que funcionan tanto con una como con múltiples cámaras. Presentamos resultados competitivos y sin falsos positivos en distintas secuencias, con imágenes adquiridas tanto a alta como a baja frecuencia, con cámaras frontales y laterales, y utilizando el mismo vocabulario y la misma configuración. Con descriptores binarios, el sistema completo requiere 22 milisegundos por imagen en una secuencia de 26300 imágenes, resultando un orden de magnitud más rápido que otras técnicas actuales.

Se puede utilizar un algoritmo similar al de reconocimiento de lugares para resolver el reconocimiento de objetos en SLAM visual. Detectar objetos en este contexto es particularmente complicado debido a que las distintas ubicaciones, posiciones y tamaños en los que se puede ver un objeto en una imagen son potencialmente infinitos, por lo que suelen ser difíciles de distinguir. Además, esta complejidad se multiplica cuando la comparación ha de hacerse contra varios objetos 3D. Nuestro esfuerzo en este trabajo está orientado a: 1) construir el primer sistema de SLAM visual que puede colocar objectos 3D reales en el mapa, y 2) abordar los problemas de

escalabilidad resultantes al tratar con múltiples objetos y vistas de éstos.

En este trabajo, presentamos el primer sistema de SLAM monocular que reconoce objetos 3D, los inserta en el mapa y refina su posición en el espacio 3D a medida que el mapa se va construyendo, incluso cuando los objetos dejan de estar en el campo de visión de la cámara. Esto se logra en tiempo real con modelos de objetos compuestos por información tridimensional y múltiples imágenes representando varios puntos de vista del objeto.

Después nos centramos en la escalabilidad de la etapa del reconocimiento de los objetos 3D. Presentamos una técnica rápida para segmentar imágenes en regiones de interés para detectar objetos pequeños o lejanos. Tras ello, proponemos sustituir el modelo de objetos de vistas independientes por un modelado con una única bolsa de palabras de características binarias asociadas a puntos 3D. Creamos también una base de datos que incorpora índices inversos y directos para aprovechar sus ventajas a la hora de recuperar rápidamente tanto objetos candidatos a ser detectados como correspondencias de puntos, tal y como hacían en el caso de la detección de bucles.

Los resultados experimentales muestran que nuestro sistema funciona en tiempo real en un entorno de escritorio con cámara en mano y en una habitación con una cámara montada sobre un robot autónomo. Las mejoras en el proceso de reconocimiento obtienen resultados satisfactorios, sin detecciones erróneas y con un tiempo de ejecución medio de 28 milisegundos por imagen con una base de datos de 20 objetos 3D.

Abstract

This work addresses two of the main difficulties present in recent visual simultaneous localization and mapping (SLAM) systems: the recognition of revisited places to close loops and build accurate maps, and the recognition of objects to enrich maps with high-level entities to enhance human-robot interaction.

In visual SLAM, image features are accumulated over time, hindering two aspects of the loop closing detection labor: the rejection of loops wrongly detected among large sets of similar looking places, and feasibly low execution time in long-term trajectories. We propose a technique based on visual vocabularies and bags of words to detect loops in a reliable and efficient manner, focusing on two key ideas: 1) exploiting the sequentiality of video streams, and 2) making the process video-rate capable.

We benefit from sequentiality by computing a normalized similarity score to compare images and increase distinguishability of correct detections. We also group together candidate matches to prevent competition between images depicting the same place. Finally, we add a temporal constraint to require consecutive detections to be consistent.

Efficiency is addressed by using inverted and direct indexes and binary features. An inverted index speeds up the comparison between place images, and the direct one allows a fast computation of feature correspondences. Binary features are for the first time used here to detect loops, providing a scalable solution able to deal with thousands of images.

Loop verification is considered by checking the consistency of scene geometry. We show reliable methods to perform this step with one or multiple cameras. We present results with no false positives in very different datasets, with images collected at high and low frame-rates and with frontal and lateral cameras, using the same vocabulary and settings. When using binary features, the whole technique requires 22ms per frame in a sequence with 26300 images, being one order of magnitude faster than previous approaches.

A similar algorithm to that for place recognition can be used to address object recognition in visual SLAM. Detecting 3D objects in this context is particularly challenging due to the potentially infinite number of different sizes, locations and poses of objects in images, which make them less noticeable. In addition, this complexity multiplies when considering several 3D objects.Our effort in this work is aimed at: 1) building the first visual SLAM system that places real 3D objects in the map, and 2) addressing the scalability issues in which multiple objects and views result.

We present a monocular SLAM system that recognizes 3D objects, inserts them into the map, and refines their 3D pose as the map is built, even when objects get out of the field of view of the camera. This is done at real-time with object models composed of three-dimensional information and multiple images representing different view-points of the object.

We then focus on the scalability of the 3D object recognition stage. We present a fast technique to segment images into regions of interest to detect small or distant objects. We propose modeling objects as a single bag of words of binary features associated to 3D points, instead of

independent view-point images. We also benefit from inverted and direct indexes, which permit a very fast object retrieval and correspondence computation, as in the loop detection case.

Experimental results show real-time performance for a hand-held camera imaging a desktop environment and for a camera mounted in a robot moving in a room-sized scenario. Our enhanced approach shows no false positive detections and an average execution time of 28ms per image with a database of 20 three-dimensional objects.

Contents

Chapter 1

Introduction

1.1 Visual simultaneous localization and mapping

Science on robotics has remarkably evolved during the last few years, and its applications are little by little becoming part of our lives. From autonomous platforms that transport products in warehouses, passing through domestic robotic vacuum cleaners, to humanoids able to make pancakes (Beetz et al. 2011), these applications present substantial challenges that are being tackled through different lines of research. One of the most important ones is simultaneous localization and mapping (SLAM), which allows mobile robots to create a map of their area as they explore it. This provides self-localization and knowledge about the environment, which is basic for any robot that must interact with its surroundings.

The simultaneous localization and mapping problem consists in estimating the position of a robot and the landmarks that it observes concurrently, as the robot moves around an unexplored area. In other words, it provides the location of the robot in a map that is created at the same time, and that changes as the robot moves. This well-founded problem has been an active research topic for the last 25 years, and it has also been tackled in a lot of different ways during this time. Durrant-Whyte and Bailey offer a comprehensive survey of the problem and its solutions (Durrant-Whyte & Bailey 2006, Bailey & Durrant-Whyte 2006). Addressing this problem involved a breakthrough in mobile robotics, since it has made it possible to provide mobile platforms with real autonomy.

The spread of inexpensive cameras has permitted SLAM approaches to benefit from visual information. In these cases, local features such as points, segments or regions are extracted from video images and used as landmarks to build maps. Thus, maps comprise geometrical entities that are repeatable and recognizable among images, and hence, suitable for tracking. At every step, the observed features must be matched in a jointly consistent manner to estimate the motion of the robot; this process is commonly known as *data association*. Building long-term maps from imagery entails an increasing complexity problem due to the large amount of data to process. This is especially evident when a robot moves along a trajectory and revisits a place, occurring a loop closure.

A moving robot must be able to detect loops by recognizing the current place as one of all the previously visited ones. Then, the estimated positions of the robot and the landmarks located at that place can be merged to close the loop, reducing their uncertainty, obtaining the correct map topology and improving the accuracy of the global map. Since the robot is locating itself along the whole trajectory, it does not seem difficult to tell whether two positions are close enough to fire a loop detection. Nevertheless, this approach is hardly feasible in practice due to the

measurement error that accumulates over time, rising the uncertainty of the estimated location. Actually, the process has to go the other way around: the detection of loop closures is necessary to locate the robot and the landmarks better. Although conceptually similar, this is a special case of the data association stage; instead of associating observations obtained in consecutive steps, former measurements must be considered. This requires to inspect all the information acquired from the map so far, and visual imagery have proved to be a reliable source of data to perform independent loop closure detection for large-scale maps (Williams et al. 2009).

Geometrical elements are very convenient for robot localization because they are easy to obtain accurately. However, these produce maps that provide little information about the real environment, from a human point of view. The lack of conceptual meaning of geometrical entities makes them unsuitable when human-robot interaction is required in complex tasks. So there is a need to move from geometrical maps to meaningful maps with *semantic* content. Semantics in mapping has been considered in terms of places (Pronobis et al. 2010), objects (Rusu et al. 2009) and relationships between them defined by ontologies (Tenorth et al. 2012). These approaches allow to answer questions such as where the kitchen is, where the dishwasher is, or where mugs are usually found. In this work, we limit the broad meaning of *semantic maps* to those that include, in addition to geometrical elements, real objects as high-level entities, since they have semantic information that is easily understood by a person in a natural way (we usually know what an object is and for what it is used). Under this point of view, visual 3D object detection and recognition gains importance in SLAM systems.

Object detection and object (or instance) recognition attain different meanings in the object retrieval literature. Usually, *object detection* refers to the task of noticing the presence of a known object in an image and of locating it, so that it can be outlined. In addition, *instance recognition* verifies the detection and yields additional information about each instance of the detected object, e.g. a rigid-body transformation locating the object in the 3D space, if possible. Detection and recognition are very related because sometimes the detection does not occur until the object is recognized and, hence, verified. For this reason, these terms are used interchangeably in this work, with the meaning of locating and recognizing the object at the same time. Therefore, object recognition answers the question of what objects are present in an image, if any, and where they are. Locating the objects in the 3D space and not just in the image is beneficial for tasks like grasping, and especially in visual SLAM, because this allows to place them in the map for informative purposes, or to be used as landmarks. On the other hand, although people are able to recognize probably hundreds of objects at a glance, this is still an open problem in computer vision. The distance to the object, its orientation, the occlusion with other scene elements, etc. are some of the factors that do not entail any deal to us, but which hinder the labor of 3D object recognition algorithms. Simply considering the kind of surface of the objects completely changes the way of approaching the problem, since the distinctive properties of patterned, plain or transparent objects differ. Rigid textured objects, on which this work focuses, are a branch of this field of study.

Although latest research has provided solutions along that line (Grundmann et al. 2011, Hsiao et al. 2010), it has usually been aside from visual SLAM, whose context imposes certain conditions to which no attention is paid, such as the low quality of video images and the short time available between acquired frames. Classical algorithms focus mainly on obtaining high recall and accurate object location in a single image basis, at the expense of increasing the execution time. However visual SLAM provides plenty of images to be exploited in order to achieve high recall and accurate object location if the sequence is exploited as a whole. In real sequences, there are frames where objects cannot be detected since they are hardly distinguishable (because of their pose, blur, distance, etc.), so spending time trying recognition is not worth it because other camera frames, where the detection may be possible, are missed. This suggests the need of a different approach

to perform fast object recognition for visual SLAM.

Nowadays, the SLAM problem as originally formulated is theoretically solved, but it still presents practical challenges that stop us from achieving robotic applications able to work in large-scale or long-term scenarios, and to share a minimum language with people. Looking toward the future of robotic mapping for human interaction, we can say that the way passes through counting on reliable and efficient methods to recognize places and objects, able to detect correct loop closures to refine maps, and to place real objects in them.

1.2 Two problems, a general framework

Both visual place and object recognition problems have a common basis: a large amount of real images requires a compact representation that allows a fast processing, in order to meet visual SLAM conditions.

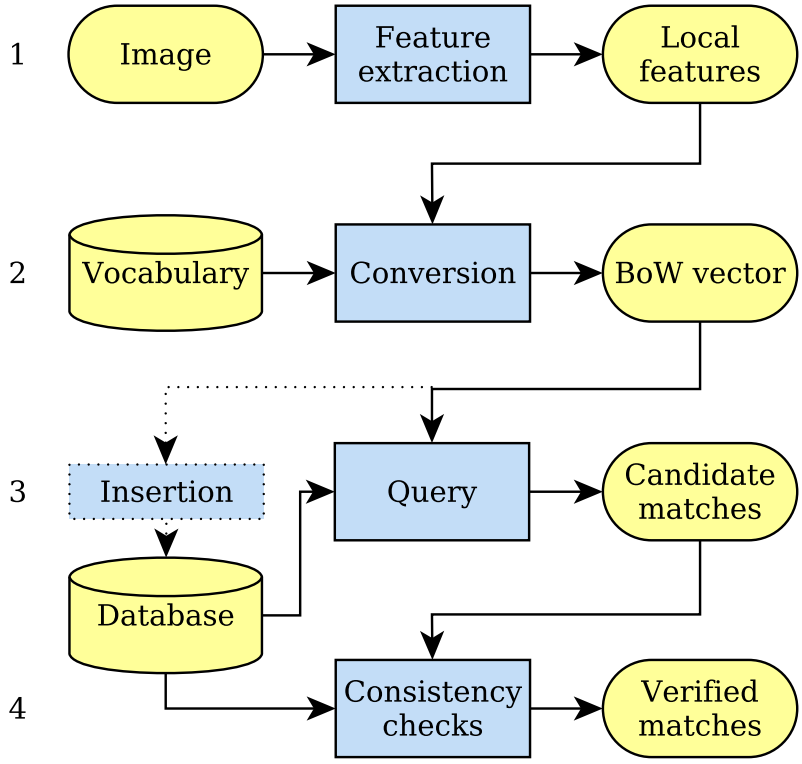


Figure 1.1: Framework to address place and object recognition. Our proposed framework works in four steps: 1) local features are extracted from an input image; 2) with a visual vocabulary, these are converted into a bag-of-words vector, 3) which is used to query a database to select as candidate match the stored bag of words with the highest similarity. If necessary by the application, the acquired bag-of-word vector can be inserted into the database, so it can be retrieved later. Finally, 4) the candidate is checked for consistency to be accepted.

Visual vocabularies (Sivic & Zisserman 2003) are able to play this role. These are mechanisms to represent images as bags of words, a data type that builds on local features and that summarizes the appearance of an image as a sparse numerical vector, or equivalently, as a histogram of frequent features, named *words*. This translates the task of comparing images into the bag-of-words space, reducing its complexity.

The compact representation permitted by bags of words is usually complemented with the efficiency of inverted indexes. In the information retrieval domain, where text documents are searched for query words, an inverted index (or inverted file) is a structure that maps words with the documents where they are present. Analogously, an inverted index can also map visual words with the images that contain them. These enable a quick content-based retrieval of images, basic in order to manage a large amount of images and achieve scalability.

Direct indexes are also useful for information retrieval, but are not widely spread in the vision area. Opposing to inverted indexes, direct ones map images to words, which can be arranged in different useful manners, depending on the task to solve.

By using together visual vocabularies and inverted and direct indexes, we build a general mechanism that offers a solution to the image comparison problem in large image databases. We propose a framework that makes use of this mechanism to address both the place and object recognition problems, as depicted by Figure 1.1. Its pipeline works in four steps:

1. Local features are extracted from an input image.

2. These are converted into a bag-of-words vector (BoW vector) by means of the visual vocabulary.

3. A database containing bags of words (from places or objects) is queried to obtain the candidate matches that present the highest appearance similarity with the input bag of words. Internally, the database is constituted by inverted and direct indexes, ensuring a quick access.

4. The candidate matches are checked for geometrical or temporal consistency with their corresponding models in the database. If they are successful, the matches are verified as a recognized place or object.

This general framework is not subjected to any specific implementation of visual vocabularies or image databases, and it can be easily adapted to the specific place and object recognition problems.

In the case of place recognition, the input is every image acquired as the robot moves, and the database is composed of the bag-of-words vectors of all the acquired images along the trajectory. Therefore, it is necessary to extend the third step by inserting the current bag-of-words vector into the database. This way, it can be matched in a future detection of a loop closure. Loop detections must pass two consistency checks to be accepted. Firstly, temporal consistency has to be satisfied. For that, it is required that the candidate match is consistent with loops detected in some of the last video frames. The process of querying the database to obtain a candidate that satisfies temporal consistency is named *loop candidate retrieval*. Secondly, the geometry of the involved scenes must match; this is the *loop verification*. We check this condition by using epipolar geometry with one and two cameras, and with Cadena et al.'s (2012) probabilistic algorithm that makes use of conditional random fields.

For object recognition, the input is the current frame acquired by a camera in which object detection is carried out. In this case, all the features extracted from the entire image in step one can be converted into a bag of words; otherwise, they can be separated into subsets and converted into several BoW vectors that result in individual queries. This second method is

useful to divide the image into smaller regions to avoid clutter while looking for small objects. The database contains models of objects that represent their appearance and geometry. Thus, when querying, the *object candidate retrieval* returns as candidate match the model with the most similar appearance to the input BoW vector. In the fourth and final step, the *object verification* checks the geometry between the candidate model and the input features for consistency.

As we present along this thesis, our proposed framework is a capable system that yields successful results in the place and object recognition problems under several challenging circumstances.

1.3 Goals and contributions of this work

Place and object recognition are topics widely studied by the robotics community, but although a great attention is paid to low-latency systems, video-rate performance, very beneficial for visual SLAM, had not been achieved. The main goal of this work is to propose novel solutions that are able to achieve reliable detections of loops and objects in heterogeneous and challenging cases, without false positives, and, at the same time, to result in video-rate performance suitable for real SLAM algorithms in large-scale scenarios.

Aiming at this goal, the contributions of our work have been:

- For the first time, we use binary features in the bags-of-words approach to solve place and object recognition, where floating-point features were usually utilized before. Binary features remove the bottleneck in these recognition problems, decreasing execution time from hundreds to tens of milliseconds.
- We enhance the bag-of-words approach by incorporating novel direct indexes for visual words to speed up the computation of corresponding points in the verification stage. These allow a reduction of the execution time of this step from one hundred to a couple of milliseconds.
- We present the benefits of exploiting in several ways the sequentiality of images gathered by a video camera to improve the reliability of detections in the loop detection problem. These novelties overcome the results of the state-of-the-art techniques.
- Departing from most previous works, to avoid over-tuning, we present extensive results of the loop detection algorithm using a single vocabulary, obtained from independent data, and the same parameter configuration, obtained from a set of training datasets, without peeking on the evaluation datasets. Our loop detection system yields no false positives in a sequence with 26300 images, and requires 22ms per frame, being one order of magnitude faster than previous approaches.
- We present the first real-time monocular SLAM system that performs recognition of 3D objects as creates the map. Objects are located in the map and their pose is refined over time. We then enhance the detection of 3D objects by using a comprehensive representation of their appearance in a single bag-of-words vector, allowing recognition to run in 28ms per image with medium-sized databases.

1.4 Publications, Videos and Software

1.4.1 Publications

This work has resulted in the following publications:

- Loop detection:

  1. Piniés P., L. M. Paz, D. Gálvez-López & J.D. Tardós (2010), 'CI-Graph SLAM for 3D Reconstruction of Large and Complex Environments using a Multicamera System', *International Journal of Field Robotics* **27**(5), 561–586.
  2. Cadena, C., D. Gálvez-López, F. Ramos, J.D. Tardós & J. Neira (2010), Robust place recognition with stereo cameras, in 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 5182–5189.
  3. Cadena, C., D. Gálvez-López, J.D. Tardós & J. Neira (2012), 'Robust place recognition with stereo sequences', *IEEE Transactions on Robotics* **28**(4), 871–885.

- Loop detection with binary features:

  1. Gálvez-López, D. & J.D. Tardós (2011), Real-time loop detection with bags of binary words, in 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 51–58.
  2. Gálvez-López, D. & J.D. Tardós (2012), 'Bags of binary words for fast place recognition in image sequences', *IEEE Transactions on Robotics* **28**(5), 1188–1197.

- 3D Object recognition:

  1. Civera, J., D. Gálvez-López, L. Riazuelo, J.D. Tardós & J.M.M. Montiel (2011), Towards semantic slam using a monocular camera, in 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1277–1284.
  2. Waibel, M., M. Beetz, R. D'Andrea, R. Janssen, M. Tenorth, J. Civera, J. Elfring, D. Gálvez-López, K. Haussermann, J.M.M. Montiel, A. Perzylo, B. Schiessle, O. Zweigle & R. van de Molengraft (2011), 'Roboearth', *IEEE Robotics Automation Magazine* **18**(2), 69–82.
  3. Gálvez-López, D. & J.D. Tardós (2013), Fast and scalable 3D object recognition with vocabularies of binary words for semantic SLAM, To be submitted to 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)'.

Although they are not tackled by this thesis, our work resulted in other research collaborations, and in turn, it was partly originated from the inspiration received from them:

1. Gálvez-López, D., K. Sjöö, C. Paul & P Jensfelt (2008), Hybrid laser and vision based object search and localization, in 'IEEE International Conference on Robotics and Automation', pp. 2636–2643.

2. Sjöö, K., D. Gálvez-López, C. Paul, P. Jensfelt & D. Kragic (2009), 'Object search and localization for an indoor mobile robot', *Journal of Computing and Information Technology* **17**(1), 67–80.

3. Majdik, A., D. Gálvez-López, G. Lazea & J.A. Castellanos (2011), Adaptive appearance based loop-closing in heterogeneous environments, in 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1256–1263.

1.4.2 Videos

The results presented in this work are complemented by the videos detailed next. These can be found in `http://webdiis.unizar.es/~dorian/videos`:

1. Real-time loop detection:
   1.a. NewCollege dataset: `GalvezTRO12_NewCollege.avi`
   1.b. Bicocca25b dataset: `GalvezTRO12_Bicocca25b.avi`
2. 3D Object recognition:
   2.a. Visual SLAM, hospital room: `CiveraIROS11_Hospital.avi`
   2.b. Visual SLAM, desktop: `CiveraIROS11_Desktop.avi`
   2.c. Bag-of-words object recognition: `Galvez13_ObjRec.avi`

1.4.3 Software

As a final contribution, all the implementation of our work is publicly available as open source software. It can be found in:

- `http://webdiis.unizar.es/~dorian`
- `http://www.ros.org/wiki/roboearth`

1.5 Structure

The layout of this work is divided into 6 chapters. We start by introducing the problem and putting our work in context with respect to the state of the art in the current Chapter 1 and in Chapter 2.

In Chapter 3 we describe the usage of visual vocabularies, bags of words and inverted indexes to present our image comparison framework used with generic image features.

In Chapter 4 we look into the computational requirements of our loop detection approach, and propose a new configuration to work at video-rate with binary features and a direct index.

Chapter 5 is devoted to the 3D object recognition problem. Here, we first present a monocular SLAM system that recognizes real 3D objects and places them in the map, by using a multi-view object modeling approach. Later, we make use of our framework, introduced in previous chapters, to enhance the object recognition algorithm.

Finally, in Chapter 6 we recapitulate the achievements made by our work, and discuss how it can be improved in a future research.

Chapter 2

Related work

2.1 Image features and bags of words

Images can be represented by very different structures. Global descriptors such as GIST (Oliva & Torralba 2001), scene texture (Torralba et al. 2003) or color moments in CIELUV space (Boutell et al. 2004) provide a single joint description of the whole image. Oppositely, there are other elements that can be used to acquire local information of the image in an individual manner. For example, corner points (Harris & Stephens 1988, Rosten & Drummond 2006), salient interest points (Lowe 2004, Bay et al. 2008), line segments (Canny 1986), invariant regions (Tuytelaars & Van Gool 2004), color regions (Ekvall & Kragic 2005), etc. Although global descriptors have shown their suitability for image recognition (Douze et al. 2009), local features present advantages in camera localization (Schaffalitzky & Zisserman 2002, Bay et al. 2005), fitting well in the visual SLAM context.

SIFT and SURF key point features. In this work, we focus on local features based on image key points. Key points are salient points that have a well-defined position in the image and that present certain stability before some image transforms. From the large amount of point features proposed by the community (Tuytelaars & Mikolajczyk 2008, Mikolajczyk & Schmid 2005), the interest points based on a scale-invariant feature transform (SIFT) (Lowe 2004) supposed a cornerstone for plenty of computer vision and robotics applications, such as scene reconstruction (Snavely et al. 2006), visual servoing (López-Nicolás et al. 2010), loop detection (Angeli et al. 2008) or object recognition (Lowe 1999). Their great popularity is due to their stability before scale and rotation changes, and reasonable affine and perspective changes. To achieve invariance, SIFT points are selected among the scale space by computing differences after applying Gaussian convolution masks to the image. The scale at which the key point is selected determines the size of the patch where the descriptor is computed. To ensure rotation invariance, the patch is rotated according to the dominant orientation of its intensity gradients. The SIFT descriptor comprises orientation histograms of several regions around the key point, yielding a 128-dimensional floating-point vector per feature. Following the line of SIFT, Bay et al. (2008) proposed to use integral images (Viola & Jones 2001) to make the blob detection and descriptor computation faster, obtaining the speeded-up robust features (SURF). SURF features provide a similar descriptor as SIFT, of 64 or 128 components, with a comparable distinctiveness and at a fraction of the computational cost of SIFT. For this reason they experienced a quick spread in robotics applications (An et al. 2009, Cummins & Newman 2008, Murillo et al. 2007, Pretto et al. 2007) and we make use of them in this work, in our initial approaches to recognize both

place and objects.

Binary features. One of the drawbacks of SIFT and SURF for some applications is the execution time required for their extraction, usually between 100 and 700ms per image. Apart from GPU implementations (Heymann et al. 2007), there are other similar features that try to reduce this computation time by, for example, approximating the SIFT descriptor (Grabner et al. 2006) or reducing the dimensionality with principal component analysis, as PCA-SIFT (Ke & Sukthankar 2004) or GLOH (Mikolajczyk & Schmid 2005) do. Another line of study goes toward more reduced ways to represent the feature information, as done for instance, by compact randomized tree signatures (Calonder, Lepetit, Fua, Konolige, Bowman & Mihelich 2010, Lepetit & Fua 2006). This approach calculates the similarity between an image patch and other patches previously trained in an offline stage. The descriptor vector of the patch is computed by concatenating these similarity values, and its dimensionality is finally reduced with random ortho-projections. This yields a descriptor that is very fast to compute and suitable for real-time applications. Binary features go a step farther by utilizing bits instead of floating-point values to summarize some kind of gradient information. Binary robust independent elementary features (BRIEF) (Calonder, Lepetit, Strecha & Fua 2010), for example, create a bit string by comparing the intensity values of some predefined pairs of pixels within a patch. These are usually computed around corner-like points obtained from Rosten & Drummond's (2006) accelerated segment tests (FAST). FAST detects as corners the points surrounded by a Bresenham circle that presents a certain pattern of dark and light pixels. FAST points together with BRIEF descriptors yield a feature that requires very little time to be computed and matched. We make use of them to solve the place recognition problem for mobile robots. On the other hand, they do not calculate any rotation or scale information, so they hardly provide invariance to them. This is not desirable for object recognition due to the multiple points of view from which an object can be seen. We address this lack by using the rotation-aware BRIEF features (ORB) (Rublee et al. 2011), which compute a dominant orientation with a simple centroid technique to rotate the pattern of pairs used by BRIEF to compute its descriptor. Others recent features as BRISK (Leutenegger et al. 2011) or FREAK (Alahi et al. 2012) resolve these issues as well, providing lightweight features with a similar distinctiveness and invariance to SIFT or SURF.

Bags of words. Feature descriptors can be further compacted by using bags-of-words representations, first proposed for images by Sivic & Zisserman (2003). This mechanism creates a dictionary of visual words, i.e. a visual vocabulary, and converts image features into vectors of words. Vocabularies can be constructed in an offline stage from training data, or in an incremental manner with the data on which they are applied (Angeli et al. 2008). The visual vocabulary of Sivic & Zisserman (2003) resulted from clustering the descriptor space. Hierarchical clustering was later proposed by Nister & Stewenius (2006) to enhance the conversion performance by linking clusters in a tree fashion, showing its reliability to perform image matching. In these trees, a cluster is represented as a point in the descriptor space, and depicted as a node in the tree. The training descriptors are clustered to create the nodes at the first level of the tree. Then, the training descriptors associated to each node are clustered again, creating child nodes. This is done recursively until obtaining the desired number of nodes. In this work, we use hierarchical clustering when working with SURF features because it has shown its effectiveness in real number spaces (Nister & Stewenius 2006). However, Trzcinski et al. (2012) noticed their loss of performance when applied on binary spaces owing to the thick boundary problem: given two clusters in the binary space, there is always a large number of points that are equidistant to them, so that they cannot be uniquely associated to one of these clusters. To mitigate this problem, Trzcinski et al. (2012) propose creating a set of randomized trees, coined *parc-trees*, to

select any of theirs nodes when an image descriptor is converted into a word. Following this idea, we also create a collection of trees when dealing with binary features for object recognition, but unlike parc-trees, we create our trees with hierarchical k-medoids, and consider leaves as words only. Lately, the use of this kind of randomized trees is becoming popular to match binary features (Muja & Lowe 2012). Aiming at binary descriptors, other structures can be used to create visual vocabularies. For example, local sensitive hashing (LSH) (Gionis et al. 1999) indexes bit strings in a collection of hash tables. A detailed comparison of some of these methods is given by Trzcinski et al. (2012).

2.2 Loop detection in visual SLAM

In the visual SLAM context, relocation, the *kidnapped robot problem* and the loop closure detection are three very related topics that have gained importance in recent years. Relocation allows the system to locate the camera in the map after tracking failure (e.g. due to rough motions and blurring defects) by inspecting recent frames. On the other hand, the kidnapped robot problem takes place when the robot must locate itself at an arbitrary position in the map, because its uncertainty is too large, or because there is no previous position at all (*wake-up* problem). Thus, all the position history is usually checked. The loop closure detection deals with the capability of a visual SLAM system to recognize at every frame a visited place, previously seen at any time. After an exploratory period, when areas non-observed for long are re-observed, standard matching algorithms fail. When they are robustly detected, loop closures provide correct map topology and data association to obtain consistent maps. Recently, a comparison among the most relevant loop closure methods for visual SLAM was presented by Williams et al. (2009). Three paradigms were compared: image-to-image, which works only in the image space, map-to-map, which uses metric map information, and image-to-map, which uses visual and metric information to perform relocation. The main conclusion of the work suggests a fusion of image-to-image and camera relocation to deal with the drawbacks of the individual methods. We focus on the image-to-map and image-to-image approaches for visual SLAM.

Image-to-map approaches. The work by Williams et al. (2007) is one of the first that added a robust module to provide a real-time monocular SLAM system with a relocation tool based on fast key point learning (Lepetit & Fua 2006). Loop detection is performed in a image-to-map framework where the camera pose is relocated using three correspondences between features and 3D points. Olson (2009) proposes selecting first a subset of map positions to perform later visual loop detection. For that, a topological graph of robot poses is maintained, so that when there is evidence of a unambiguous topological loop, visual matches between SIFT features are computed between their associated images to find a rigid-body transformation. Eade & Drummond (2008) also join visual loop detection and relocation by using a visual vocabulary to select matching candidates. These are then confirmed by using epipolar geometry between monocular images. This yields the relative pose of the camera with respect to the map, that enables relocation. Nevertheless, we combine visual vocabularies with additional techniques that generalizes on the number of cameras. As Williams et al. (2009) conclude, for small environments, map-to-image methods achieve nice performance, but despite their robustness, they do not scale well for large environments, where image-to-image methods behave better.

Image-to-image approaches. If we focus on image-to-image techniques, we find that appearance-based methods are particularly suitable for online place recognition due to the richness in information that cameras provide. The basic image-to-image technique consists in build-

ing a database from the images collected online by the robot, so that the most similar one can be retrieved when a new image is acquired. If they are similar enough and they present consistency (usually geometrical consistency), a loop closure is detected. In recent years, many algorithms that exploit this idea have appeared (Paul & Newman 2010, Konolige et al. 2010, Cummins & Newman 2008, Angeli et al. 2008, Callmer et al. 2008), basing the image matching on comparing them as numerical vectors in the bag-of-words space (Sivic & Zisserman 2003).

FAB-MAP, the image-to-image method considered in the work by Williams et al. (2009), was the first successful appearance-only method to perform loop detection. For that, their authors proposed a probabilistic framework in which their system learned a generative appearance model based on a Chow-Liu tree (Chow & Liu 1968) encoding the words' co-visibility probability. With this, they could compute the probability of any two sets of observations being originated from the same location. Hence, given a vocabulary and an approximate probabilistic model of observations, it is possible to compute from a set of image features a probability distribution function over the places already visited, where loop closures appear as peaks. Although the algorithm complexity is linear in the number of places, learning a generative model is an offline process. Their authors improved their system in FAB-MAP 2.0 (Cummins & Newman 2011) by using an inverted index along with the vocabulary, as we do in this work. This system has proved to be very successful in large scale environments. With omnidirectional vision, it can run with full precision (no false positives), obtaining a recall of 48.4% and 3.1%, in trajectories 70Km and 1000Km in length (with a geometrical verification). FAB-MAP has become the gold standard regarding loop detection, but its robustness decreases when the images depict very similar structures for a long time, which can be the case when using frontal cameras, as we show in this work with results that overcome those by FAB-MAP.

Visual vocabularies are a powerful tool to detect loops, as well as versatile, as Angeli et al. (2008) shows. In their work two visual vocabularies (for appearance and color) are created online in an incremental fashion. The two bag-of-words representations are used together as input of a Bayesian filter that estimates the matching probability between two images, taking into account the matching probability of previous cases, very similarly to FAB-MAP. In contrast to these probabilistic approaches, we exploit the sequentiality of the acquired images and rely on a temporal consistency check to consider previous matches and enhance the reliability of the detections.

Features for loop closing. In most loop closing works (Paul & Newman 2010, Angeli et al. 2008, Callmer et al. 2008) the features used are SIFT or SURF. Here, we start by using SURF features in our framework as well. The work by Konolige et al. (2010) offers a qualitative change in this aspect, since it uses compact randomized tree signatures (Calonder, Lepetit, Fua, Konolige, Bowman & Mihelich 2010) instead. Our work bears a resemblance with the one by Konolige et al. (2010) in that we also propose enhancing the loop detection system by reducing the execution with efficient binary features. This leads us to use, on a second stage, a bag of binary words for the first time on this problem. Reducing the execution time allows to increase the working frequency of the loop detector. Unlike other approaches, we also propose a technique to prevent images collected in short time intervals and depicting the same place from competing among them during the place matching, avoiding difficulties arisen when running at high frequency.

Use of geometrical information. Image-to-image techniques are highly scalable for thousands of images, but they are not exonerated from failures. Features can be mismatched between images due to perceptual aliasing; an effect that is augmented by the discretization of visual vocabularies. Although addressing perceptual aliasing may require to fully merge the detection of

loops with topological maps (Werner et al. 2012), its effects in the appearance-based loop detection can be mitigated just by incorporating geometrical information to avoid false positives. For example, FAB-MAP 3D, recently proposed by Paul & Newman (2010), additionally includes in the feature descriptors the 3D distances provided by a laser scanner. This results in higher recall for the same precision in the experiments of FAB-MAP 2.0 (Cummins & Newman 2011), at the expense of requiring a more sophisticated hardware configuration. The complete place recognition problem can also be addressed using only 3D range data. For instance, Steder et al. (2010) extracts feature points from range images obtained by a 3D laser scanner in areas where the gradient changes significantly. They maintain a database where range images, described by these features, are stored, which is queried to obtain matching range images and detect loop closures. This system has high computational requirements compared with systems based on bags of words, but higher recall is attained. An important limitation is that this system cannot distinguish between locations with similar shape but different appearance, such as corridors, or with a different background beyond the sensor's range.

Nevertheless, instead of incorporating 3D data to the detection step, the road taken most often to consider geometry has been to add a geometrical check to verify the spatial consistency between the features of matching place images. Cummins & Newman (2011) incorporated a simplified constraint check consisting in finding a single rotation around the vertical axis with an omnidirectional camera installed on a car in FAB-MAP 2.0. With a normal camera, the distance between image features can be checked, as done by Callmer et al. (2008), who require some pairs of features in two matched images to minimize their spatial distance to ensure at least a weak geometrical consistency. However, it is more common to turn to epipolar geometry. The work by Valgren & Lilienthal (2010) checks the geometrical configuration of the matching scenes by means of an epipolar constraint. However, this is carried out against the complete image database, which can become inefficient for large environments. In this work, similarly to Angeli et al. (2008), we start by using an epipolar condition as well, but we apply it only to the most promising loop candidate, obtaining a better suited system for large-scale maps. Any of the cited verification methods require to compute feature correspondences. Unlike all the previous works, we present how a direct index can be introduced in the bag-of-words approach to speed-up this computation.

Other works make use of the geometrical information provided by stereo cameras. Majdik et al. (2011) use a stereo camera to measure the distance between reconstructed 3D points, and verify a loop if the sets of distances are consistent between two stereo pairs. Konolige et al. (2010) use a stereo camera as well but impose a stricter condition by computing the spatial 3D transformation between the matching images. However, they do not use any filter to consider consistency with previous matches, and this leads them to apply the geometrical check to several loop candidates. In this work we make use of two different methods when dealing with two cameras. We use epipolar geometry to require geometrical consistency between the matching images by each camera. This is enhanced by requiring image features to be already included into the 3D map created by the CI-Graph SLAM algorithm presented by Piniés & Tardós (2008). In the specific case of stereo cameras, we utilize Cadena's (2011) CRF-Matching algorithm, which models appearance and geometrical information in a probabilistic manner with conditional random fields (CRF). Cadena's (2011) CRF-Matching is an extension of the algorithm by Ramos et al. (2007) to match 2D laser scans associated to visual texture. Although Ramos et al. (2007) proposed the possibility of detecting loop closures with CRFs by taking the maximum log-likelihood among the matches between the current and all previous scans, this required a comprehensive comparison with all the previous scans, which is impractical in real applications. Furthermore, their metric did not provide a way to distinguish between true and false loop closures. We overcome both problems in this work by using CRF-Matching together with our

bag-of-words image matching framework and exploiting image sequentiality.

2.3 Object recognition for SLAM

As we introduced in Chapter 1, there are several ways to augment geometrical maps with semantic content. Here, we consider including objects into SLAM maps as high-level elements as a basic form of semantic mapping.

Objects as high-level map elements. Currently, the map produced by most SLAM algorithms is a joint estimation of geometric entities, as points and lines, without any semantic meaning or annotations attached to it. Although object recognition can enrich these maps, it has been scarcely combined with SLAM. The work by Galindo et al. (2005) is one of the few examples, using a map hierarchy together with object recognition in order to obtain additional information to classify the places where the objects are found. A similar idea is presented by Vasudevan et al. (2007), where the objects are recognized and located in the 3D space by means of a stereo camera. Vasudevan et al. (2007) carry out recognition in an single view basis, where SIFT features from an input image are matched with those from a model image. This recognition approach has been widely used due to its simplicity and good results (Sjöö et al. 2009, Zender et al. 2008, Gálvez-López et al. 2008, Meger et al. 2008, Ekvall et al. 2006). After recognition, the objects are inserted in the map. For example, in the works by Gálvez-López et al. (2008) and Sjöö et al. (2009) the objects are included into a 2D map laser map by estimating their positions, given the real size of their planar bounding boxes.

Object recognition can also be used to enhance the SLAM algorithm, as done by Bao et al. (2012), who detects objects, regions and points to reconstruct the 3D map. They impose constraints to the expected pose of the objects in the scene to improve the reconstruction. However, it makes it more difficult to deal with objects found at unexpected positions or without a clear scene context. In this work, we present a monocular SLAM system that builds 3D maps with 3D general objects, making no assumptions about the point of view they can be recognized. Differently from the works above, we do not only use the object appearance but also the estimated geometry in the object model. This information considers the whole surface of the 3D object and not a single view, so it is not constrained to bounding boxes. This allows objects to be tracked by the monocular SLAM after the insertion; hence making it possible to refine their 3D pose as the camera moves around the map. Our work bears a resemblance with that by Castle & Murray (2011), which registers planar objects into an EKF-based monocular SLAM. Their research makes use of SIFT features to construct the appearance model of the objects and to insert in the SLAM map three of the boundary corners of the plane. Compared to this approach, we are able to overcome the planar restriction and to deal with any object geometry.

Single-view object models. As introduced above, object recognition based on a single view has attained successful results in the semantic mapping field. The basic idea (Lowe 1999) consists in creating beforehand a collection of images depicting objects and extract their local features, so that when an input image is given, matches can be computed between their features and those from the model images. The detection is successful if some model obtain enough consistent matches. Since feature matching is usually very time-consuming, Lowe (1999) proposed using a k-d tree to speed up this step when using SIFT features.

This appearance-based recognition approach can be enhanced by representing images with visual vocabularies, as Sivic & Zisserman (2003) showed. The work that they presented describes an approach for object retrieval which searches for and localizes all the occurrences of

a user-outlined object in a video. The object is represented by a bag of words built on a set of viewpoint invariant region descriptors, so that recognition can proceed successfully despite changes in viewpoint, illumination and partial occlusion. The temporal continuity of the video within a shot is used to track the clusters and reject unstable regions reducing the effects of noise in the descriptors.

The work by Sivic & Zisserman (2003) became a *de facto* standard to address single-view object recognition in large sets of images. As Szeliski (2010, Chapter 14) describes, it can be summarized in the following steps. First, a collection of features is acquired from a set of images; these are clustered in the descriptor space and the visual vocabulary is obtained. The image database is then created by computing the *term frequency – inverse document frequency* (*tf-idf*) vector of each image. These are the bag-of-words vectors of the image stored in the database, which are indexed by an inverted index. The image retrieval is performed by extracting features from and input image and computing its bag-of-words vector with tf-idf values. It is compared with the other vectors in the database, accessed with the inverted index, by computing a similarity score. The top-ranked database images are selected as matching candidates and their spatial consistency is checked to select the final matches.

Along this line, Nister & Stewenius (2006) used a hierarchical vocabulary to achieve successful detections in large object databases of three different applications. They retrieved objects by matching images depicting them under different points of view, that were stored in a database with 6376 images in total. They also performed recognition with 40000 planar CD covers and arbitrary objects in a one million movie shot database, showing that hierarchical vocabularies offer an even more scalable solution to this recognition problem.

The larger the database, the more prone to mismatches the recognition is. To address this issue, Chum et al. (2007) used a query-expansion method to increase the recall when detecting objects by its appearance. For this, all initial candidates are re-ranked using an affine homography, so that the database is queried again with the best first candidates.

Recognizing objects modeled as single images is useful from the point of view of computer vision; however, for visual SLAM it is more advantageous to deal with 3D objects that can be located in the 3D space. Nevertheless, all these approaches show the reliability of visual vocabularies to perform object recognition, which can be translated into the 3D domain.

3D Object models. A common approach to recognize objects in single images and compute its 3D pose requires modeling the object as a collection of local image features as SIFT or SURF associated to 3D coordinates in an object frame. Then, given an image, its local features are matched against those from the model, and the pose is obtained by solving with random sample consensus (RANSAC) (Fischler & Bolles 1981) the perspective-n-point (PnP) problem. This is a classical problem in photogrammetry consisting in determining the location in space from which an image was obtained by recognizing a set of control points (Fischler & Bolles 1981). Matching is usually speeded up by discretizing feature descriptors into clusters with some of the visual vocabulary techniques described above.

Following the line of the current state of the art (Collet et al. 2011, Rublee et al. 2011, Sattler et al. 2011, Pangercic et al. 2011, Grundmann et al. 2011, Hsiao et al. 2010), we compare the similarity between images and object models composed of 3D points attached to image descriptors. We do it with two different approaches to model the appearance of the objects: with independent sets of SURF features, and with a single bag of binary words built on ORB features, applying a policy of allowing several descriptors per 3D point to capture richer viewpoint information. The later differs from previous works in that we use binary features with a visual vocabulary composed of several trees of binary nodes.

Pangercic et al. (2011) build a large database by using a SIFT vocabulary tree trained with

the object images. They propose a method to build a vocabulary incrementally when new objects are added. In our approach, we use independent images to build a single vocabulary which does not need incremental building. In real scenarios, objects do not appear well separated from the background, but together with the rest of the scene clutter. To avoid background clutter, especially when detecting small objects, it is necessary to focus the recognition on salient areas in the images, named *regions of interest* (ROIs). To divide query images into ROIs, Pangercic et al. (2011) cluster the scene 3D point cloud provided by other devices, such as a Kinect camera, and back-project them into the image. To obtain ROIs, we rely on a novel combination of the clustering methods k-means++ (Arthur & Vassilvitskii 2007) and medoidshifts (Sheikh et al. 2007) which use the image as input only, consuming short time.

Hsiao et al. (2010) discretize the SIFT descriptor space into three levels in a hierarchical manner. They show the benefits of computing feature matches at all the levels, obtaining more putative correspondences which yield a higher object recognition rate. However, they do not select an object candidate to match, so that the RANSAC step may be overburdened, leading to a large increase of the execution time when considering several objects. When using bags of words, we are able to retrieve the best candidate model when querying the database, avoiding an exhaustive comparison with all the models.

The approach by Sattler et al. (2011) can also handle large databases. They quantize the SIFT descriptor space with k-d trees (Muja & Lowe 2009), and compute correspondences between those descriptors discretized as the same visual word only. In our case, we use a direct index to set the discretization level at which correspondences are computed, so we can consider coarser levels to be less restrictive. Their approach is not aimed at semantic mapping, but to recognition of buildings in high-resolution pictures. Thus, although they speed up feature matching, they do not focus on real-time performance with low-resolution cameras.

Execution time. The works above do not provide recognition at video frequency due to the difficulties pointed out: extraction of regions of interest, selection of candidate model and computation of corresponding points to obtain a rigid-body transformation. We have to add to them the first time-consuming step: the extraction of robust features, such as SIFT or SURF, which can cope with scale, rotation and other changes. The computation of these features takes more than 100ms per image. Rublee et al. (2011) showed that their binary feature ORB can be used for object recognition at a fraction of the time required by SIFT or SURF, by performing simple feature matching with LSH tables with independent views of the objects. In this work, we make use of ORB in our 3D object recognition approach based on bags of words, obtaining successful and fast results using trees with a direct index instead of LSH tables. In addition, we comprise the multiple views of the object as a single bag-of-words vector, avoiding redundant comparisons with features seen from several points of view.

There is usually a trade-off between the execution time spent to find the 3D pose of a recognized object in an image and its accuracy, so that an accurate pose has a higher computational cost. The works above focus mainly on accuracy for applications other than visual SLAM, such as grasping, so that execution time is not a priority. Their target is to obtain a precise location of the object from a single image. For that, they usually try lots of candidate locations to check which one matches the input image the best, requiring large execution time.

On the other hand, visual mapping provides plenty of images to be exploited in order to achieve high recall and accurate object location if the sequence is exploited as a whole. Visual SLAM is able to refine a noisy pose of a 3D object inserted in the map over time as the SLAM algorithm reduces the uncertainty of the map landmarks. Furthermore, in real high-rate video sequences there are frames in which recognition cannot be successful because objects are not distinguishable at all due to their distance, their pose, image artefacts, motion blur, noise, etc.

Thus, it is desirable to reject those frames quickly and process the following ones, where the detection may be possible, instead of spending time trying recognition and missing frames.

Our bag-of-words approach allows a scalable 3D object recognition. This is also achieved by Collet et al. (2011), who perform recognition at 300ms per 640×480 image with 91 objects by exploiting parallelism in a GPU/CPU architecture. Our proposal goes a step farther providing 3D object recognition at video frequency on a conventional CPU. This is very beneficial for robotic interaction tasks in general and for semantic mapping in particular, because although visual SLAM can be performed at video frequency (Klein & Murray 2009, Davison et al. 2007), this had not been achieved for object recognition.

Chapter 3

Appearance-based loop detection

3.1 Image database

An image codes a large amount of information. Even a low resolution 320×240 image contains dozens of thousands of pixels that are translated into a lot of kilobytes of computer memory. This entails a great difficulty when this information must be managed in real time. The way to ease this issue is to reduce the image to its most descriptive features doing without those pixels that provide little information, transforming later the features into *bags of words*, obtaining a sparse and compact representation of the image, suitable to be efficiently indexed in an image database.

3.1.1 Images as bags of words

Although local image features describe the most salient areas of an image and put up with some image transformations, images represented by their local features can still take large memory. For example, it is very common to describe a small 640×480 image with 300–500 point-based features. When using SIFT or SURF features, their descriptors take between 75 and 250KB, which is a large amount of data.

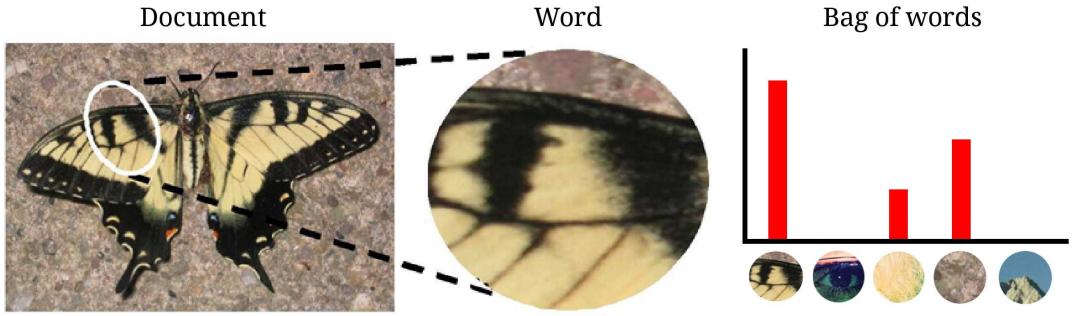


Figure 3.1: Example of text retrieval concepts applied to the visual domain. Images are visual documents whose local features are converted into visual words, allowing to represent their appearance as histograms of words. Credit: Nister & Stewenius (2006).

In order to mitigate this, image features are reduced by means of a *visual vocabulary* (Sivic & Zisserman 2003), drawing an analogy with the vocabularies firstly used for information retrieval in text documents (Baeza-Yates & Ribeiro-Neto 1999). In text retrieval, documents were considered sets of unordered words and represented as histograms of word occurrence frequencies. Similarities between two documents were made in terms of how many similar words they contained and how descriptive they were. These concepts are transfered to the visual domain by considering images as documents composed of *visual words* that represent their features, as illustrated in Figure 3.1. As in the text case, a visual vocabulary is an structure to represent a set of image features as a more simple histogram, allowing a compact representation with low storage requirements and fast image comparison. The visual vocabulary consists of visual words, which are a set of predefined descriptors. Any other feature can be converted into that visual word which is closest in the descriptor space. Then, the conversion of a set of image features results in a vector of visual words called *bag of words*. Each entry of an image bag-of-words vector corresponds with the value that each vocabulary word is given in that image. Instead of frequency terms, visual bag-of-words values are set according to the descriptiveness of each word, depending on how often they are expected to occur in images. This information can be obtained by counting the times each word appears in a set of heterogeneous training images. Following the text retrieval example, words such as prepositions and articles would obtain a low weight because they are very common and do not provide salient information about the content of a certain text document.

Visual vocabularies usually comprise between 10K and 1M words. Since this number is remarkably larger than the words extracted from an image, bag-of-words vectors use an sparse representation to avoid storing empty entries. As a result, the appearance of an image with 500 features will be represented with a sparse vector of fewer than 500 floating-point values.

3.1.2 Visual vocabularies from descriptor space clusters

A visual vocabulary is created by dividing the descriptor space into a finite set of enumerated clusters, so that any point in the space can be represented by the cluster containing it. At the expense of approximating, this mechanism provides a twofold benefit: it allows to represent a multidimensional descriptor vector as a single integer (the index of its cluster), and facilitates the match between similar features, since those that are close in the descriptor space are likely to lie in the same cluster.

The visual vocabulary is created offline by discretizing the descriptor space into W clusters, the so-called *visual words*. Sivic & Zisserman (2003) present a single quantization of the space into W clusters, but Nister & Stewenius (2006) proposes a hierarchical clustering which improves efficiency. In this case, the space is recursively clustered into k_w regions that become finer and finer, up to L_w clustering levels, yielding the $W = k_w^{L_w}$ final clusters. This makes the vocabulary be structured as a tree, as the example shown in Figure 3.2. With the clustering done by Sivic & Zisserman (2003), a point in the descriptor space must check W clusters to select the one to which it belongs. The advantage of the method proposed by Nister & Stewenius (2006) is that it permits decreasing the number of comparisons to $k_w \cdot L_w$, because only a subset of clusters are considered at each step. This results in a more efficient approach to convert features into words, which can deal with larger vocabularies to achieve higher retrieval quality.

To build the hierarchical vocabulary tree, we extract a rich set of features from some training images, independently of those processed later. The descriptors extracted are first discretized into k_w clusters by using the k-means++ algorithm (Arthur & Vassilvitskii 2007). This algorithm starts picking k_w random and well-distributed samples as cluster candidates, which are represented as single points in the descriptor space. The rest of the samples are then associated

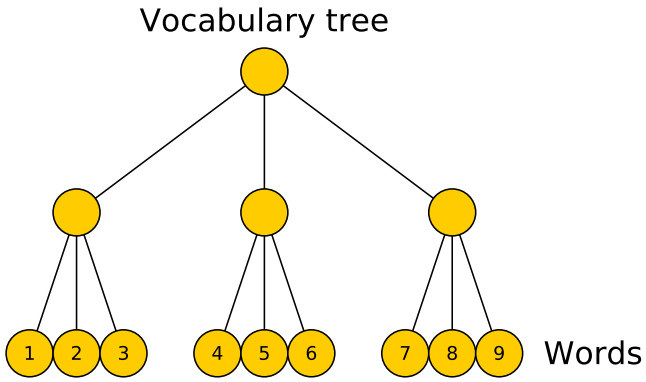
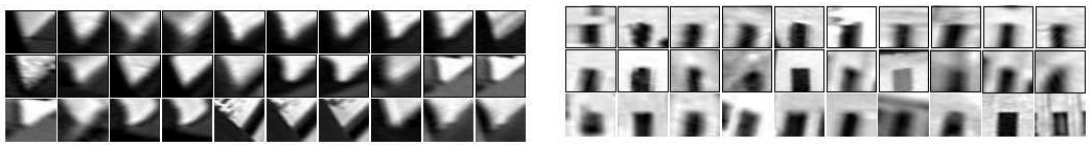


Figure 3.2: Toy example of a vocabulary tree with branching factor $k_w = 3$ and $L_w = 2$ depth levels. Each node represents a cluster in the descriptor space, and each level, a discretization factor, from coarse (closer to the root) to fine. Leaves are the words of the vocabulary.



(a) Word example 1

(b) Word example 2

Figure 3.3: Example of image patches that create visual words. As a result of the descriptor space clustering, visual words represent similar featurue descriptors. Credit: Sivic & Zisserman (2003).

to their closest cluster candidates, and new clusters are computed as the centroids of each group. This process is repeated iteratively until clusters converge. The final clusters form the first level of nodes in the vocabulary tree. Subsequent levels are created by repeating the k-means++ operation with the descriptors associated to each node, up to L_w levels. We finally obtain a tree with $W = k_w^{L_w}$ leaves, which are the words of the vocabulary. Figure 3.3 illustrates the results that may be obtained after clustering the descriptor space. Words are obtained as the centroid of the descriptors of image patches with similar appearance.

3.1.3 Word weighting and image conversion

Each word of the vocabulary is given a weight according to its relevance in the training corpus, decreasing the weight of those words which are very frequent and, thus, less discriminative. During the tree building, we weight each word w_i with its inverse document frequency (idf):

$$\text{idf}(i) = \log\left(\frac{N}{n_i}\right) \tag{3.1}$$

where N is the number of training images, and n_i, the number of images which contain the word w_i.

To convert an image I into a bag-of-words vector $\mathbf{v} \in \mathbb{R}^W$, the descriptors of its features traverse the tree from the root to the leaves, by selecting at each level the intermediate nodes

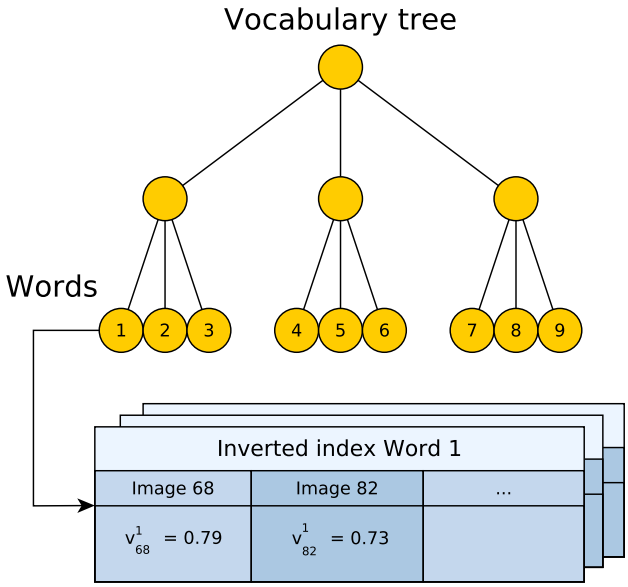


Figure 3.4: Example of image database composed of a vocabulary tree and an inverted index. The inverted index stores the weight of the words in the images in which they appear.

that minimize a distance function, such as the Euclidean one. This allows us to calculate the term frequency (*tf*) of each word in this image:

$$\mathrm{tf}(i, I) = \frac{n_{iI}}{n_I} \tag{3.2}$$

where n_{iI} stands for the number of occurrences of word w_i in image I, and n_I, for the number of words in I. The i-th entry of $\mathbf{v}$ is then given the value $v^i = \mathrm{tf}(i, I) \times \mathrm{idf}(i)$, obtaining the *term frequency – inverse document frequency* (*tf-idf*) weight as proposed by Sivic & Zisserman (2003). Bag-of-words vectors are finally normalized to mitigate the effects of dissimilar number of words between vectors when these are compared (Nister & Stewenius 2006). We use the L_1-norm since it yields better results than the L_2-norm according to Nister & Stewenius (2006).

Sivic & Zisserman (2003) analyze the relative merits of the *tf* and *idf* terms, and show the benefits of the complete *tf-idf* weight to increase retrieval accuracy in comparison with using just the *term frequency* factor or binary bag-of-words vectors, where entry values v^i are set either 1 or 0 to indicate the presence of a word in an image.

In information retrieval, words which are very common among text documents (such as prepositions or articles) are ignored because they are hardly discriminative; these are named *stop words*. Sivic & Zisserman (2003) propose considering the most frequent visual words as stop words to suppress them and improve the image matching. However, as other authors (Yang et al. 2007), we did not obtain more accurate image matches when stopping the most frequent words in the vocabularies of our initial tests, so we do without stop words in this work.

3.1.4 Image database for efficient indexing

We build a database to index images collected when traversing a trajectory. Our image database relies first on a visual vocabulary, since the lightweight representation yielded by bags of words enables to manage a large number of images. The next step to perform loop detection in a video sequence is to store images efficiently. The database must provide quick image comparisons and retrieval when queried in order to be reliable in large-scale mapping. An *inverted index* structure, also known as inverted file, is used for this purpose, since it has proved advantageous for image retrieval in general (Sivic & Zisserman 2003) and for loop detection in particular (Cummins & Newman 2011).

The inverted index is a structure that stores for each word w_i in the vocabulary a list of images I_t where it is present, being t the timestamp at which the video frame is acquired. This is very useful when we must query the database to compare a given image with all those stored before, since it allows to perform comparisons only against those images that have some word in common with the query one. In addition to image references, we augment the inverted indexes to store also the weight of each word in those images to enable a quick access. Thus, an inverted index stores for each word w_i a list of pairs $\langle I_t,\, v_t^i \rangle$, where v_t^i is the value of the i-th word in image I_t. Each time an image is added to the database, a new pair is added to the inverted indexes of the words it contains; they are accessed when the database is queried. The composition of the image database results as shown in Figure 3.4.

3.2 Loop candidate retrieval

To detect loop closures, we propose a method that we firstly presented in Piniés et al. (2010) and later improved in Cadena et al. (2010) and Cadena et al. (2012). It is depicted in Figure 3.5 and follows these steps for each image acquired as the robot moves:

1. The image is converted into a bag of words.
2. We search the database for the current image to retrieve those scenes whose similarity is high enough.
3. The image is added to the database by updating the inverted index.
4. We compute a normalized score that adapts to the appearance of each image to obtain a similarity value.
5. The match with the highest score is checked for temporal consistency with previous scenes to obtain a loop closing candidate.
6. Finally, if the best candidate passes a geometrical verification, the loop detection is accepted. This step is addressed in Section 3.3.

3.2.1 Database query

By means of the visual vocabulary we can measure the resemblance between images just by comparing their bag-of-words vectors. There are several statistical metrics one can apply to two vectors. Sivic & Zisserman (2009) make a comprehensive comparison between the properties of the statistical distances commonly used: L_1-norm, L_2-norm, Bhattacharyya coefficient (Bhattacharyya 1943), Kullback-Leibler (KL) divergence (Kullback & Leibler 1951) and χ^2 distance. Here, we use a score based on the L_1-norm distance, denoted d_{L_1}, between two images to

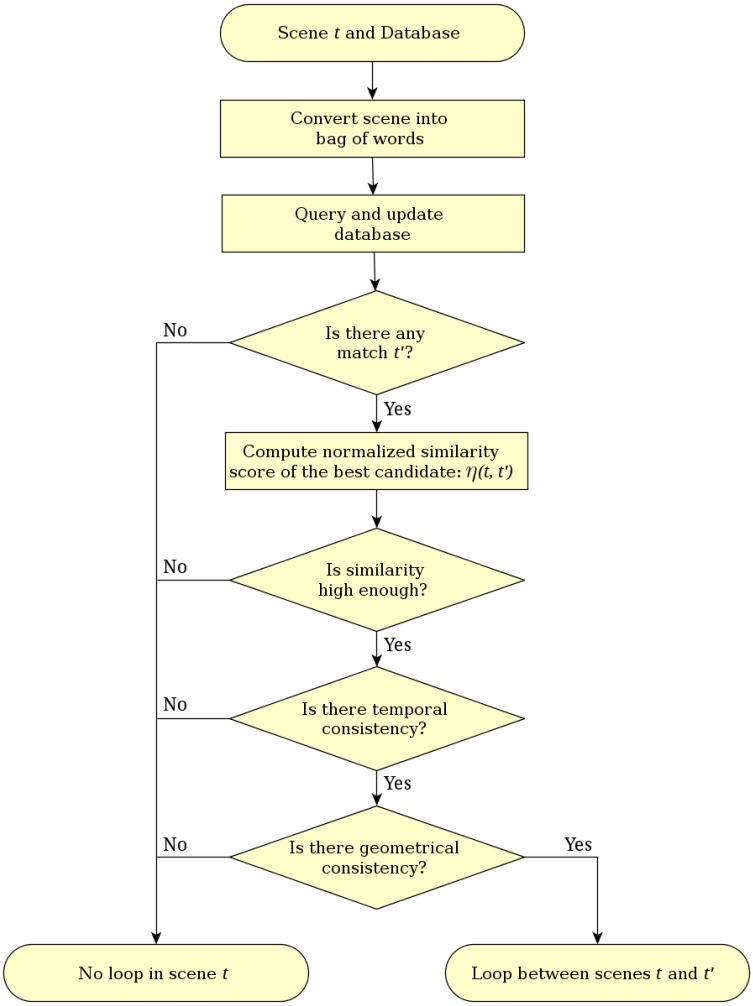


Figure 3.5: Scheme of our loop detection approach applied to each scene at time t in a sequence.

detect loops, as done by Nister & Stewenius (2006). Given two normalized bag-of-words vectors $\mathbf{v}$ and $\mathbf{w}$, the score $s_{L_1}(\mathbf{v}, \mathbf{w})$ is defined as

$$s_{L_1}(\mathbf{v}, \mathbf{w}) = 1 - \frac{1}{2} d_{L_1}(\mathbf{v}, \mathbf{w}) \tag{3.3}$$

$$= 1 - \frac{1}{2} \left\| \mathbf{v} - \mathbf{w} \right\|_1 . \tag{3.4}$$

The score $s_{L_1}(\mathbf{v}, \mathbf{w})$ varies in [0..1]. It obtains its highest value for maximum similarity, and 0 when there is no resemblance at all. As noticed by Nister & Stewenius (2006), we can express the distance d_{L_1} in terms of common vector entries to take advantage of the inverted index later. Let $\mathcal{V} = \{i \mid v^i \neq 0\}$, $\overline{\mathcal{V}} = \{i \mid v^i = 0\}$ be the set of indexes of words of $\mathbf{v}$ which are non-empty

Algorithm 1: Similarity score computation

Input : Vector $\mathbf{v}_t$, Inverted indexes $\mathcal{I}(i)$ for each i-th word
Output: Set of similarity scores $\mathcal{S} = \{s_{L_1}(\mathbf{v}_t, \mathbf{w}_{t_j}) \mid t_j \in [1,\, t - \tau_c)\}$

$\mathcal{S} \leftarrow \{s_{L_1}(\mathbf{v}_t, \mathbf{w}_{t_j}) \leftarrow 0 \mid t_j \in [1,\, t - \tau_c)\}$
foreach word $i \in \mathbf{v}_t$ **do**
  foreach $\langle t_j,\, w^i_{t_j} \rangle \in \mathcal{I}(i)$ **do**
    $s_{L_1}(\mathbf{v}_t, \mathbf{w}_{t_j}) \leftarrow s_{L_1}(\mathbf{v}_t, \mathbf{w}_{t_j}) + \min\left(v^i_t,\, w^i_{t_j}\right)$
  end
end
return $\mathcal{S}$

and empty respectively, and analogously for $\mathcal{W}$ and $\overline{\mathcal{W}}$, then

$$d_{L_1}(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V} \cap \mathcal{W}} |v^i - w^i| + \sum_{\mathcal{V} \cap \overline{\mathcal{W}}} |v^i| + \sum_{\overline{\mathcal{V}} \cap \mathcal{W}} |w^i| \tag{3.5}$$

Since $\|\mathbf{v}\|_1 = \|\mathbf{w}\|_1 = 1$ and

$$\sum_{\mathcal{V} \cap \overline{\mathcal{W}}} |v^i| = \sum_{\mathcal{V}} |v^i| - \sum_{\mathcal{V} \cap \mathcal{W}} |v^i|, \tag{3.6}$$

$$\sum_{\overline{\mathcal{V}} \cap \mathcal{W}} |w^i| = \sum_{\mathcal{W}} |w^i| - \sum_{\mathcal{V} \cap \mathcal{W}} |w^i| \tag{3.7}$$

we obtain

$$d_{L_1}(\mathbf{v}, \mathbf{w}) = 2 + \sum_{\mathcal{V} \cap \mathcal{W}} |v^i - w^i| - |v^i| - |w^i| \tag{3.8}$$

which yields

$$s_{L_1}(\mathbf{v}, \mathbf{w}) = \frac{1}{2} \sum_{\mathcal{V} \cap \mathcal{W}} \left|v^i\right| + \left|w^i\right| - \left|v^i - w^i\right|. \tag{3.9}$$

Furthermore, if vector entries are non-negative, as ensured by *tf-idf*, we can write

$$s_{L_1}(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V} \cap \mathcal{W}} \min\left(v^i,\, w^i\right). \tag{3.10}$$

When the database is queried with a vector $\mathbf{v}_t$, the score s_{L_1} is computed for all those stored vectors $\mathbf{w}_{t_j}$ which have at least one word in common with $\mathbf{v}_t$. For that, the inverted index of each word in $\mathbf{v}_t$ is checked. Since bag-of-words vectors are sparse and the score s_{L_1} is defined in terms of common words, this operation is performed fast by simply accessing the values stored in the inverted indexes. Due to the large overlap between consecutive images, it is necessary to disallow local matches with images added to the database in the last τ_c seconds, because their similarity is always high but they do not represent any loop closure. The value of τ_c depends mainly on the frequency at which the images are acquired and their expected variability. All this process is summarized in Algorithm 1.

3.2.2 Exploiting video sequentiality

After querying the database, the image matches are ranked by their scores, and only the match $\langle \mathbf{v}_t, \mathbf{w}_{t'} \rangle$ that maximizes $s_{L_1}(\mathbf{v}_t, \mathbf{w}_{t'})$ is kept as a loop candidate. The range between which the score s_{L_1} varies is very dependent on the query image and the distribution of words it contains. Usually, query images with a higher amount of words yield higher scores s_{L_1}. This makes, for example, that the score between two images from the same place, with few words, may be lower than the score obtained after comparing two images taken at different places but with richer collections of words. This difficulty prevents from setting a threshold to discard matches between images with low resemblance.

For this reason, we propose the *normalized similarity score* η as a result of scaling the score s_{L_1} for a query vector $\mathbf{v}_t$ with its expected score.

$$\eta_{L_1}(\mathbf{v}_t, \mathbf{w}_{t'}) = \frac{s_{L_1}(\mathbf{v}_t, \mathbf{w}_{t'})}{s_{L_1}(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})} \tag{3.11}$$

Here, we approximate the expected score of $\mathbf{v}_t$ as $s_{L_1}(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$, where $\mathbf{v}_{t-\Delta t}$ is the bag-of-words vector of the image previously acquired in the sequence. Since consecutive images present a large overlap if they are gathered close in time, we consider the previous frame as the most similar image to the given one. Note that the score η could also be defined for any other distance metric and not just s_{L_1}. Those cases where $s_{L_1}(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$ is small (e.g. when the robot is turning or an image is blurred) can erroneously cause high scores. Thus, we skip the images that do not reach a minimum $s_{L_1}(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$ or a required number of features. This minimum score trades off the number of images that can be used to detect loops with the correctness of the resulting score η. We use a small value to prevent valid images from being discarded. We then reject the match if $\eta_{L_1}(\mathbf{v}_t, \mathbf{w}_{t'})$ does not achieve a minimum threshold, denoted α. Otherwise, the loop candidate is checked for temporal consistency.

We impose a temporal constraint to detect loops under the premise that a loop closure is sustained by several image matches taking place consecutively. Thus, a loop candidate between images at time t and t' is kept if there exist matches $\langle \mathbf{v}_t, \mathbf{w}_{t'} \rangle$, $\langle \mathbf{v}_{t-\Delta t}, \mathbf{w}_{t_1} \rangle$, $\langle \mathbf{v}_{t-2\Delta t}, \mathbf{w}_{t_2} \rangle$, ..., for a short time interval of τ_l seconds, that are pairwise consistent. There is consistency if the difference between consecutive timestamps t', t_1, t_2, ..., is small (i.e. within τ_d seconds). These temporal values are selected according to the movement speed of the robot and the depth of the scenes in the image sequences of our experiments. We later show that these values work successfully in other datasets as well. If the temporal consistency does not hold, the loop candidate is rejected. Otherwise, the next step is to fulfill a geometrical verification to finally accept the loop detection.

3.3 Loop verification

Up to this point, pairs of images $\langle \mathbf{v}_t, \mathbf{w}_{t'} \rangle$ that are loop candidates pass a similarity score $\eta_{L_1}(\mathbf{v}_t, \mathbf{w}_{t'}) \geq \alpha$ and a temporal constraint. In addition, the location of the features in the images can be checked for geometrical consistency in order to avoid false positive detections. In a real application, we can have one or more cameras depending on the robotic platform. In this work, we propose different methods to fulfill this geometrical check taking advantage of all the hardware resources available.

3.3.1 Epipolar geometry for a single camera

The epipolar geometry is the intrinsic projective geometry between two views, which does not depend on the scene, but just on the camera internal parameters and the relative pose of the views (Hartley & Zisserman 2004). This geometrical information is encapsulated by a 3×3 matrix called *fundamental matrix*.

We can make use of the epipolar geometry to find geometrical information between two single images representing a real loop. This yields a well-known method to verify the consistency of two loop candidate images, considered the gold-standard technique for monocular systems. This geometrical check consists in computing feature correspondences between the two images to calculate a fundamental matrix supported by a significant number of correspondences. If features are located in consistent locations in the two views, there must exist a fundamental matrix $\mathbf{F}$ that satisfies the relation $\mathbf{x'}^\top \mathbf{F} \mathbf{x} = 0$, being $\mathbf{x} = \lambda(x, y, 1)^\top$, $\mathbf{x'} = \lambda'(x', y', 1)^\top$ homogeneous coordinates of corresponding points from the loop closing images.

In order to find corresponding points in both images, we match pairs of features which are the closest ones in the descriptor space and that satisfy the neighbor-ratio condition proposed by Lowe (2004), by which their distance must be far below the distance to the second-closest feature. Given the set of features $\mathcal{F}$ and $\mathcal{F}'$ of two loop candidate images I_t and $I_{t'}$, a distance function d between two feature descriptor vectors, such as the Euclidean distance, and the acceptance distance ratio ε_c, usually set to 0. 6-0. 8, the set of correspondences $\mathcal{C}$ is

$$\begin{aligned} \mathcal{C} = \{\langle f_i,\, f'_j \rangle | \quad & f_i \in \mathcal{F},\ f'_j \in \mathcal{F}', \\ & \forall f_k \in \mathcal{F} \backslash \{f_i\},\ d(f_i, f'_j) < d(f_k, f'_j), \\ & \forall f'_k \in \mathcal{F}' \backslash \{f'_j\},\ \frac{d(f_i, f'_j)}{d(f_i, f'_k)} < \varepsilon_c \}. \end{aligned} \tag{3.12}$$

A fundamental matrix can be computed from corresponding points by the normalized 8-point algorithm (Hartley 1997). This consists in solving the homogeneous linear system yielded by extending the equations $\mathbf{x'}^\top \mathbf{F} \mathbf{x} = 0$ given by at least 8 pairs of points, on condition that the system is well conditioned. Due to measurement noise, the input can be a higher number of pairs to over-determine the system and obtain a solution that minimizes the total square error.

In practice, dozens of corresponding points are usually obtained in two actual loop closing images, but some are incorrectly associated (wrong correspondence matches). Those correspondences cannot be used to estimate the fundamental matrix and must be rejected. For that, we use the random sample consensus (RANSAC) by Fischler & Bolles (1981), a robust algorithm to select correct correspondences. It involves selecting iteratively a random subset of correspondences to estimate the model of a candidate fundamental matrix $\widetilde{\mathbf{F}}$ with the algorithm above. Then, it checks how well each model fits all data and selects the one that maximizes the number of correspondences whose reprojection error is below a threshold ε_r, known as *inliers*. The projection of a point $\mathbf{x}$ from one image I by means of the fundamental matrix yields an *epipolar line* in the other image I', denoted $\mathbf{l'}$, which represents the projected ray that goes from the optical center of the camera related to I to the real point that generated $\mathbf{x}$:

$$\mathbf{l'} = \mathbf{F} \mathbf{x}, \tag{3.13}$$

with form $\mathbf{l'} = (a', b', c')^\top$. We denote as $\mathbf{l'}^*$ the normalized value of the line:

$$\mathbf{l'}^* = \left(\frac{a'}{\sqrt{a'^2 + b'^2}}, \frac{b'}{\sqrt{a'^2 + b'^2}}, \frac{c'}{\sqrt{a'^2 + b'^2}} \right)^\top. \tag{3.14}$$

The reprojection error of each correspondence is given by the distance from each data point to its corresponding epipolar line. If data are normalized, this can be performed with a dot product operation. Then, the amount of inliers $\widetilde{n}$ of a candidate fundamental matrix $\widetilde{\mathbf{F}}$ is obtained as

$$\widetilde{n} = \left| \{ \langle f_i,\, f'_j \rangle \in \mathcal{C} \mid \left| \mathbf{x}'_j \cdot \left(\widetilde{\mathbf{F}} \mathbf{x}_i \right)^* \right| \leq \varepsilon_r \} \right|, \tag{3.15}$$

being $\mathbf{x}_i$, $\mathbf{x}'_j$ the normalized homogeneous coordinates of the key points of features f_i and f'_j. The final fundamental matrix is obtained from all the inlier correspondences yielded by the best $\widetilde{\mathbf{F}}$ given by RANSAC. The loop candidate images pass this geometrical check if the final fundamental matrix is supported by a minimum number of correspondences, typically 12 (Hartley & Zisserman 2004).

3.3.2 Epipolar geometry for multiple cameras

In Piniés et al. (2010) we presented CI-Graph SLAM, a full large-scale SLAM system based on conditionally independent submaps for a robotic platform with a trinocular camera rig (left, right and top cameras). The main advantage of conditionally independent submaps (Piniés & Tardós 2008, Paz et al. 2008) over independent ones is that common information between maps can be consistently shared and transmitted taking into account all the information available about a feature.

CI-Graph SLAM tracks Harris corners (Harris & Stephens 1988) and reconstructs the 3D points of those features which are found in several cameras. One camera is selected as the reference camera to initialize new features; in the implementation this is the right camera of the trinocular system. Using the known extrinsic calibration between cameras, the recently introduced feature is predicted in the other camera images where we perform an active search over an uncertainty region. The rigid transformation between cameras allows us to obtain the depth information of nearby features. For the rest of the steps of the SLAM algorithm, each camera predicts and updates features independently. Figure 3.6 shows an example of the system when building a local map along a university library of an indoor experiment. We can see how features in the map are predicted and searched over right, left and top images. A reconstruction is also shown both in top and lateral view for the resulting submap.

When an image point cannot be reconstructed in 3D because it has not been tracked for enough frames yet, it can be predicted in the other camera images by means of the epipolar line yielded by the fundamental matrix between the cameras. Given a set of n calibrated cameras C_1, $\ldots$, C_n mounted on fixed positions, and hence with known relative poses, we can compute the fundamental matrices between each pair of cameras $\mathbf{F}_{ij}$. The relation between the relative pose between two cameras (with rotation matrix $\mathbf{R}_{ij}$ and translation vector $\mathbf{t}_{ij}$) and the fundamental matrix $\mathbf{F}_{ij}$ that their views form is given by the *essential matrix* $\mathbf{E}_{ij}$. This is the specialization of the fundamental matrix to the case of normalized image coordinates when calibration is provided (Hartley & Zisserman 2004), which is defined as

$$\mathbf{E}_{ij} = \mathbf{K}_j^\top \mathbf{F}_{ij} \mathbf{K}_i = \mathbf{R}_{ij} \left[\mathbf{t}_{ij} \right]_\times , \tag{3.16}$$

where $\mathbf{K}_i$ encapsulates the intrinsic parameters of camera C_i (focal length f, optical center c),

$$\mathbf{K}_i = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}_i , \tag{3.17}$$

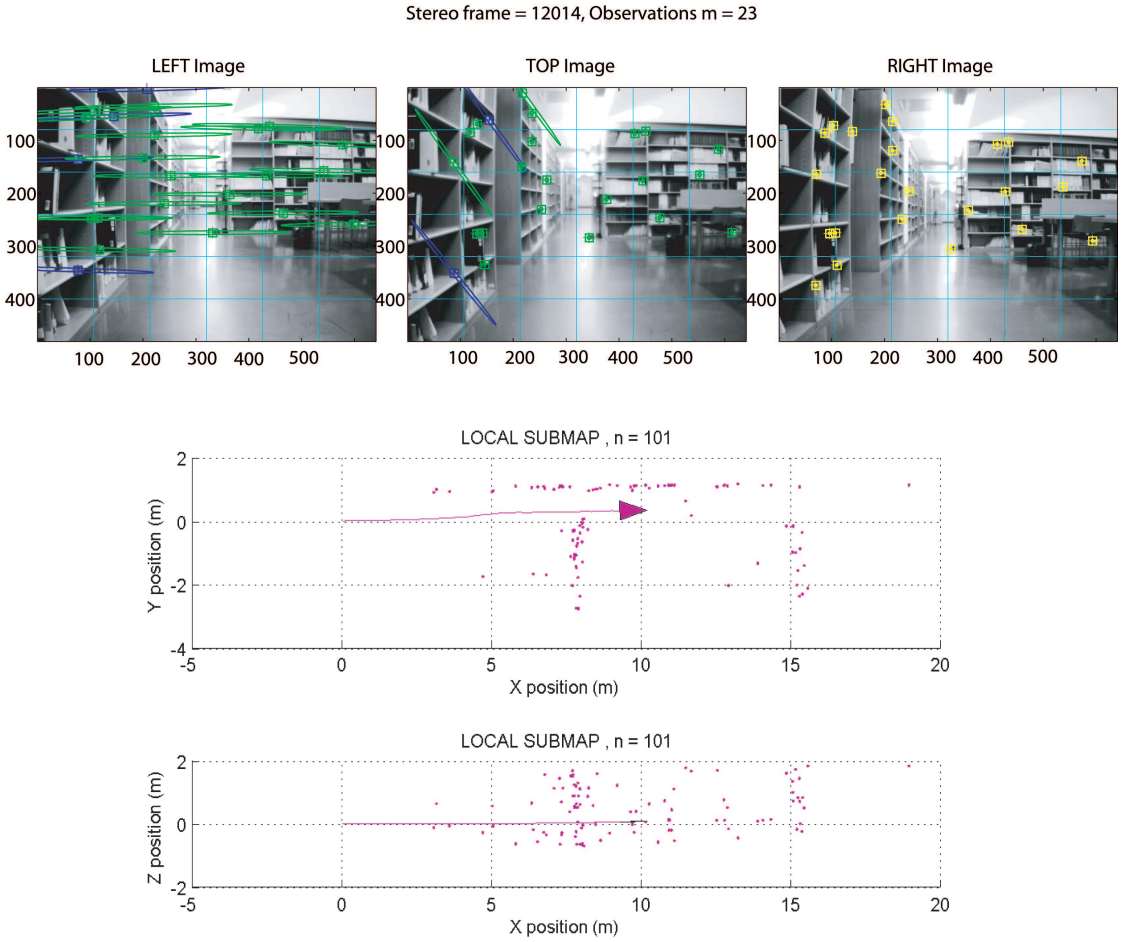


Figure 3.6: Top: CI-Graph SLAM system performing trinocular tracking. Large ellipses are produced after projecting recently initialized map features using the corresponding extrinsic calibration. Bottom: Top and lateral views of the local submap reconstruction.

and

$$[\mathbf{t}_{ij}]_\times = \begin{bmatrix} 0 & -c & b \\ c & 0 & -a \\ -b & a & 0 \end{bmatrix} \tag{3.18}$$

when $\mathbf{t}_{ij} = (a, b, c)^\top$. Thus, we can write

$$\mathbf{F}_{ij} = \left(\mathbf{K}_j^\top\right)^{-1} \mathbf{R}_{ij} \left[\mathbf{t}_{ij}\right]_\times \mathbf{K}_i^{-1} \tag{3.19}$$

The Harris features acquired at instant t' to build the map are tracked by CI-Graph SLAM. This system is accompanied by a feature management strategy that deletes non-persistent features to avoid an unnecessary growth in population. This means that not all those features acquired at t' survive some cycles later at time t. Note that the kind of features used to detect loops and to build the map does not need to be the same: the map is built with Harris corners, but we detect loops with SURF features (Bay et al. 2008). SURF features are well suited

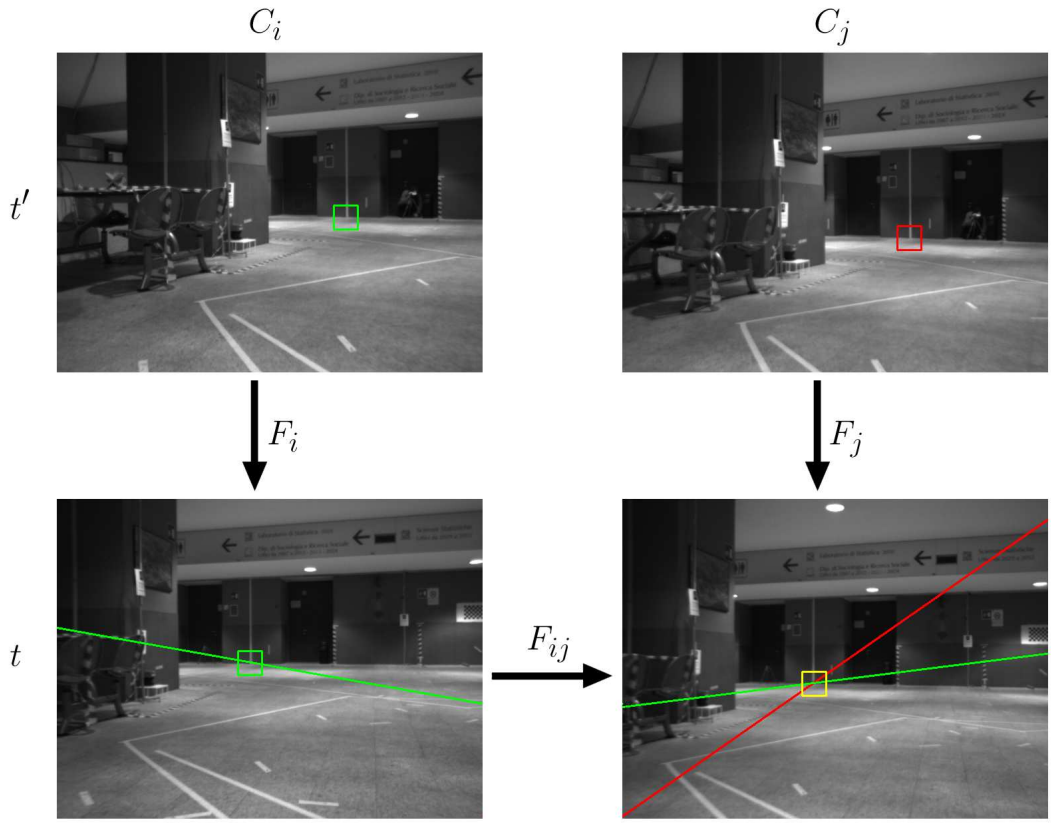


Figure 3.7: Multi-camera geometrical check performed by CI-Graph SLAM. The persistent Harris features present in each $I^i_{t'}$ are looked for in the corresponding I^i_t by means of the epipolar lines projected by $\mathbf{F}_i$. The features matched in each I^i_t are double-checked then for pairwise consistency between cameras. For that, I^j_t is searched for features found in I^i_t by $\mathbf{F}_{ij}$ and vice versa (one way is illustrated only).

to deal with perspective and scale changes, being able to detect loops in difficult cases. On the other hand, Harris corners can be tracked with better accuracy (Tuytelaars & Mikolajczyk 2008), which makes them a better option to build SLAM maps with little noise in the reconstructed 3D points.

In this multi-camera context, we can apply the geometrical check based on epipolar geometry to each camera individually, but we can also benefit from the known extrinsic calibration to check for geometrical consistency among the cameras in the rig. In addition, we can take advantage of the features tracked by CI-Graph SLAM to make the verification more robust. In order to make sure the loop candidate retrieval is supported by persistent features and not by spurious ones, we check how many SURF features of those that provided the detection coincide with Harris points tracked by the CI-Graph SLAM algorithm. Putting these two aspects together, the geometrical check results as follows.

When a loop candidate is obtained between time t and t' by means of SURF features got from one of the images of the camera system, we get images I^i_t and $I^i_{t'}$ from two or more cameras

C_i, $i = 1, 2, \ldots$ Then, we compute the fundamental matrices $\mathbf{F}_i$ between images I_t^i and $I_{t'}^i$ with the SURF features, as explained in Section 3.3.1. If any of these matrices cannot be computed, the geometrical check fails.

By means of $\mathbf{F}_i$, we are able to search I_t^i for Harris corners that appeared in $I_{t'}^i$ and are still alive in the SLAM map, as shown in Figure 3.7. This search is performed by normalized cross-correlation of 15×15 patches extracted along the epipolar lines yielded by $\mathbf{F}_i$ for each Harris feature in $I_{t'}^i$. If the loop candidate is valid, enough consistent features must be found. Since several cameras are available, we double-check the Harris features for pairwise consistency between cameras. Given any pair of cameras C_i and C_j, we check if those persistent Harris corners found in I_t^i and I_t^j appear in consistent locations in the image, according to the epipolar lines generated by $\mathbf{F}_{ij}$ and $\mathbf{F}_{ji}$, computed in advance as explained above. This is illustrated by Figure 3.7 as well.

The loop candidate images pass this geometrical check if the final number of corresponding features is high enough. The more pairs of cameras considered, the more restrictive check results. For CI-Graph SLAM, using just the pair of cameras that presented the widest baseline sufficed to obtain robust results without overburden the system, requiring 8 final correspondences. Note as well that when this geometrical check is passed, the data association between map features at time t and t' is obtained, allowing any SLAM system to add a constraint to correct the map.

3.3.3 Conditional random fields for stereo cameras

When two calibrated cameras are mounted on a stereo rig, we can obtain a *disparity image* with 3D information by triangulation of points appearing in both images. This metric data can be incorporated to the loop verification to obtain a more robust geometrical check (Majdik et al. 2011, Konolige et al. 2010). In this work, we propose computing descriptors from 3D data in addition to image data to reason about them by means of conditional random fields (CRFs), a probabilistic undirected graphical model first developed for labeling sequence data (Lafferty et al. 2001). For that, we use Cadena's (2011) CRF-Matching, an algorithm able to jointly reason about the association of features, presented working together with our loop detection approach in Cadena et al. (2012). CRF-Matching was recently proposed for matching 2D laser scans (Ramos et al. 2007) and matching image features (Ramos et al. 2008). Cadena (2011) extends CRF-Matching to reason about the association of data provided by the stereo camera system in both image space and in 3D space. This allows our system to consider all information provided by a stereo pair, coming from both near and far objects.

Figure 3.8 shows the modified loop detection algorithm when including CRF-Matching as geometrical verification, with the new steps highlighted. The first modification takes place before applying the CRF-Matching. After verifying the temporal consistency of the loop candidate, its normalized similarity score η_{L_1} is checked again to see if it is reliable enough not to perform any further verification, i.e. skipping the CRF-Matching. For that, we set a threshold α^+ from which we directly accept the loop candidate by its appearance if $\eta_{L_1} \geq \alpha^+$. We do this because the computation of the CRF descriptors and likelihoods, which we explain later, can be time-consuming.

If the similarity score η_{L_1} is not trustworthy enough to confirm the loop candidate, we run the CRF-Matching algorithm proposed in Cadena's (2011) Ph.D thesis. Given the two images I_t and $I_{t'}$ of a loop candidate, CRF-Matching is applied in three steps:

1. Two graphs $G_{3\mathrm{D}}$ and G_I are created from 3D points and image features of I_t and $I_{t'}$. These are used to compute geometrical and appearance descriptors.

2. Conditional random fields infer the probability of those graphs matching.

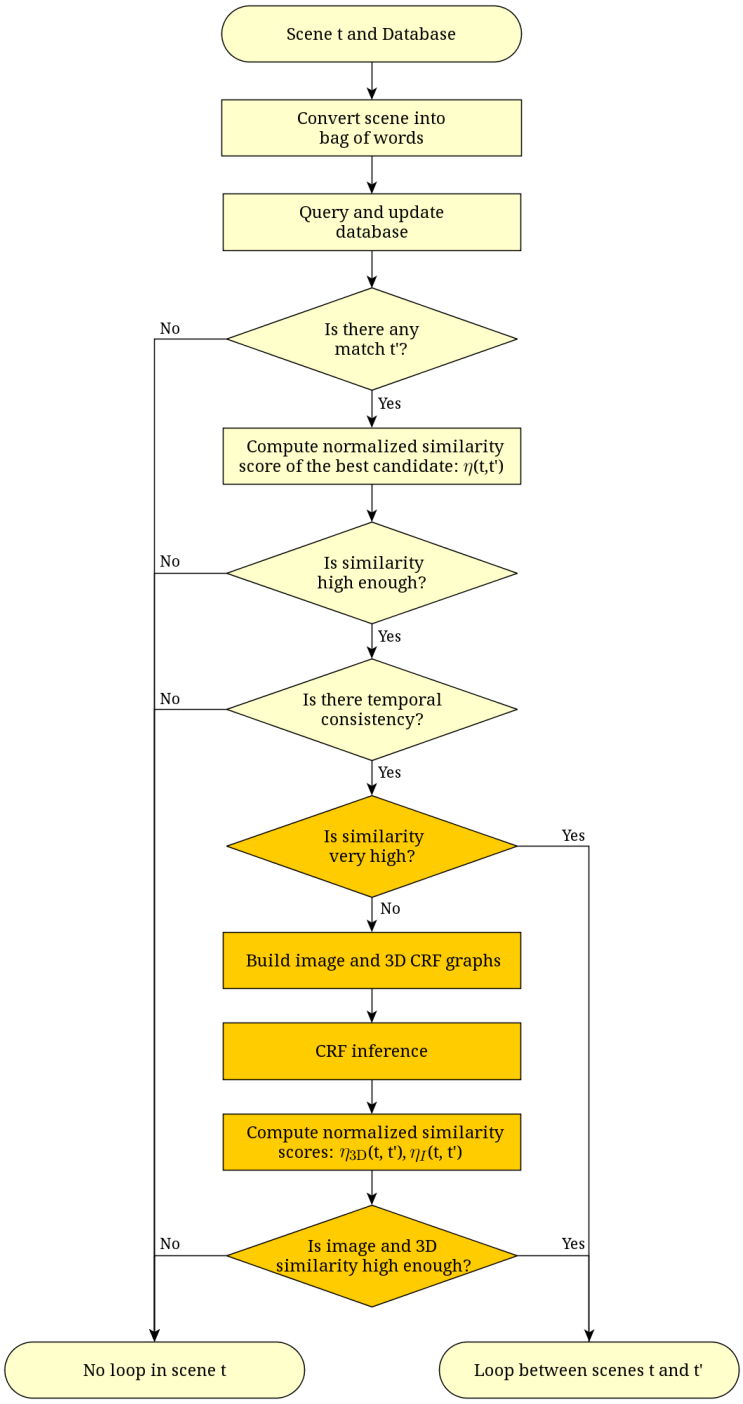
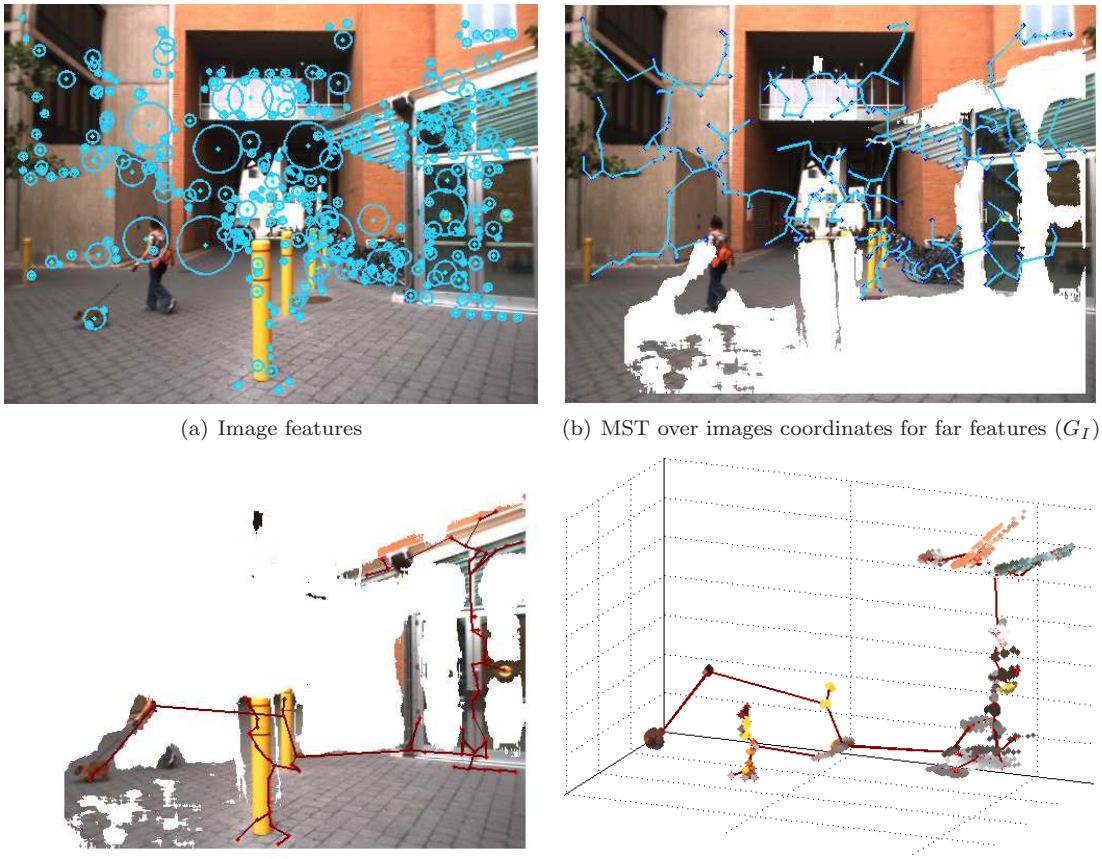


Figure 3.8: Scheme of our loop detection approach with the CRF-Matching verification (darker nodes).

3. Two normalized scores, η_{3D} and η_I, are created from the matching probability obtained before, taking image sequentiality into account. The loop is verified by comparing η_{3D} and η_I with the control parameters α_{3D} and α_I and checking whether the desired level of similarity is reached. This step was firstly proposed together with our loop detection algorithm in Cadena et al. (2012).

For the reader's convenience, we detail the most relevant steps of the CRF-Matching algorithm as we use it in our work. For further detail, please see Appendix A.

As graph structure for the CRFs in our problem, we propose the use of the minimum span-



(a) Image features

(b) MST over images coordinates for far features (G_I)

(c) MST over 3D metric coordinates for near features (G_{3D})

(d) 3D visualization of G_{3D}

Figure 3.9: Scene from an outdoor environment. In each scene we get the features over one image of the stereo pair 3.9(a), and compute the two MSTs: one for features with 3D information (near features), and the other for the remaining ones (far features). On 3.9(b), we show the graph for far features (G_I) in blue, in dark red the graph for near features (G_{3D}) on 3.9(c). We apply the CRF-Matching over both graphs. The MST of G_{3D} is computed according to the metric coordinates, here projected over the images only for visualization. On 3.9(d), we show G_{3D} in metric coordinates with the 3D point cloud (textured) of each vertex in the tree. The MST gives us an idea of the dependencies between features in a scene, and enforce the consistency of the features association between scenes.

ning tree (MST), where vertices are the features detected in the images, and edge weights are Euclidean distances between them. Because we code near information in the 3D metric space, and far information in image coordinate space, each type of visual information is represented in a separate graph, as shown in Figure 3.9: $G_{3\mathrm{D}}$ models the near objects, i.e. those pixels with dense information from the stereo, and hence with 3D information; and G_I, the far objects from pixels without disparity information. The nodes of the graphs are the feature locations, and the edges of the graphs result from computing the minimum spanning tree, according to the Euclidean distances between the pixel coordinates in the case of G_I, and between the 3D metric coordinates in the case of $G_{3\mathrm{D}}$.

The aim of the CRF-Matching algorithm is to compute the set of corresponding features between two scenes at t and t' with highest probability. This is denoted $p(\mathbf{x}|\mathbf{z})$, where $\mathbf{z}$ stands for the observed image features and their properties, and $\mathbf{x}$ is a vector of hidden states $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \rangle$ that codes for each of the n features of I_t the matching feature out of the m ones present in $I_{t'}$, i.e. $\mathbf{x}_i \in \{0, 1, 2, \ldots, m\}$, where the additional state 0 is the no-match state. This conditional distribution is written as a log-linear combination of *descriptor functions* $\mathbf{f}$:

$$p(\mathbf{x}|\mathbf{z}) = \rho \exp \left\{ \sum_{\mathbf{q} \in \mathcal{Q}} \mathbf{w}^T \cdot \mathbf{f}(\mathbf{x_q}, \mathbf{z}) \right\} \tag{3.20}$$

where $\mathcal{Q}$ is a set of subgraphs of $G_{3\mathrm{D}}$ and G_I, $\mathbf{w}_q$ a vector of weights of each descriptor function $\mathbf{f}$, and ρ is a normalization factor. The idea behind this equation is to exploit the information provided by features which are close in the image and the space, instead of treating each feature individually. For an extended explanation of this model, the reader is referred to Appendix A.

The CRF matcher can employ arbitrary local functions to describe shape, image properties, or any particular aspect of the data. These are the descriptor functions $\mathbf{f}$, which describe differences between shape (only for $G_{3\mathrm{D}}$) and appearance (for $G_{3\mathrm{D}}$ and G_I) of the features. The descriptor functions that we use are the following:

1. **Shape difference**: These functions capture how much the local shape of dense stereo data differs for each possible association. We use the geodesic, PCA and curvature distance.

   The *geodesic distance*, defined as the sum of Euclidean distances between points in the minimum spanning tree, provides information about the density of the neighborhood of each node of the graph. It can be calculated for different neighborhoods representing local or long-term shape information. Given 3D points $z^{3\mathrm{D}}_{t,i}$, $z^{3\mathrm{D}}_{t',j}$ and a neighborhood k, the geodesic distance is computed as:

   $$\mathbf{f}_{geo}(i, j, k, z^{3\mathrm{D}}_t, z^{3\mathrm{D}}_{t'}) = \left| \sum_{l=i}^{i+k-1} \| z^{3\mathrm{D}}_{t,l+1} - z^{3\mathrm{D}}_{t,l} \| - \sum_{l=j}^{j+k-1} \| z^{3\mathrm{D}}_{t',l+1} - z^{3\mathrm{D}}_{t',l} \| \right| \tag{3.21}$$

   where i and j correspond to the hidden state $\mathbf{x}_i$ that associates the feature i of the scene t with the feature j of the scene t'. The neighborhood k of $\mathbf{x}_i$ in the graph corresponds to all the nodes separated k nodes from $\mathbf{x}_i$. In our implementation, this function is computed for $k \in \{1, 2, 3\}$. A similar metric is used to match 3D laser scans by Anguelov et al. (2005).

   We also use Principal Component Analysis over the dense 3D point cloud that is contained within some spheres centered in the graph nodes (textured points in Figure 3.9(d)). The radius of these spheres is given by the key point scale provided by the feature extractor, such as the SURF detector. The *PCA distance* is computed as the absolute difference

between the variances of the principal components of a dense point cloud $z^{pca}_{t,i}$ in scene t and $z^{pca}_{t',j}$ in scene t':

$$\mathbf{f}_{PCA}(i,j,z^{pca}_t, z^{pca}_{t'}) = \left| z^{pca}_{t,i} - z^{pca}_{t',j} \right| \tag{3.22}$$

Another way to consider local shape is by computing the difference between the curvatures of the dense point clouds. This value is computed as:

$$\mathbf{f}_{curv}(i,j,z^{c}_t, z^{c}_{t'}) = \left| z^{c}_{t,i} - z^{c}_{t',j} \right| \tag{3.23}$$

where $z^c = \dfrac{3s_3}{s_1+s_2+s_3}$, and $s_1 \geq s_2 \geq s_3$ are the *singular values* of the point cloud of each node.

2. **Visual appearance**: These descriptor functions capture how much the local appearance from the points in the image differs for each possible association. We use the *feature distance*, i.e. the Euclidean distance between the descriptor vectors for each possible association:

$$\mathbf{f}_{descr}(i,j,z^{descr}_t, z^{descr}_{t'}) = \left\| z^{descr}_{t,i} - z^{descr}_{t',j} \right\| \tag{3.24}$$

Ramos et al. (2008) also include as descriptor functions the distances between the individual dimensions of the descriptor space. We disregard this information because in our training and validations data we did not find a significant improvement in accuracy in spite of the great increase of the size of the weight vector.

3. **Pairwise distance**: All functions described above are unary, in that they only depend on a single hidden state $\mathbf{x}_i$ in scene t. In order to generate mutually consistent associations it is necessary to define functions over the edges which relate the hidden states in the CRF to each other. The *pairwise distance* measures the geometrical consistency between the associations of *two* hidden states $\mathbf{x}_i$ and $\mathbf{x}_j$ and observations $z_{t,i}$, $z_{t,j}$ from scene t and observations $z_{t',k}$ and $z_{t',l}$ in scene t':

$$\mathbf{f}^{3D}_{pair}(i,j,k,l,z^{3D}_t, z^{3D}_{t'}) = \left| \|z^{3D}_{t,i} - z^{3D}_{t,j}\| - \|z^{3D}_{t',k} - z^{3D}_{t',l}\| \right|, \tag{3.25}$$

$$\mathbf{f}^{I}_{pair}(i,j,k,l,z^{I}_t, z^{I}_{t'}) = \left| \|z^{I}_{t,i} - z^{I}_{t,j}\| - \|z^{I}_{t',k} - z^{I}_{t',l}\| \right|. \tag{3.26}$$

The z^{3D} are in metric coordinates for G_{3D}, and z^I in pixels for G_I.

To check the geometrical consistency of the loop closing candidate, we compute the negative log-likelihood (Λ) from the most likely configuration of $\mathbf{x}$ (i.e., the maximum a posteriori, or MAP, estimation), as explained in Section A.3, between the scenes at time t and t', $\Lambda_{t,t'}$, and compare it with that obtained from the scene in $t - \Delta t$, $\Lambda_{t,t-\Delta t}$. The negative log-likelihood Λ^{3D} of the MAP association for G_{3D} provides a measure of how similar two scenes are in terms of close range, and Λ^I for G_I in terms of far range. Thus, in order to compare how similar the current scene is with the scene in t', $\Lambda_{t,t'}$, with respect to how similar the current scene is with the scene in $t - \Delta t$, $\Lambda_{t,t-\Delta t}$, we use again a normalized similarity score:

$$\eta_{3D} = \frac{\Lambda^{3D}_{t,t'}}{\Lambda^{3D}_{t,t-\Delta t}}, \tag{3.27}$$

$$\eta_{I} = \frac{\Lambda^{I}_{t,t'}}{\Lambda^{I}_{t,t-\Delta t}}. \tag{3.28}$$

Scores η_{3D} and η_I are compared to α_{3D} and α_I, control parameters of the level of similarity demanded, where a smaller α means a higher demand. By choosing different parameters for near and far information we can reach a balance between the weight of each source of information. The loop candidates pass the geometrical check if the corresponding scores are below these thresholds.

3.4 Experimental evaluation

We conducted different experiments to check the performance of our loop detection algorithm. For that, the first section of our experimental evaluation is devoted to explain the methodology to measure the quality of the results, and to introduce the datasets used. We start evaluating the ability of our proposal to retrieve pairs of image that are valid loop candidates. For that, we compare our system with a simple bag-of-words approach, where only the best candidate match is used, ignoring previous matches. We show the relative merits of using our normalized similarity score η and the temporal consistency. We also check how different vocabulary configurations (size and source of training images) affect the results. Finally, we compare our approach with all our proposed techniques for geometrical verification with two versions of the state-of-the-art FAB-MAP algorithm (Cummins & Newman 2008, Cummins & Newman 2011).

3.4.1 Methodology

In order to evaluate our loop detection system, there are some issues that have to be addressed first, such as how to measure the correctness of the results. Although these aspects are usually assumed to be of general knowledge, little detail is given in the literature. Here, we explain the methodology we followed to evaluate our system.

3.4.1.1 Datasets

Dataset	Description	T (m)	R (m)	S (m/s)	I (px × px)
Bicocca 2009-02-25b	Indoors, static	760	113	0.5	640 × 480
Bovisa 2008-10-04	Outdoor, static	1718	208	0.75	640 × 480
Bovisa 2008-10-06	Mixed, dynamic	1892	268	0.88	640 × 480
Malaga 2009 Parking 6L	Outdoors, dynamic	1192	162	2.8	1024 × 768

Table 3.1: Datasets used for evaluation. Legend: T, total dataset length; R, revisited length; S, average speed of the robot; I, image size.

We evaluated our system with sequences of images taken 1 second apart from three public datasets from the Rawseeds Project (Rawseeds 2007-2009) and with the Malaga parking lot 6L dataset (Blanco et al. 2009), as shown in Table 3.1. We use the first three of them to tune the parameters of our algorithm, whereas the last one is used just for evaluating the final configuration.

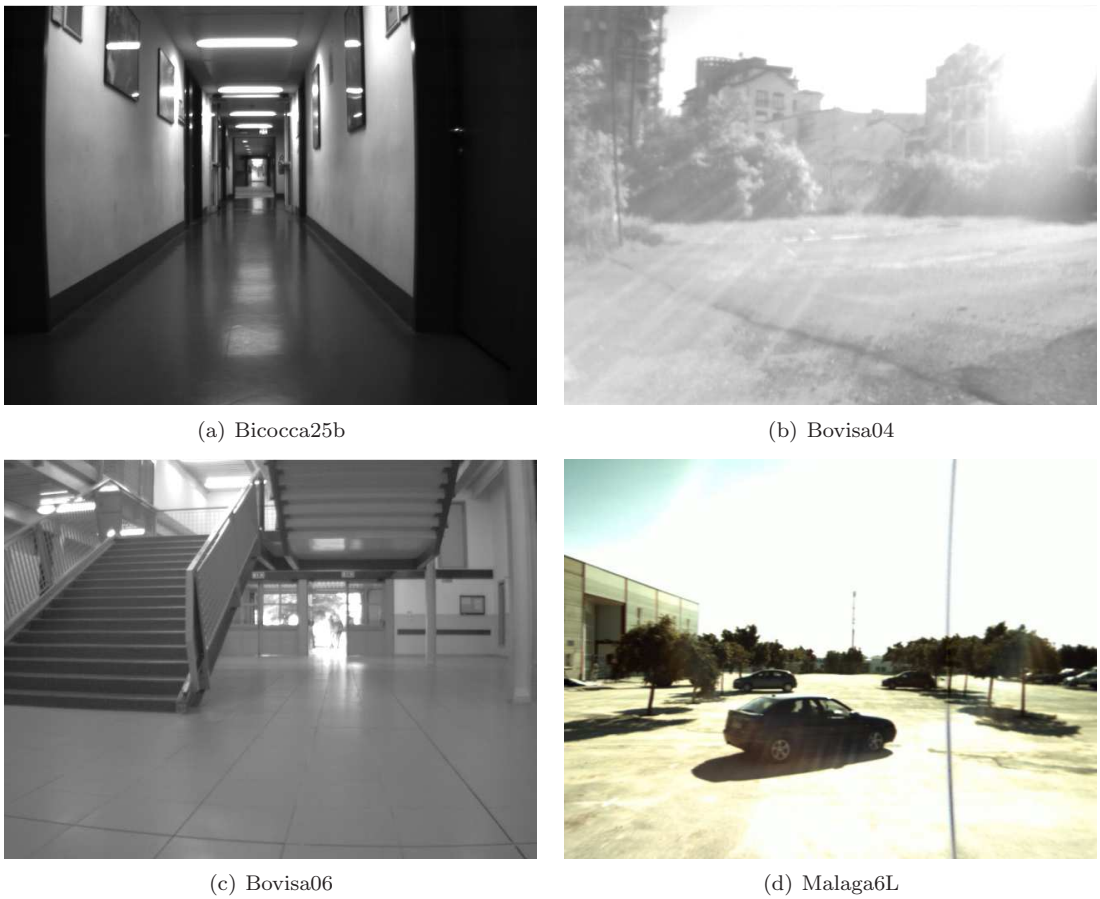
The Rawseeds data were collected by a robotic platform in different static, dynamic and mixed environments. The camera rig were formed by a stereo Videre Design STH-DCSG-VAR system (cameras on the left and right hand sides), with 18cm of baseline, and a third DCSG camera (on top) completing the trinocular system. These cameras acquired gray-scale images at 640 × 480px. Figure 3.10 shows some image examples. The indoor dataset, Bicocca25b, consists of 26335 trinocular image frames collected during 30 minutes at 15 FPS along a path of some 760m inside a university building where around 113m correspond to fragments of revisited

trajectory. We consider this dataset particularly challenging due to the intrinsic difficulty of extracting features in several places with lack of distinctive texture. In addition, the vehicle performs rough rotations in very narrow corners. In this dataset it is possible to identify 6 loops.

The outdoor dataset, Bovisa04, consists of 34173 trinocular frames, acquired in 38 minutes. During this time the vehicle travels across the surroundings of a university campus describing a trajectory of 1.718km with 9 main loops identified (5 loops smaller than 50m and 4 loops larger than 100m). The difficulty of this dataset stems from the little variability of the image background. This causes most of the features to lie in distant objects. In addition, part of the trajectory is performed in a rough terrain.

The mixed dataset, Bovisa06, contains 32240 trinocular frames acquired in 36 minutes across the same campus than the outdoor dataset, but with a different trajectory. In this dataset, the 1.892km long trajectory goes also inside buildings and describes 8 loop areas. The main difficulty of this dataset is due to the presence of people walking around, which produces some dynamic features which are not reliable to detect loop closures.

The Malaga6L dataset (Blanco et al. 2009) is a public dataset with 3474 pairs of images of 1024×768px size, acquired at 7.5fps with a stereo AVT Marlin F-131C camera, with 86cm of baseline, mounted on a car. The sequence depicts a parking area with a lot of similar-looking



(a) Bicocca25b (b) Bovisa04

(c) Bovisa06 (d) Malaga6L

Figure 3.10: Examples of datasets

trees in a sunny day, under illumination conditions that causes artifacts in the images, and some moving cars. This dataset presents 5 loop areas.

3.4.1.2 Ground truth

To measure the correctness of our results we compare them with a ground-truth reference. The Rawseeds datasets are provided with robot trajectory ground truth both for indoor and outdoor scenarios. Indoors, the nature of the ground truth solution relies on a fixed system formed by a bunch of cameras located in the area where the vehicle starts its trajectory and which is several times revisited. From this initial solution, an extended ground-truth solution on the full trajectory was obtained using the pose-graph based optimization algorithm developed by Grisetti et al. (2009). Laser scans are used to build the pose based graph where manual inspection and selection of scans are performed to guarantee a high accurate solution that matches the initial ground truth. Further details are publicly available at Rawseeds (2007-2009). Outdoors, partial ground truth is available from a GPS device with a precision of 0.9m. The Malaga6L dataset also provides a a centimeter-level ground truth from three RTK GPS receivers and one consumer-grade USB GPS receiver (Blanco et al. 2009).

We use the ground truth trajectory to build a ground truth of images depicting the same place and hence loop closures. For that, we manually create a list of the actual loop closures by comparing the images visually. Since these datasets contain thousands of images taken at high frequency, it is unfeasible to establish a direct connection between single images. Thus, this list is composed of time intervals, where each entry in the list encodes a query interval associated with a matching interval. The matched images depict the same place but are taken some distance apart, which is usually short. When the robot traverses the same place several times, matching intervals may appear more than once in the ground truth.

3.4.1.3 Correctness measure

We measure the correctness of the loop detection results with the *precision* and *recall* metrics. Precision is defined as the ratio between the number of correct detections and all the detections fired, and recall, as the ratio between the correct detections and all the loop events in the ground truth. Intuitively, precision summarizes the reliability of the answer yielded by the loop detector when it fires a loop closure detection, and recall, how likely it is to detect all the existing loops. A perfect loop detector, or any other classification system, would result in 100% precision and recall; i.e. all the loops, and only actual loops, are correctly detected.

A match fired by the loop detector is a pair of query and matching timestamps. To check if it is correct, i.e. a true positive (tp), the ground truth is searched for an interval that contains, or is close to contain, these timestamps. If they are not present in the ground truth, the detection is a false positive (fp). The total number of loop events in the ground truth (#total) is computed as the length of all the query intervals in the ground truth multiplied by the frequency at which the images of the dataset are processed. When a query timestamp is associated to more than one matching timestamp in the ground truth because of multiple traversals, only one of them is considered to compute the amount of loop events. Precision (P) and recall (R) are defined as

$$P = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fp}}, \tag{3.29}$$

$$R = \frac{\mathrm{tp}}{\mathrm{tp} + \mathrm{fn}} = \frac{\mathrm{tp}}{\#\mathrm{total}}, \tag{3.30}$$

where the false negatives (fn) are those loop events present in the ground truth but not detected as loops by the system.

3.4.2 Loop candidate retrieval

We started evaluating how each step of our loop candidate retrieval stage, hereafter "BoW", affects the correctness of the resulting matching candidates.

For this experiment, we created a bag-of-words vocabulary tree based on SURF features with $k_w = 9$ branching factor and $L_w = 6$ depth levels, yielding around 530K words. For this, we used a small amount of training data: 200 images uniformly distributed in time from another Rawseeds dataset, *Bovisa 2008-09-01* (Rawseeds 2007-2009). This is a static dataset that depicts a mix of indoor and outdoor areas. The trajectory of the robot in the Bovisa04 and Bovisa06 datasets has some overlap in location with the dataset used for training; this is not the case with the Bicocca25b dataset, which was collected in a different campus. However, training and testing datasets were acquired one month time apart. The rest of the parameters of the BoW detector are given in Table 3.2.

We defined two algorithms to show the benefits of exploiting sequentiality:

1. Simple bag-of-words approach: the matching image that simply maximizes the similarity score is retrieved.

2. Our proposed BoW algorithm with temporal consistency.

At the same time, we evaluate each of them both with the raw score s_{L_1} and the normalized similarity score η_{L_1}, defined in Section 3.2.1 and Section 3.2.2. As explained earlier, the score s_{L_1} is based on the L_1-norm distance between two bag-of-words vectors, whereas the score η_{L_1} is the score s_{L_1} scaled to the expected value between each input image and their correct matches.

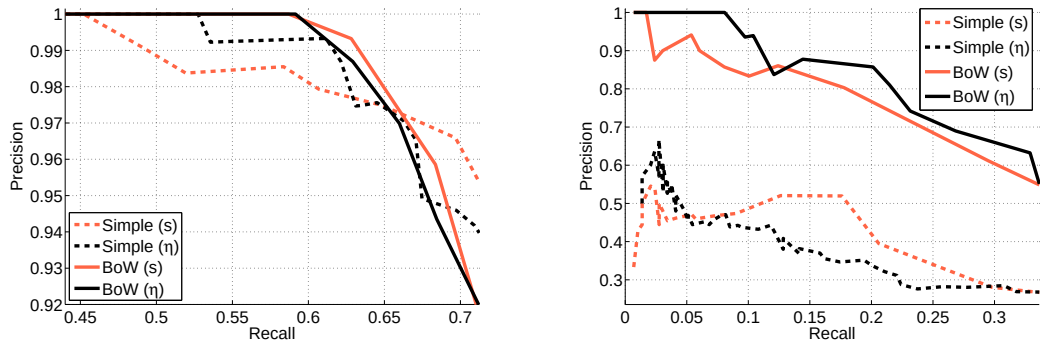
In Figure 3.11 we show the precision-recall curves yielded by each configuration as varying the minimum required score α. We can see that our BoW with temporal consistency and score η_{L_1} obtained the highest recall for 100% precision in the datasets.

Our BoW with temporal consistency outperforms the simple bag-of-words approach. This is specially noticeable in the outdoor Bovisa04 dataset, where the simple one is not able to achieve 100% precision, so that final results would be hardly reliable. The simple bag-of-words approach failed in this dataset because distant objects, such as buildings, are visible in many images, causing incorrect matches. Requiring temporal consistency reduces these cases because it is unlikely to obtain several consecutive matches with the same wrong place. This makes clear the usefulness of the temporal consistency.

Regarding the score, η_{L_1} attained higher recall for full precision than the score s_{L_1} both indoors and outdoors. In the indoor dataset, the behavior of both scores with our BoW approach with temporal consistency was similar, but the advantage of η_{L_1} is clear with the simple approach, when just the match with the maximum score is chosen. This shows that η_{L_1} is able to provide more discriminative scores than s_{L_1}.

Disallow local (τ_c)	20s
Temporal constraint length (τ_l)	4s
Distance between consecutive timestamps (τ_d)	2s
Min. $s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$	0.1

Table 3.2: BoW detector parameters for the experiments



(a) Bicocca25b (b) Bovisa04

Figure 3.11: Precision and recall obtained by our proposed BoW approach and a simple algorithm to select loop candidates, with no further geometrical verification.

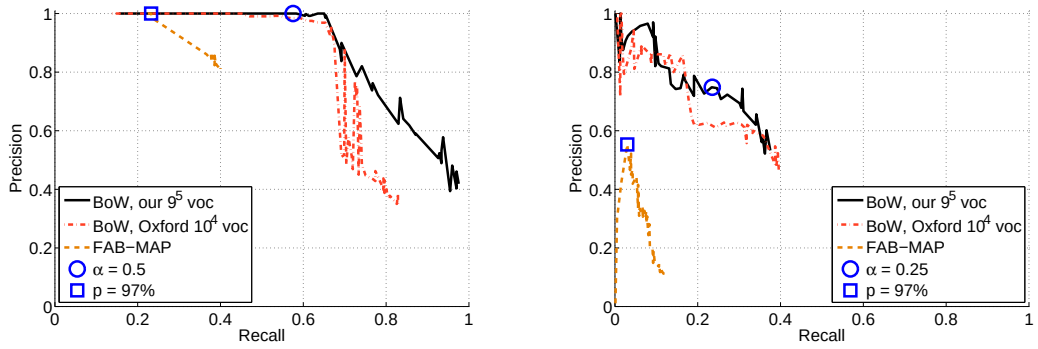
3.4.3 Results with CI-Graph SLAM

3.4.3.1 State-of-the-art comparison: FAB-MAP

Our proposed loop detection method was compared with the first version of FAB-MAP (Cummins & Newman 2008), a well known state-of-the-art technique, thanks to an implementation made available by the authors. FAB-MAP has proved to be very successful detecting loops in long trajectories for which it delivers very few false positives. The FAB-MAP system represents images with a bag of words and uses a Chow Liu tree (Chow & Liu 1968) to learn offline the words' co-visibility probability. FAB-MAP has become the gold standard regarding loop detection when using sequences of images with little overlap. Given the current image, FAB-MAP returns the probabilities of being in one of the previous locations or in a new one. The matched image is that which maximizes the probability as long as it reaches a threshold p that must be set by the user.

The FAB-MAP implementation available online provides two vocabularies: one was created from an indoor image sequence and consists of 10000 words; the other contains 10987 words calculated from an outdoor image sequence. As we do with BoW, we disallow local matches occurred to very recent images. For this experiment, we created two new vocabularies from richer training data with SURF features. The first one was created with 1300 images obtained from the same dataset used in Section 3.4.2, with a size of 58K words distributed in a tree of branching factor $k_w = 9$ and $L_w = 5$ depth levels. Since this vocabulary differs from those used by FAB-MAP, both on size and origin of training images, we also built a second vocabulary. This is used to make a comparison between results yielded by BoW and FAB-MAP when they use vocabularies of similar characteristics. The size of this second vocabulary was $k_w = 10$, $L_w = 4$, resulting in 10K words, and it was created from 1074 images collected in New College, Oxford, publicly available thanks to the authors of FAB-MAP (Cummins & Newman 2008).

We ran both BoW and FAB-MAP in the indoor and outdoor datasets (Bicocca25b and Bovisa04). In order to compare their results, we did not apply any geometrical check to the calculated matches at this moment. Figure 3.12 shows the precision-recall curves obtained by varying the parameters α for BoW (defined in Section 3.2.2) and p for FAB-MAP. We also show the curve when using our second vocabulary. In both cases, the shape of the BoW precision-recall curves is similar either using our Rawseeds vocabulary ($k_w = 9$, $L_w = 5$) or the Oxford



(a) Bicocca25b (b) Bovisa04

Figure 3.12: Precision-recall curves of our BoW method and FAB-MAP in two of the datasets, with no geometrical verification. The chosen working values of α and p of each method are also highlighted.

one ($k_w = 10$, $L_w = 4$). We can also see that BoW outperforms FAB-MAP in these datasets when using the Oxford vocabulary. This suggests that the size of the vocabularies and the origin of the training images are not decisive when comparing BoW and FAB-MAP. We see that in the indoor dataset both methods attain 100% precision, but it decreases quicker with FAB-MAP. The outdoor case is more challenging due to lighting conditions and the little variability of the background. Here, the performance of FAB-MAP drops, as shown in Figure 3.12(b). This leads us to guess that FAB-MAP is affected by the configuration of our cameras. The FAB-MAP model considers the environment as "a collection of discrete and disjoint locations". For that reason, in the work by Cummins & Newman (2008), images were taken perpendicular to the motion of the robot, so that the overlap was negligible. However, since our three cameras face forward, there are far-off objects (e.g. buildings) that persist for many frames, thus making scenes overlap and be less discriminative. It is easier for BoW to overcome those cases because our score η_{L_1} adapts to each scene, taking into account the similarity between consecutive frames.

We can also compare the results obtained in Figure 3.12 in terms of vocabulary size. The figure shows that the 58K-word vocabulary yields slightly better results than the vocabulary with 10K words. This improvement may be due to the little overlap between the training and the testing images in the Bovisa04 dataset; however, the enhancement is also present in Bicocca25b, where there is no overlap at all. Furthermore, looking back to Figure 3.11, where our system was tested with a vocabulary built with little training data (200 images), but organized in a larger tree with 530K words, we see that the recall for 100% precision is higher in the Bovisa04 dataset when using 530K words instead of 58K (3.1% and 1.4% respectively). This shows that the size of the vocabulary has more importance than the origin of the training data in our BoW approach when using SURF features, and that a large vocabulary, e.g. 530K words, tends to yield better results than smaller ones.

3.4.3.2 Verification with CI-Graph SLAM and epipolar geometry

In view of these results, in our CI-Graph SLAM system (Piniés et al. 2010), we used BoW with the Rawseeds 58K-word vocabulary and set, for each dataset, different values of the similarity acceptance threshold α. Figure 3.13 shows two examples of loops correctly detected in both datasets. In the outdoor dataset, $\alpha = 0.25$ exhibits a good precision-recall balance. Indoors, a

(a) Bicocca25b Input

(b) Bicocca25b Match

(c) Bovisa04 Input

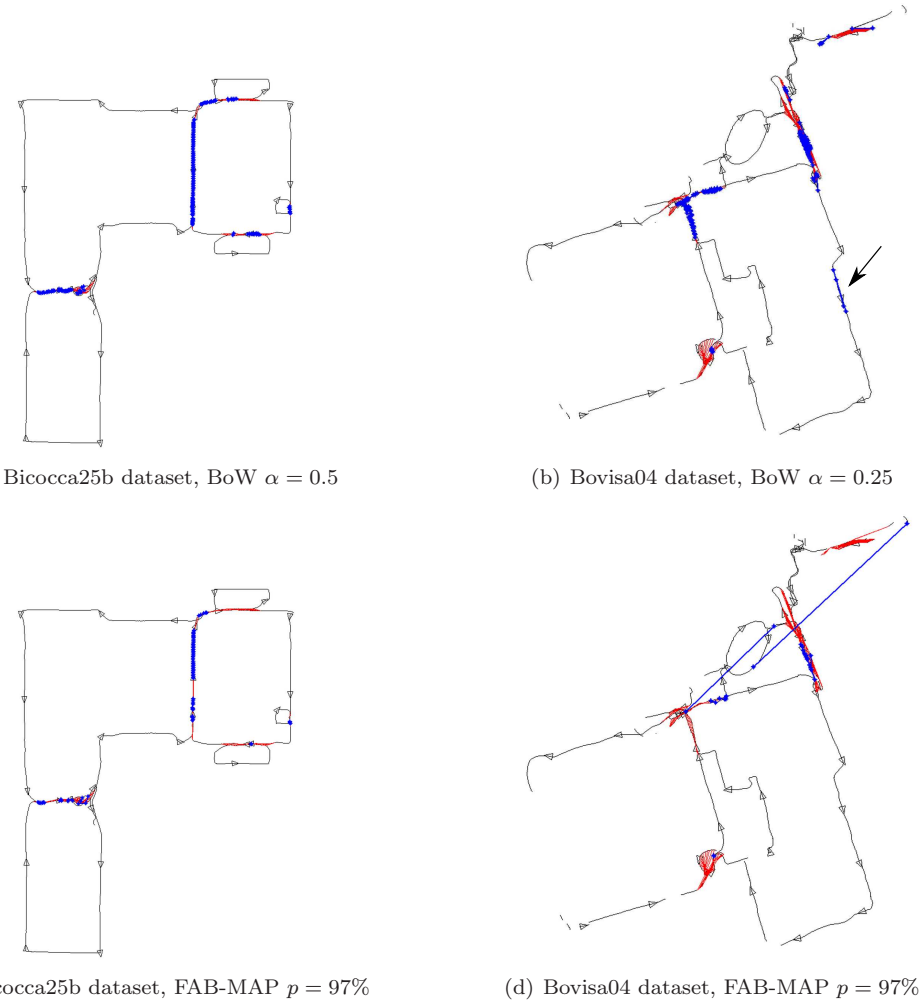
(d) Bovisa04 Match

Figure 3.13: Examples of successful loop detections

(a) Input image

(b) Discarded match

Figure 3.14: Match correctly discarded by BoW in the Bicocca25b dataset with $\alpha = 0.5$. A strict value of α allows to reject several perceptual aliasing cases like this one.



(a) Bicocca25b dataset, BoW $\alpha = 0.5$

(b) Bovisa04 dataset, BoW $\alpha = 0.25$

(c) Bicocca25b dataset, FAB-MAP $p = 97\%$

(d) Bovisa04 dataset, FAB-MAP $p = 97\%$

Figure 3.15: Loops detected by our BoW method with $\alpha = 0.5$ and $\alpha = 0.25$ and FAB-MAP with $p = 97\%$ in both datasets, with an epipolar geometry check perform on two cameras individually. In Bovisa04, BoW produces the two mismatches pointed out by the arrow. Black lines and triangles denote the actual trajectory of the robot; light red lines, loops from ground truth, and dark blue lines, places matched.

stricter value $\alpha = 0.5$ is required to avoid mismatches caused by perceptual aliasing in several similar-looking corridors, as shown in Figure 3.14. With these configurations and applying the epipolar geometry verification described in Section 3.3.1 to SURF features in the left and right cameras individually, we detected the loops shown in Figure 3.15. Ground truth matches are marked in red, whereas the detected loop closure positions are highlighted with blue lines on the trajectory path. We also present the matches obtained by FAB-MAP with the working value $p = 97\%$, where it attains its higher precision rate (see Figure 3.12). We can see that BoW yields no mismatches in the indoor dataset, and that most of the loop closures are detected. Outdoors, the largest loops were found several times, although recall is smaller. There are also two false detections due to matches between close positions, with no loop, on the right hand side of the

(a) Input image (b) Matched image

Figure 3.16: Example of mismatch obtained by FAB-MAP in the Bovisa04 dataset with $p = 97\%$

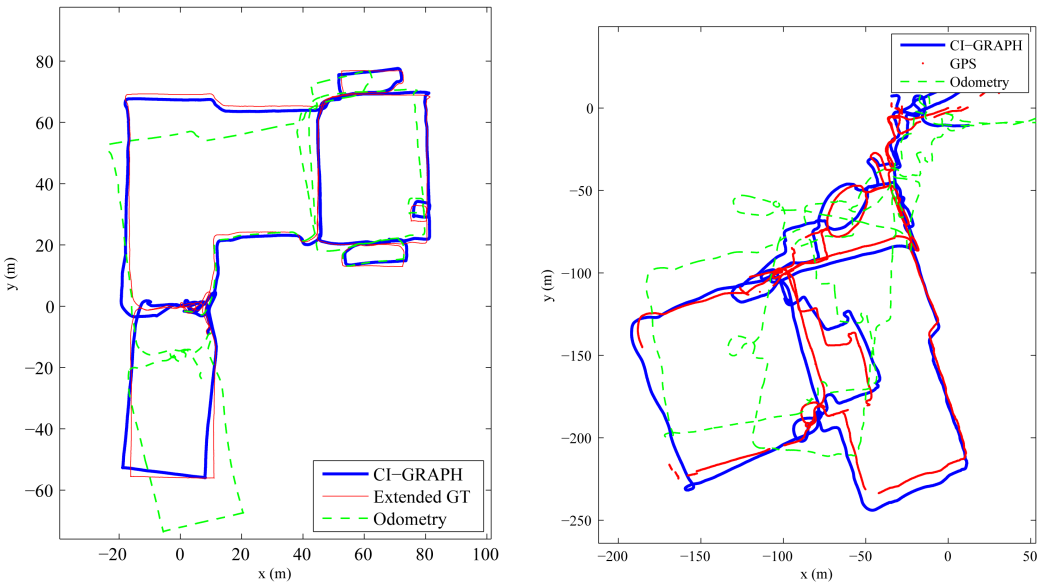
map (pointed out by an arrow in Figure 3.15(b)). In general, BoW detects more loop closures than FAB-MAP. Figure 3.16 shows one of the mismatches present in Figure 3.15(d), avoided by BoW.

When we performed the multi-camera cross-check verification proposed in Section 3.3.2, with Harris corners in addition to the SURF features, those mismatched loops produced by BoW were discarded. With no false positives, the complete CI-Graph SLAM system with the BoW loop detector resulted in the trajectories depicted in Figure 3.17. CI-Graph SLAM delivered a good precision for the indoor dataset achieving a mean absolute error of 1.64m and $\sim$ 1.66deg. In the outdoor experiment, after traversing 1.72Km, CI-Graph delivered a mean absolute error of 6.96m and a maximum error of $\sim$ 17m. This error is mainly on account of missing the detection of the last loop closure, taking place between the starting and end positions of the trajectory, in the top right hand side of the map in Figure 3.15(b). Although the loop is found when using the single camera epipolar constraint, it is falsely rejected when the cross-check is done, preventing the pose from correcting. This case is illustrated in the next section, in Figure 3.24, where we show the robustness of our probabilistic loop verification technique.

3.4.3.3 Execution time

In these experiments, the loop closing algorithm ran in a different thread than the CI-Graph SLAM algorithm. We used the OpenCV 1 library (OpenCV 2009) and our own initial implementation for the visual vocabulary, image database and loop detector for these purposes.

With this configuration, the loop closing algorithm ran in real time, at 0.57s per frame on average. The maximum peak was 1.25s on the datasets tested. This times included extracting SURFs, inspecting the bag-of-words tree, applying the loop detection algorithm and calculating the fundamental matrices. The most demanding step was the SURF extraction, that took about 0.41s per stereo image on average, whereas calculating the fundamental matrices took 0.13s. On the other hand, converting SURF features into words and maintaining the inverted index could be done in 28ms.



(a) Bicocca25b (b) Bovisa04

Figure 3.17: CI-Graph results for Bicocca25b and Bovisa04 experiments, compared with ground truth and odometry, when loops are detected with BoW and verified with the multi-camera epipolar constraint.

3.4.4 Results with CRF-Matching

In these experiments, we use the four datasets presented above and the same parameters for our system that those shown in Table 3.2. We reused the 530K word vocabulary presented in Section 3.4.2 and its 200 training images to learn the weights for the descriptor functions of the CRF matcher. For this, we obtained the SURF features from these images and selected those with 3D information from the dense point cloud given by the stereo system. Then, we ran a RANSAC algorithm over the rigid-body transformation between the images at time t and $t - \delta_t$. Since the stereo system has high noise in the dense 3D information, we selected $\delta_t = 1/15$s. The same procedure was done over the remaining SURF features with no 3D information, where we obtained the labels by calculating the fundamental matrix between the images. These two steps resulted in an automatic learning of the CRF labels. Although this automatic labeling can return some outliers, the learning algorithm has demonstrated being robust in their presence. We used the optimization based on the BFGS quasi-Newton method provided by MATLAB to find the weights that minimized the negative log pseudo-likelihood. In both G_{3D} and G_I, the weights obtained suggested that the most relevant functions in the CRF matcher were $\mathbf{f}_{desc}$ and $\mathbf{f}_{pair}$.

3.4.4.1 State-of-the-art comparison: FAB-MAP 2.0

We also compared our system with the second version of the state-of-the-art technique FAB-MAP 2.0[1] (Cummins & Newman 2011). In this version, their authors tackle the scalability of their system by defining a sparse approximation to the FAB-MAP model, suitable for implementation using an inverted index. Their system shows its reliability at obtaining 100% precision in

[1] Available online: `http://www.robots.ox.ac.uk/~mobile`

trajectories 70Km and 1000Km in length.

When using FAB-MAP 2.0, We used its indoor vocabulary for Bicocca25b, and the outdoor one, for the Bovisa04 and Bovisa06 datasets. To look into deeper detail the FAB-MAP 2.0 abilities, we tune its set of parameters in order to obtain the best performance in each experiment. For the reader's convenience, we give next a short description of those parameters:

- p: Probability threshold. The minimum matching probability required to accept that two images were generated at the same place.
- $P(\text{obs}|\text{exist})$: True positive rate of the sensor. Prior probability for detecting a feature given that it exists in the location.
- $P(\text{obs}|\neg\text{exist})$: False positive rate of the sensor. Prior probability for detecting a feature given that it does not exist in the location.
- $P(\text{newplace})$: Probability for new place. Prior probability to determine whether the last acquired image is a new place.
- σ: Likelihood smoothing factor. Factor for smoothing the likelihood values through consecutive places.
- *Motion Model*: Model Motion Prior. This biases the matching probabilities according to the expected motion of the robot. A value of 1.0 means that all the probability mass goes forward, and 0.5, that probability goes equally forward and backward.
- *Blob Resp. Filter*: Blob Response Filter. All the SURF points with a blob response below this threshold are discarded.
- *Dis. Local*: Disallow N local matches. Set the prior to be zero on the last N places.

We chose two parameter sets in order to obtain different results, as shown in Table 3.3:

1. The default parameter set that is provided by the authors. The probability threshold p is taken as 0.99, considering obtaining as few false positives as possible. When we use this configuration, we check the results yielded by FAB-MAP for geometrical consistency.
2. A modified parameter set is tuned to obtain the maximum possible recall at full precision. The idea behind of this tuning is to use as place recognition system only the FAB-MAP 2.0, without geometrical verification. For the outdoor dataset this parameter set is the same than the default set, only changing the probability threshold.

Since the last available version of the FAB-MAP 2.0 software does not implement the geometrical check described by Cummins & Newman (2011), we apply the epipolar geometry verification described in Section 3.3.1 when using the default parameter.

Firstly, we compared the correctness of our BoW detector with that of FAB-MAP 2.0 retrieving loop candidates, both with no geometrical verification. Figure 3.18 shows the precision-recall curves resulting in the three Rawseeds datasets. We obtained them by varying the minimum confidence value expected for a loop closure candidate of BoW, α, and the probability of acceptance p of FAB-MAP 2.0. We can observe that the curve of BoW dominated those of FAB-MAP 2.0, even without geometrical verification. As it was expected, when we choose carefully the parameters of FAB-MAP 2.0, the results we obtain are much better than when using the configuration by default. This is specially noticeable in the indoor dataset, where there were false positives in all the cases with the default parameters. This is due to the several similar-looking corridors and libraries this dataset presents.

3.4.4.2 Verification with CRF-Matching and epipolar geometry

We then added the geometrical verification stage to BoW and FAB-MAP 2.0 and compared the results of different systems:

- BoW with CRF-Matching, described in Section 3.3.3,
- FAB-MAP 2.0 with GC,
- BoW with GC.

We show an example of the results obtained by our BoW+CRF-Matching system in the Bicocca25b dataset in Figure 3.19. These were obtained by varying α and p, with α^+ fixed to 0.6. For the BoW+CRF-Matching system, we set the α_{3D} and α_I parameters of the CRF matcher in order to obtain 100% precision. We performed this test to compare recall with the state of the art. In view of results shown in Figure 3.18, we selected the working value $\alpha = 0.15$ to accept loop closing candidates. Note that this value of α is far below the values of 0.3 and 0.5 we used in the previous experiments in Section 3.4.3. Our aim is to show that we can relax the loop detection threshold, and find a value suitable for heterogeneous datasets, that can detect more loops, whereas relying on a more robust geometrical verification to disregard the spurious detections that appear in such a case. The rest of the parameters are shown in Table 3.4.

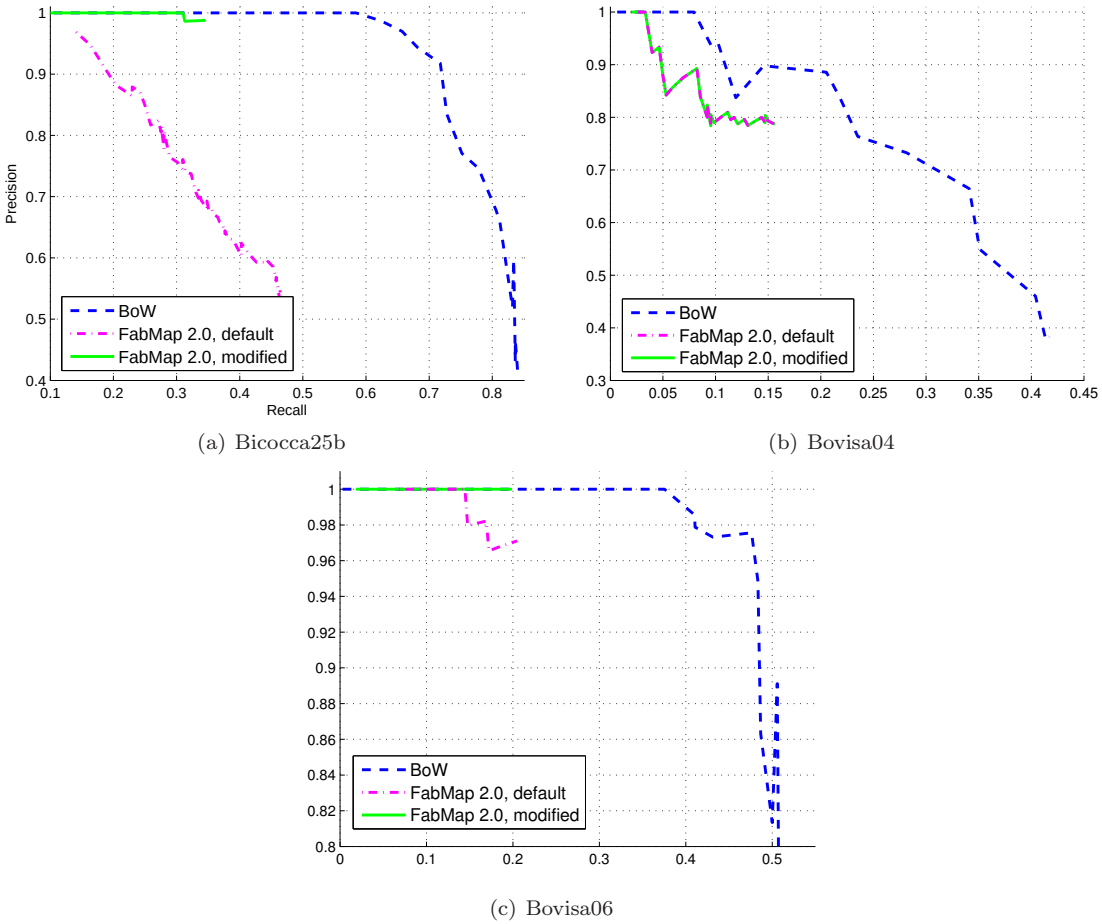
The results of FAB-MAP 2.0 over the datasets are shown in Figures 3.20(a), 3.21(a) and 3.22(a) for the default set of parameters plus the geometrical checking with the epipolar constraint, and in Figures 3.20(b), 3.21(b) and 3.22(b) for the modified set of parameters. Please, note that FAB-MAP 2.0 with the modified configuration does not need geometrical verification, since we selected the parameters aiming to obtain no false positives. Again, as expected with

		Outdoor	Indoor	Mixed
	default	modified		
p	0.99	0.96	0.5	0.3
P(obs|exist)	039	0.39	0.31	0.37
P(obs|¬exist)	0.05	0.05	0.05	0.05
P(newplace)	0.9	0.9	0.9	0.9
σ	0.99	0.99	1.0	1.0
Motion Model	0.8	0.8	0.8	0.6
Blob Resp. Filter	25	25	25	25
Dis. Local	20s	20s	20s	20s

Table 3.3: Parameters of FAB-MAP 2.0 for the experiments

	Indoor	Outdoor	Mixed
Disallow local (τ_c)	20s	20s	20s
Temporal constraint length (τ_l)	4s	4s	4s
Distance between consecutive timestamps (τ_d)	2s	2s	2s
Min. $s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$	0.1	0.1	0.1
Candidate min. threshold (α)	0.15	0.15	0.15
Candidate max. threshold (α^+)	0.6	0.6	0.6
CRF 3D Likelihood threshold (α_{3D})	1	1.5	1.5
CRF I Likelihood threshold (α_I)	1.3	1.7	1.7

Table 3.4: Parameters of BoW + CRF-Matching for the experiments



(a) Bicocca25b

(b) Bovisa04

(c) Bovisa06

Figure 3.18: Precision and recall without geometrical check of BoW and FAB-MAP 2.0 with two parameter sets.

the modified parameters, FAB-MAP 2.0 obtained greater recall at full precision than with the parameters by default, although, some loop closures were not detected. We detail some cases: in the indoor Bicocca25b dataset, Figure 3.20(a), the big area on the beginning of the map (start-end), especially important in the experiment because if no loop is detected in that area, a SLAM algorithm could hardly build a correct map after having traversed such a long path (around 300 metres). Outdoors, in Bovisa04, as shown in Figures 3.21(a) and 3.21(b), the biggest loop was missed in the starting and final point of the experiment, in the marked area (O1) in the map. An example of a false negative in this area is shown in Figure 3.24(a). This dataset is challenging due to the illumination and blur present in the images. And this entails an added difficulty for FAB-MAP since the significant overlap of distant objects between consecutive images decreases its discriminative ability, as we showed in previous section. For the experiment in the dynamic mixed environment, Bovisa06, important loop closures were missed again, e.g. M1 and M2 areas in Figure 3.22(a). Examples of those false negative cases are shown in Figures 3.24(b) and 3.24(c). In the false negative cases that we show in Figure 3.24, both configurations of FAB-MAP 2.0 reported a probability of *new place* greater than 0.999.

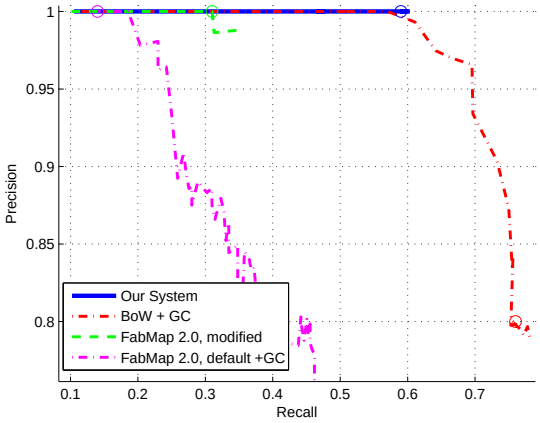


Figure 3.19: Precision-recall curve of our BoW + CRF-Matching system for the Bicocca25b dataset. We also show FAB-MAP 2.0 and our BoW stage with epipolar constraint (GC) as verification. Note that FAB-MAP 2.0 with the modified configuration needs no GC. Working points are marked in each curve, according to Table 3.3 and Table 3.4.

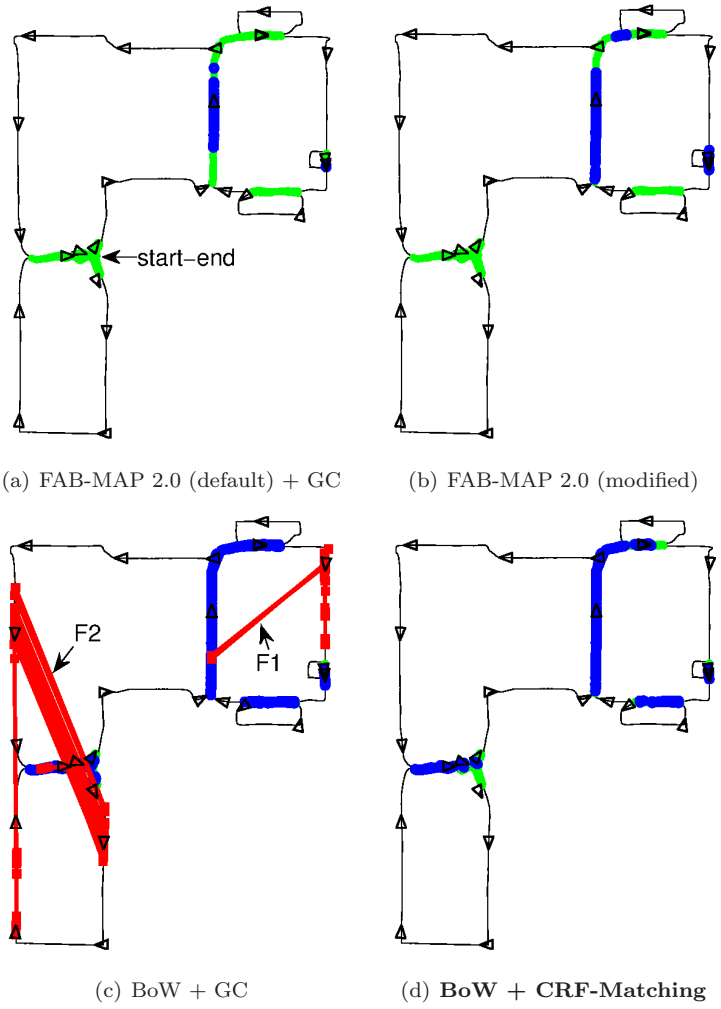
In order to show the improvements of the CRF-Matching verification stage, we checked the candidates given by BoW with the same GC technique we described in Section 3.3.1. The results are shown in Figures 3.20(c), 3.21(c) and 3.22(c). In Figure 3.20(c) all the loop closure areas were detected but with too many false positives due to the perceptual aliasing (see Figure 3.23); this is disastrous for any SLAM algorithm. As we previously showed in Figure 3.15, we can eliminate the false positive cases by imposing a more demanding value $\alpha = 0.5$ and applying the epipolar constraint to more than one camera. In the outdoor and mixed datasets the precision was 100%, sacrificing recall and, more important, the detection of loop closure areas. As we can see in Figure 3.19 we can tune the parameters of BoW+GC to attain full precision, but at the cost of sacrificing recall. This also makes the performance of this system not good and stable across environments and conditions.

The results of the BoW+CRF-Matching system over the datasets are shown in Figures 3.20(d), 3.21(d) and 3.22(d), and the comparative statistics of all experiments is made in Table 3.5. In the indoor experiment we can detect all the loop closure areas at 100% precision. In the outdoor and mixed datasets we keep full precision, higher recall level and most of the loop closure areas detected.

Our system detected successfully the loops of Figure 3.24 as true positives. The three cases shown were verified by the CRF stage. Our CRF matcher reports the follow η scores: in O1, Figure 3.24(a), $\eta_{3\mathrm{D}} = 1.24$ and $\eta_I = 1.37$, in M1, Figure 3.24(b), $\eta_{3\mathrm{D}} = 0.4$ and $\eta_I = 1.67$, and in M2, Figure 3.24(c), $\eta_{3\mathrm{D}} = 1.29$ and $\eta_I = 1.24$. Note that with the corresponding $\alpha_{3\mathrm{D}}$ and α_I parameters for indoors, such cases would be rejected.

Furthermore, our CRF matcher is robust against perceptual aliasing. For instance, the false positives obtained with the geometrical checking in the indoor sequence, see Figure 3.23, was correctly discarded. In the case of F1, Figure 3.23(a), our both graphs of the CRF matcher, $G_{3\mathrm{D}}$ and G_I, rejected it with $\eta_{3\mathrm{D}} = 1.45$ and $\eta_I = 1.79$. And in F2, Figure 3.23(b), our CRF matcher rejected it by the far information coded in G_I, with $\eta_{3\mathrm{D}} = 0.95$ and $\eta_I = 1.47$.

With the parameters by default of FAB-MAP 2.0 we cannot obtain full precision in the indoor dataset, even with $p = 0.99$. As explained in Figure 3.23(c), we have to verify the loop closures detected with the GC to attain full precision, obtaining lowest recall in the three datasets. With



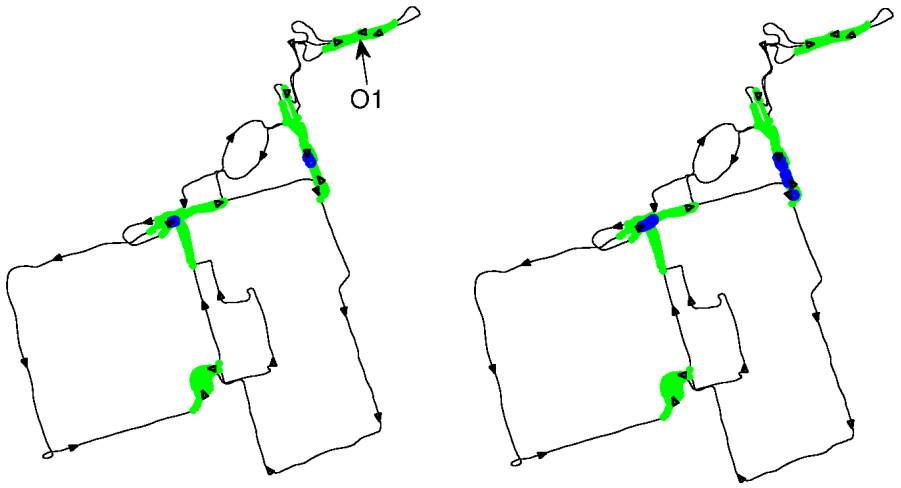
(a) FAB-MAP 2.0 (default) + GC

(b) FAB-MAP 2.0 (modified)

(c) BoW + GC

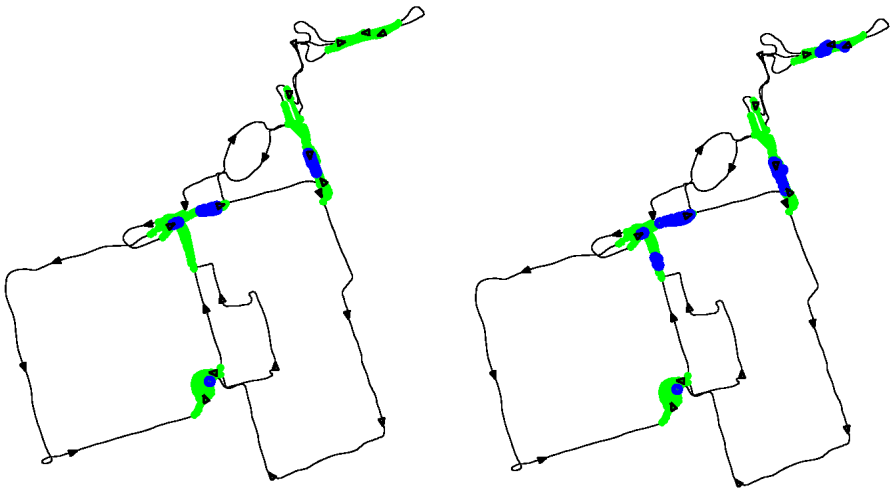
(d) **BoW + CRF-Matching**

Figure 3.20: Loops detected by each of the methods in the Bicocca25b dataset. Black lines and triangles denote the trajectory of the robot; light green lines, actual loops, deep blue lines denote true loops detected, and light red lines denote false loops detected. In Figure 3.23 we show the false positive cases F1 and F2.



(a) FAB-MAP 2.0 (default) + GC

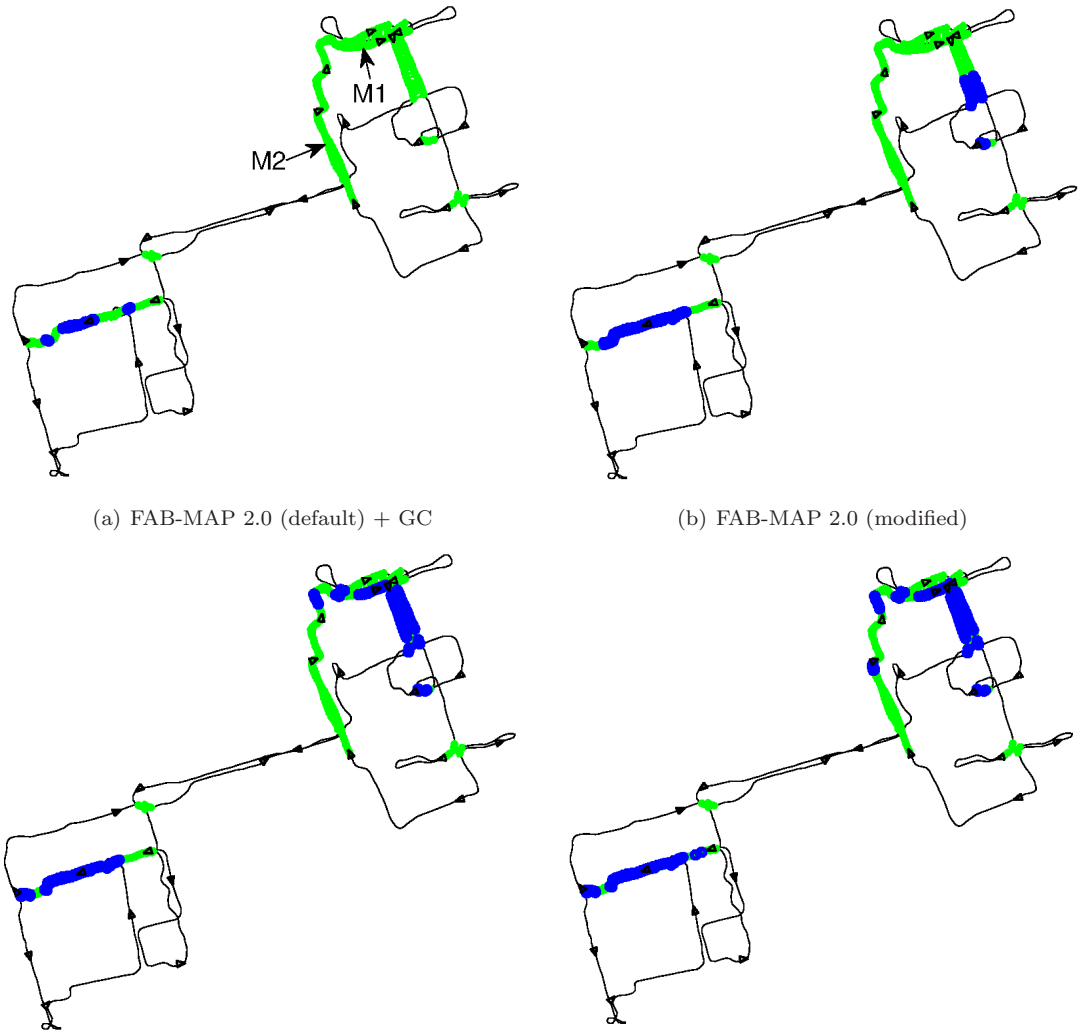
(b) FAB-MAP 2.0 (modified)



(c) BoW + GC

(d) **BoW + CRF-Matching**

Figure 3.21: Loops detected by each of the methods in the Bovisa04 dataset. Black lines and triangles denote the trajectory of the robot; light green lines, actual loops, deep blue lines denote true loops detected. In Figure 3.24 we show the false negative case O1.



(a) FAB-MAP 2.0 (default) + GC

(b) FAB-MAP 2.0 (modified)

(c) BoW + GC

(d) **BoW + CRF-Matching**

Figure 3.22: Loops detected by each of the methods in the Bovisa06 dataset. Black lines and triangles denote the trajectory of the robot; light green lines, actual loops, deep blue lines denote true loops detected. In Figure 3.24 we show the false negative cases M1 and M2.

(a) F1

(b) F2

(c) False positive of FAB-MAP 2.0 (default) rejected by the GC.

Figure 3.23: False positive cases obtained by BoW plus GC in the Bicocca25b dataset in (a) and (b). In (c), two different corridors make FAB-MAP 2.0 produce a false positive ($p = 0.9989$) with the default parameters. However, it is correctly rejected by the geometrical check.

(a) O1

(b) M1

(c) M2

Figure 3.24: False negatives in the Bovisa04 and Bovisa06 datasets that our method can successfully detect but FAB-MAP 2.0 misses. FAB-MAP 2.0 sets query image of case (a) as new place with a probability of 0.99947; of (b) with 0.99997 and 0.99902 with the default and modified set of parameters respectively, and of (c) with 1.0 and 0.9993. These scenes correspond to the biggest loops in the trajectories.

	Precision	Recall	Loop zones found/actual
Bicocca25b			
FAB-MAP 2.0 def. + GC	100%	14.1%	2/6
FAB-MAP 2.0 mod.	100%	30.6%	2/6
BoW + GC	79.8%	76.3%	6/6
BoW + CRF-Matching	100%	59.1%	6/6
Bovisa04			
FAB-MAP 2.0 def. + GC	100%	0.7%	2/9
FAB-MAP 2.0 mod.	100%	3.3%	2/9
BoW + GC	100%	7.0%	3/9
BoW + CRF-Matching	100%	11.15%	6/9
Bovisa06			
FAB-MAP 2.0 def. + GC	100%	3.7%	1/8
FAB-MAP 2.0 mod.	100%	19.9%	3/8
BoW + GC	100%	29.9%	4/8
BoW + CRF-Matching	100%	32.8%	5/8

Table 3.5: Results for Rawseeds datasets

the modified configuration, we tuned the parameters of FAB-MAP 2.0 aiming to maximize the precision. With this approach we obtained 100% precision in the Bicocca25b, Bovisa04 and Bovisa06 datasets with 30.6% recall and 2/6 loop areas detected, 3.3% recall and 3/9 areas, and 19.9% recall and 3/8 areas, respectively.

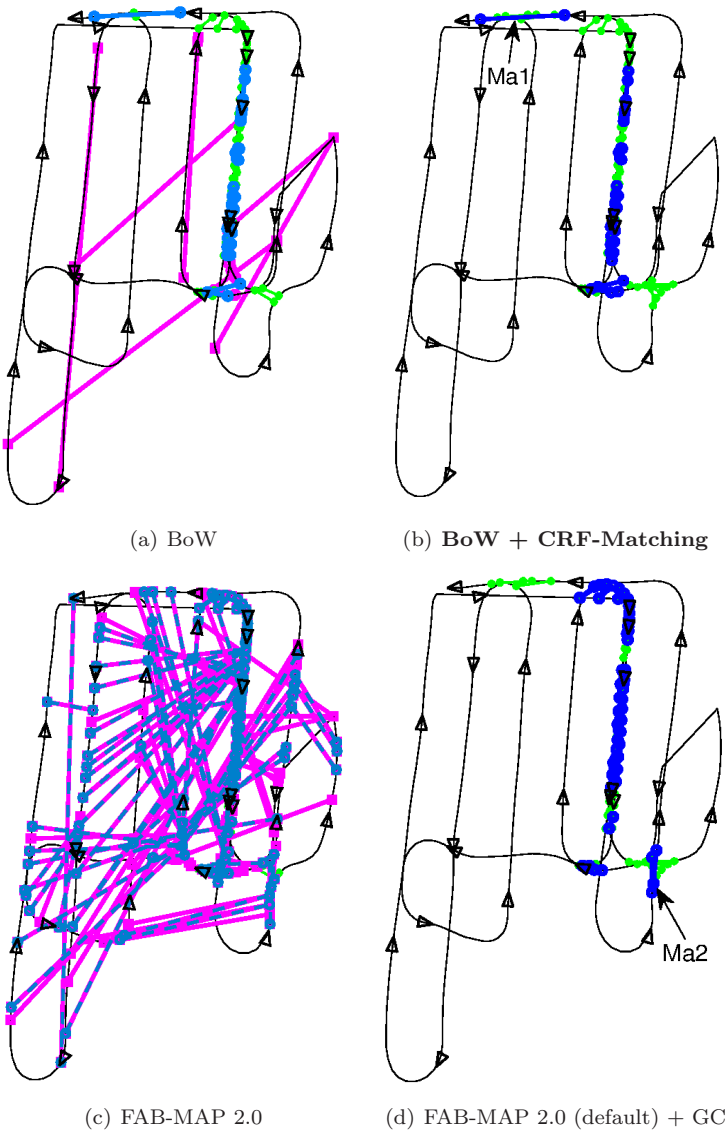
We also tried to tune the parameters of FAB-MAP 2.0 to maximize recall without paying attention to the precision, which can be improved later by using the geometrical constraint. With that approach, we could attain 100% precision in the Bovisa04 and Bovisa06 datasets, but false positives remained indoors, obtaining 75% precision only. As in the case of BoW+GC shown in Table 3.5, the geometrical check was not able to filter out all the incorrect loop candidates suffering from perceptual aliasing. In the mixed dataset, Bovisa06, the recall obtained, 15%, was lower than that observed with the other FAB-MAP 2.0 configuration. The same situation occurred outdoors, in Bovisa04, except for unrealistically low thresholds, like $p = 0.3$, that yielded a recall up to 5%.

No hand-tuning test

After obtaining the best results in the different datasets of the Rawseeds Project comparatively, we tested our BoW+CRF-Matching system over a different dataset, the Malaga parking lot 6 (Blanco et al. 2009). As in the previous case, we carry out the place recognition task at 1fps. This dataset, as the indoor one above, was collected in a completely different location from the

	Precision	Recall	Loop zones found/actual
FAB-MAP 2.0 def. + GC	100%	67.9%	4/5
FAB-MAP 2.0 mod.	41.5%	81.2%	4/5
BoW + CRF-Matching	100%	41.8%	4/5

Table 3.6: Results for the Malaga6L dataset



(a) BoW (b) **BoW + CRF-Matching**

(c) FAB-MAP 2.0 (d) FAB-MAP 2.0 (default) + GC

Figure 3.25: Loops detected by our system and FAB-MAP 2.0 in the Malaga6L dataset. Black lines and triangles denote the trajectory of the robot; light green lines, actual loops. In (a) high confidence detections (light blue) are accepted and unclear detections (magenta) are subject to verification. In (c) detections with $p = 0.99$ (default) are in dashed light blue, detections with $p = 0.96$ (mod.) are magenta. Detections with $p = 0.99$ which are verified with GC are shown in (d). In (b) and (d), deep blue lines denote true loops detected.

one were our training images were acquired. The main challenge is to test our system with the configuration already used in the previous experiments on a different vehicle and stereo camera system. For that, we kept the same vocabulary and CRFs' weights, as well as the parameters used in the outdoor Bovisa04 dataset, as shown in Table 3.4. In order to compare the results, we ran FAB-MAP 2.0 with the configuration that obtained the best result for the outdoor experiment, and also with the default parameter set, $p = 0.99$, and filtering the results with the epipolar constraint GC as above.

The results over the Malaga6L dataset are shown in Figure 3.25 and in Table 3.6. With no changes in our system we attain full precision despite the increased speed of this vehicle. FAB-MAP 2.0 with the configuration for best performance in the outdoor experiment obtains higher recall here, but precision falls down to 42%, unacceptable for any SLAM system. The configuration that exhibited bad performance in recall in the outdoor experiment, FAB-MAP 2.0 def. plus GC, attains higher recall compared to our system (68% vs. 42%), but both methods find 4 out of 5 loop closure zones. We show in Figure 3.26 two examples of those loops found by one and not by the other, Ma1 and Ma2.

Note in Figure 3.25(c) that FAB-MAP 2.0 alone has bad performance. It returns a large number of detections, more than half false loops. The increase in the number of alarms as compared with the Rawseeds experiments is due to the higher speed of the vehicle, 2.8m/s vs. 0.8m/s (see Table 3.1). This results in less overlap between consecutive processed frames, increasing the maximum values of the probability distribution over the sequence. Still, it is susceptible to perceptual aliasing due to overlapping in the far information. This is corrected with GC because this dataset does not suffer from strong perceptual aliasing in near information, in contrast with the indoor dataset, Figure 3.23.

Our loop detection stage discriminates better, still detecting the most of loop closure zones, see Figure 3.25(a). As expected, our verification stage correctly decides over the unclear cases (magenta lines in Figure 3.25(a)). The final result of our full system is shown in Figure 3.25(b).

3.4.4.3 Execution time

Our online BoW+CRF-Matching system runs at 1fps. We have a research implementation in C++ using the OpenCV 2 library. For the BoW stage, we rewrote the implementation we used in Section 3.4.3 to optimize some of its functions (cf. Section 1.4.3).

In Table 3.7 we show the average and maximum times for each stage of the system on a 2.3 GHz Intel Core i3 CPU M350 and 4GB of RAM. For the whole system, the average and the maximum times were computed only when all the stages were executed. Note that the maximums for each stage happened in different cases. That is more evident in the inference process for $G_{3\mathrm{D}}$ and G_I: when an image provides more 3D points, less background information remains. In an image, the number of nodes between $G_{3\mathrm{D}}$ and G_I are complementary. The execution time reported by BoW includes converting SURF features into words, accessing and updating the inverted index and performing the loop detection, and the execution time reported by the CRF Matcher, computing the minimum spanning trees, the corresponding descriptors and the inference for each graph. The time for the whole system includes computing the 3D point cloud from the disparity map and writing and reading the SURF descriptors and point clouds on disk.

(a) Ma1

(b) Ma2

Figure 3.26: Cases marked in Figure 3.25(b) and Figure 3.25(d) from Malaga6L dataset. Case (a) Ma1 is a loop found by our BoW + CRF-Matching approach. Case (b) Ma2 is a loop found by FAB-MAP 2.0 (default) + GC. Places are correctly recognized although the position of the cameras are some meters apart. Both cases are considered as true positives.

	SURF extraction	BoW	CRF Matcher		Whole system
			G_{3D}	G_I	
Average	0.15	0.01	0.15	0.15	0.47
Maximum	0.30	0.04	0.36	0.65	1.04

Table 3.7: Computational times for our BoW + CRF-Matching system (in s)

3.5 Discussion

In this chapter we presented a novel technique based on bag-of-words visual vocabularies to detect loop closures in sequences of images collected by mobile platforms with one or multiple cameras. We explicitly exploited the advantages of the sequentiality of the video stream to reinforce the reliability of the detections. Due to false positive cases in which appearance-based matches can result, we also proposed different approaches to check for geometrical verification, making the best of the hardware resources available in each platform. The important lesson that we can learn from this is that we must always apply a verification stage over detected loops based on appearance.

We evaluated our place recognition system in different environments (indoor, outdoor and mixed, and static and dynamic) from public datasets. In all cases our system could attain 100% precision (no false positives), detecting the most and especially important loop closure zones. Our results are at least on a par with the state of the art. In our experiments, our technique outperformed FAB-MAP, even with vocabularies of similar origin and size and without further geometrical verification. We conjecture that this is because the effectiveness of FAB-MAP must decrease when the camera looks forward, because FAB-MAP models the environment as "a collection of discrete and disjoint locations" (Cummins & Newman 2008). However, in our experiments the cameras face forward, and distant objects (e.g., buildings in outdoor scenes) persist for many frames, making scenes overlap and be less discriminative. This causes the matching probability mass of FAB-MAP to be flattened over the scenes. It is easier for our system to overcome those cases because our normalized similarity scores (η_{L_1}, η_{3D}, η_I) for matching acceptance are computed at each frame and take into account the similarity between consecutive frames.

Our place recognition system is able to run in real time, processing scenes at one frame per second. In most cases, after extracting the SURF features (max. 300ms), our system only takes 11ms to detect if there are possible loop closures, and 300ms to verify them when necessary. In the following chapter we consider the use of cheaper feature extractors that can speed up this process without a negative impact in precision and recall.

In our experiments, the best α thresholds for matching acceptance turned out to be different for indoor and for outdoors scenarios. These parameters will also depend on the velocity of motion, mainly because we use images from the previous second as reference in the comparisons. Nevertheless, our system has demonstrated a stable performance, always at full precision, for different environments, cameras and conditions. Systems such as a simple bag-of-words approach or FAB-MAP, both aided by an epipolar constraint, can obtain good results if adequately tuned in each case. However, the same configuration can result in very poor performance in others. In the following chapter, we work on the configuration aspects of our system to obtain more robust results in a wider set of heterogeneous evaluation datasets.

An important line of future work is addressing the place recognition problem over time. Our system performs well in multi-day sessions using parameters learned in different months, and this is also true of alternative systems such as FAB-MAP. The environment can also change during the operation in the same session (see Figure 3.24). Our algorithm is also able to detect places

revisited at different times of day, while alternative systems sometimes reject them in order to maintain high precision. Several extensions are possible for operation in longer periods of time. The vocabulary for the BoW has shown to be useful in different environments, which suggests that a rich vocabulary does not require frequent updates. The learned parameters in the CRF stage can be re-learned in sliding window mode depending on the duration of the mission. The system will then be able to adjust to changing conditions. In cases of periodical changes, such as times of day or seasons, we will need to maintain several environment models and selecting the most appropriate for a given moment of operation.

Chapter 4

Real-time loop detection

4.1 Real-time systems

In the previous chapter we presented our system to detect loops in sequences of images acquired with single or multiple cameras. As shown, our approach can work at 1–2Hz on a usual desktop computer when performing a geometrical verification based on conditional random fields with stereo data. Although this performance suffices in many robotic applications, it is not balanced with the video frame rate achieved by SLAM algorithms (Davison et al. 2007). This increases the complexity of real systems since it may cause SLAM algorithms to run in two decoupled threads, needing later synchronization, as in the CI-Graph SLAM case (cf. Section 3.3.2): one to perform the main SLAM functionality, and the other just to detect loop closures.

Real-time systems gain special relevance in mobile platforms moving at high speed. For example, Cummins & Newman (2011) address the loop detection problem with a camera on a car, in a trajectory 1000Km in length along England's roads. In the near future, robotic systems will be fully integrated with cars, and autonomous or semi-autonomous functionality will be a reality. Signs of this are the research lines set out by, for instance, the DARPA Grand Challenge competition (DARPA 2007) or Google's driverless car. When SLAM is performed by a car moving in an urban or intercity area, one second time may mean a difference of 10–30 meters in the position of the vehicle, which may cause the miss of valuable information for mapping. Compact and less resource-consuming systems can reduce the execution time for loop detection.

A lowering of resource requirements is also beneficial for low power devices, such as mobile phones, which are currently capable of running SLAM algorithms (Klein & Murray 2009) and are able to provide new applications for localization, augmented reality, etc.

In this chapter, we introduce novel enhancements to our loop detection algorithm to make it able to retrieve loop candidate images and to establish point correspondences between images at video rate, much faster than current approaches, with a conventional CPU and a single camera. We initially presented this work in Gálvez-López & Tardós (2011), and improved it later in Gálvez-López & Tardós (2012). The main speed improvement comes from the use of features from accelerated segment tests (FAST) (Rosten & Drummond 2006) combined with a slightly modified version of the binary robust independent elementary features (BRIEF) (Calonder, Lepetit, Strecha & Fua 2010), whose descriptors are binary vectors where each bit is the result of an intensity comparison between a given pair of pixels around a key point.

We introduce a bag of words that discretizes a binary space, and augment it with a direct index, in addition to the usual inverse index. To the best of our knowledge, this is the first time a binary vocabulary is used for loop detection. We also introduce a technique to prevent images

collected in the same place from competing among them when the database is queried. We achieve this by grouping together those images that depict the same place during the matching. We show a novel use of the direct index to efficiently obtain point correspondences between images, speeding up the geometrical check during the loop verification.

4.2 Bags of binary words

Extracting local features (keypoints and their descriptor vectors) is usually very expensive in terms of computation time when comparing images. For example, SURF requires 150–300ms per image to provide features. This is often the bottleneck when these kinds of techniques are applied to real-time scenarios. To overcome this problem, in this part of the work we use FAST key points (Rosten & Drummond 2006) and the state-of-the-art BRIEF descriptors (Calonder, Lepetit, Strecha & Fua 2010) to create a hierarchical vocabulary tree of binary words.

4.2.1 Features from accelerated segment tests

Features from accelerated segment tests (FAST) are points that satisfy a cornerness condition based on simple gray intensity comparisons. An image point $\mathbf{p}$ becomes a FAST corner if there exist n contiguous pixels in a Bresenham circle of radius 3 around $\mathbf{p}$ that are brighter than pixel $\mathbf{p}$ plus a fixed threshold α_f, or darker than its value minus the threshold. Figure 4.1 shows an example. Here, we accept a corner if the condition applies to $n = 9$ pixels (FAST-9 in the literature).

The computation of FAST key points is accelerated by selecting the pixel tests in order such that candidate points can be rejected after a very few comparisons when they are not corners. This is done according to a machine learning procedure that uses the ID3 algorithm (Quinlan 1986) on training data to produce a decision tree. Tree branches define at each step the test in the Bresenham circle that is more informative about the cornerness condition of the pixel

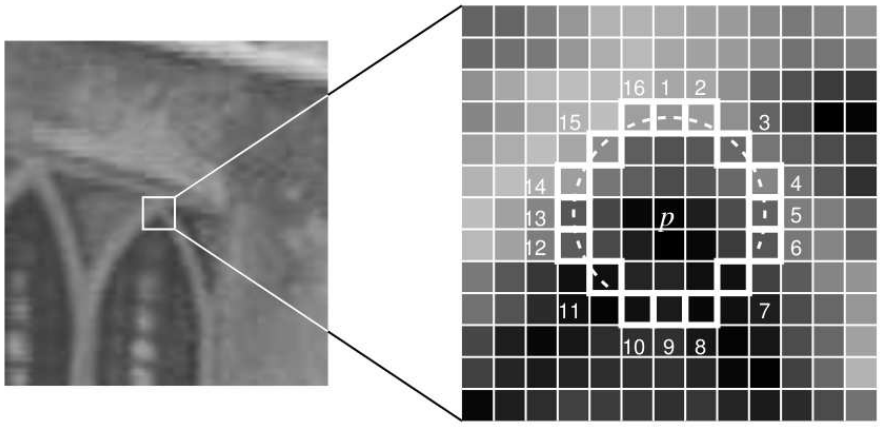


Figure 4.1: Candidate FAST corner $\mathbf{p}$. The highlighted squares are the pixels of the Bresenham circle used in the corner detection. The corner is accepted if the arc indicated by the dashed line passes through n contiguous pixels which are brighter than $I(\mathbf{p}) + \lambda_f$ or darker than $I(\mathbf{p}) - \lambda_f$. Credit: Rosten & Drummond (2006)

according to the training data. This training is done in an offline stage and a static decision tree is used later.

The cornerness of FAST key points is valued with a score defined as the maximum sum of the absolute difference between the set of brighter or darker pixels in the contiguous arc and the center pixel. Intuitively, this score conveys a sense of the persistence of the corner, and it is used to perform non-maxima suppression and ignore those corners that are adjacent to others expected to be more reliable.

Since only a few pixels need checking, obtaining FAST key points requires very little time. Rosten & Drummond (2006) reported 1.33ms in a 2.6GHz computer to extract 500 features, including the non-maxima suppression step, proving successful for real-time applications (Gauglitz et al. 2011).

4.2.2 Binary robust independent elementary features

We combine FAST key points with binary robust independent elementary features (BRIEF). For each FAST key point, we draw a square patch around them and compute a BRIEF descriptor. The BRIEF descriptor of an image patch is a binary vector where each bit is the result of an intensity comparison between two of the pixels of the patch. The patches are previously smoothed with a Gaussian kernel to reduce noise. Given beforehand the size of the patch, $S_b \times S_b$, the pairs of pixels to test are randomly selected in an offline stage. In addition to S_b, we must set the parameter L_b: the number of tests to perform (i.e., the length of the descriptor). For a point $\mathbf{p}$ in an image, its BRIEF descriptor vector $\mathbf{B}(\mathbf{p})$ is given by:

$$B^i(\mathbf{p}) = \begin{cases} 1 & \text{if } I(\mathbf{p}+\mathbf{a}_i) < I(\mathbf{p}+\mathbf{b}_i) \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in [1,\, L_b] \tag{4.1}$$

where $B^i(\mathbf{p})$ is the i-th bit of the descriptor, $I(\cdot)$ the intensity of the pixel in the smoothed image, and $\mathbf{a}_i$ and $\mathbf{b}_i$ the 2D offset of the i-th test point with respect to the center of the patch, with value in $\left[-\frac{S_b}{2} \dots \frac{S_b}{2}\right] \times \left[-\frac{S_b}{2} \dots \frac{S_b}{2}\right]$, randomly selected in advance. Note that this descriptor does not need training, just an offline stage to select random points that hardly takes time.

The original BRIEF descriptor proposed by Calonder, Lepetit, Strecha & Fua (2010) selects each coordinate of the test points $\mathbf{a}_i$ and $\mathbf{b}_i$ according to a normal distribution $\mathcal{N}(0, \frac{1}{25}S_b^2)$. However, we found that using short test pairs, i.e. offsets which are close in the image space, yielded better results (Section 4.4.2). We select each coordinate j of these pairs by sampling the distributions $a_i^j \sim \mathcal{N}(0, \frac{1}{25}S_b^2)$ and $b_i^j \sim \mathcal{N}(a_i^j, \frac{4}{625}S_b^2)$. Note that this approach was also proposed by Calonder, Lepetit, Strecha & Fua (2010), but not used in their final experiments. For the descriptor length and the patch size, we chose $L_b = 256$ and $S_b = 48$, because they resulted in a good compromise between distinctiveness and computation time.

The main advantage of BRIEF descriptors is that they are very fast to compute and to compare. For example, Calonder, Lepetit, Strecha & Fua (2010) reported 17.3μs per keypoint when $L_b = 256$ bits. Since one of these descriptors is just a vector of bits, measuring the distance between two vectors can be conveniently done with the Hamming distance: by counting the amount of different bits between them, which is implemented by counting the number of ones after an *xor* operation. This is more suitable in this case than calculating the Euclidean distance, as usually done with SIFT or SURF descriptors, composed of floating-point values. Note however that unlike SIFT or SURF, no key point orientation or scale is computed at any stage, so that these descriptors are not so robust before image transforms. As we show in our experiments, we can do without that invariance in many cases and still obtain a high recall at full precision regarding loop detection.

4.2.3 Hierarchical binary vocabulary tree

Unlike the SURF case, with BRIEF descriptors we discretize now a binary descriptor space, creating a more compact vocabulary. This vocabulary is structured again as a tree by performing hierarchical k-medians clustering with the k-means++ seeding. This algorithm computes the median value of each dimension of the descriptor vectors individually to obtain centroids, which in the binary space is equivalent to set the most populated value of each dimension. When there is the same number of one and zero values, the resulting mathematical median is not binary, so that an arbitrary value of 0 is set.

Unlike real spaces, the quantization of binary spaces is prone to produce several clusters which are equidistant to some points, as depicted in Figure 4.2. This is detrimental for the nearest neighbor problem, since near points in the space can be associated to different clusters, as Trzcinski et al. (2012) noticed. Nevertheless, our experimental evaluation shows that this performance loss is not relevant, even nonexistent in some cases, in the loop detection problem, when we compare the results with a SURF vocabulary.

Binary words are also given an *idf* weight during the creation of the vocabulary. When a binary descriptor is converted into a word, it traverses the tree by selecting the nodes that minimize the Hamming distance, that is to say, by selecting at each level the node with fewest dissimilar bits. After the conversion, the weight of words obtained from an image are complemented to obtain the *tf–idf* weight (cf. Section 3.1.3). The *tf-idf* values allow bag-of-words vectors to be compared with the L_1 distance again, yielding the normalized similarity score η_{L_1} when exploiting video sequentiality during the loop candidate retrieval.

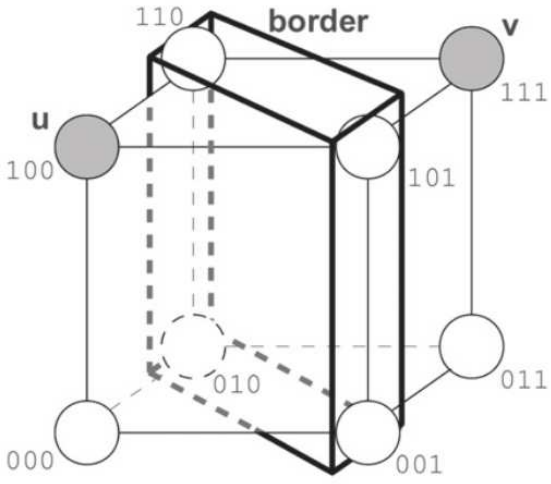


Figure 4.2: Thick boundary problem in the binary space. Unlike Euclidean spaces, the borders between two clusters are not smoothly defined in a Hamming space. In this case, given clusters centered in **u** and **v**, all the points in the rectangular area are equidistant to both of them, so they may belong to any. Credit: Trzcinski et al. (2012).

4.3 Video-rate loop retrieval and verification

We improve our loop detection approach with the compact features that result by combining FAST key points with BRIEF descriptors. The basic aspects of the algorithm are those presented in Chapter 3, however, we define now a visual vocabulary with binary words, clustering a binary space, and introduce the notion of matching image grouping to avoid images acquired in the same place from clashing when the database is queried. Furthermore, we introduce a direct index to efficiently compute feature correspondences. All these additions allow us to increase the working frequency of our loop detector.

4.3.1 Temporal consistency for groups of matches

The appearance model of an image is again represented with a bag-of-words vector $\mathbf{v}$, and the resemblance between images in a sequence is measured with the normalized similarity score η_{L_1}.

One of the difficulties that arise when querying the image database to look for candidate images is that there is not a single peak in the set of matching η_{L_1} scores, but several high-scored matches. These are mainly due to consecutive images acquired in the same place when the robot moves slowly, or when far-off objects are present for a long time. Selecting the matching image that maximizes η_{L_1} in those cases can cause the temporal constraint to fail.

Consider the example given in Figure 4.3, supposing the acquisition of images at every $\Delta t = 1$ second. Query images at timestamps t and $t-1$ are shown in the column on the left, and the matched images retrieved from the database, which are close in time, on the right. Matched images are from left to right in descending order of η_{L_1} score. In this example, all the matched images are valid since they depict the same place as the query images, but obtain different scores

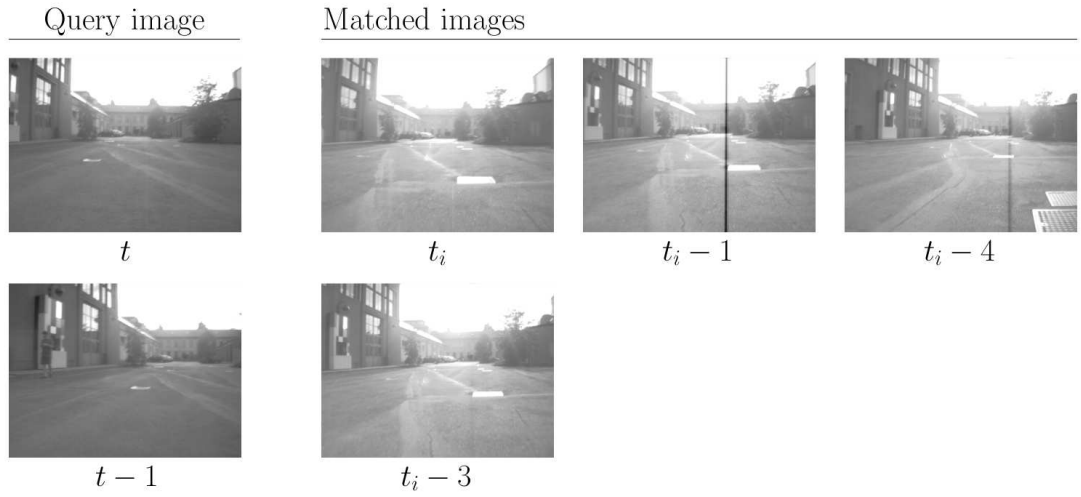


Figure 4.3: Example of false rejection of the temporal constraint. Images are acquired at every $\Delta t = 1$ second, and matches are from left to right in descending order of η_{L_1} score. When matches that maximize the η_{L_1} score for each query are selected, i.e. $\langle t, t_i \rangle$ and $\langle t-1, t_i - 3 \rangle$, the temporal consistency condition fails, since $t_i - (t_i - 3) > \tau_d = 2$s. Nevertheless, if the match $\langle t, t_i - 1 \rangle$ is selected instead, the temporal consistency condition holds and the loop candidate is not falsely rejected.

because of slight variations in the image due to camera pose, illumination artifacts or other factors. According to our original algorithm, if we pick out just the match that obtains the highest η_{L_1} score, we will consider matches $\langle t, t_i \rangle$ and $\langle t-1, t_i-3 \rangle$. However, the difference between the selected matches is not within the temporal constraint threshold, i.e. $t_i - (t_i - 3) > \tau_d = 2\text{s}$, and the temporal consistency test fails. On the other hand, if the match $\langle t, t_i - 1 \rangle$ is selected instead, which is valid since images at t_i and $t_i - 1$ depict the same scene, the temporal consistency condition holds and the loop candidate is not falsely rejected. Note that this problem is aggravated when the time between consecutive acquired images Δt is lower. To solve this issue and to prevent images that are close in time to compete among them when the database is queried, we group them into *islands* and treat them as only one match.

We use the notation $\mathcal{T}_i$ to represent a set of timestamps $\{t_{n_i}, \ldots, t_{m_i}\}$, T_i for the time interval $[\min(t \in \mathcal{T}_i), \max(t \in \mathcal{T}_i)]$, and $\mathbf{V}_{\mathcal{T}_i}$ for an island that groups together the matches with entries $\mathbf{v}_{t_{n_i}}, \ldots, \mathbf{v}_{t_{m_i}}$. Therefore, several matches $\langle \mathbf{v}_t, \mathbf{v}_{t_{n_i}} \rangle, \ldots, \langle \mathbf{v}_t, \mathbf{v}_{t_{m_i}} \rangle$ are converted into a single match $\langle \mathbf{v}_t, \mathbf{V}_{\mathcal{T}_i} \rangle$ if the gaps between consecutive timestamps in $t_{n_i}, \ldots, t_{m_i}$ are small. The islands are also ranked according to a score H:

$$H(\mathbf{v}_t, \mathbf{V}_{\mathcal{T}_i}) = \sum_{j=n_i}^{m_i} \eta_{L_1}(\mathbf{v}_t, \mathbf{v}_{t_j}) \tag{4.2}$$

The island with the highest score is selected as matching group and continue to the temporal consistency step. Besides avoiding clashes between consecutive images, the islands can help establish correct matches. If I_t and $I_{t'}$ represent a real loop closure, I_t is very likely to be similar also to $I_{t' \pm \Delta t}$, $I_{t' \pm 2\Delta t}$, ..., producing long islands. Since we define H as the sum of scores η_{L_1}, the H score favors matches with long islands as well.

After obtaining the best matching island $\mathbf{V}_{\mathcal{T}'}$, we check it for temporal consistency with previous queries, extending the temporal constraint applied in Section 3.2.2 to support islands. The match $\langle \mathbf{v}_t, \mathbf{V}_{\mathcal{T}'} \rangle$ must be consistent with k previous matches $\langle \mathbf{v}_{t-\Delta t}, \mathbf{V}_{\mathcal{T}_1} \rangle, \ldots, \langle \mathbf{v}_{t-k\Delta t}, \mathbf{V}_{\mathcal{T}_k} \rangle$ such that intervals T', T_1, T_2, ..., T_k are close to overlap. In the example given above in Figure 4.3, the set $\mathcal{T}'$ is composed of timestamps $\{t_i, t_i - 1, t_i - 4\}$, and $\mathcal{T}_1$, of $\{t_i - 3\}$; since there is overlap between $T' = [t_i - 4, t_i]$ and $T_1 = [t_i - 3, t_i - 3]$, matches $\langle \mathbf{v}_t, \mathbf{V}_{\mathcal{T}'} \rangle$ and $\langle \mathbf{v}_{t-1}, \mathbf{V}_{\mathcal{T}_1} \rangle$ are consistent. If an island passes the temporal constraint, we keep only the match $\langle \mathbf{v}_t, \mathbf{v}_{t'} \rangle$, for the $t' \in \mathcal{T}'$ that maximizes the score η_{L_1}, and consider it a loop candidate, which has to be finally accepted by the geometrical verification stage.

4.3.2 Direct index

The visual vocabulary and the inverted index are often the only structures used in the bag-of-words approach for searching for images and performing loop detection. As a novelty in this general approach, we also make use of a direct index to conveniently store the features of each image. Complementary to the inverted index, which maps words with the images in which they are present, a direct index stores for each image the words it contains.

The main functionality of a direct index is to retrieve the content of a database entry, but we use it for other purpose. We separate the nodes of the vocabulary tree according to their level l of clustering coarseness, starting at leaves, with level $l = 0$, and finishing in the root, $l = L_w$. For each image I_t, we store in the direct index the nodes at a certain level l that are ancestors of the words present in I_t, as well as the list of local features f_{tj} associated to each node. Thus, instead of words, we fill the direct index with pairs of any node of the vocabulary tree and lists of local image features. This is illustrated by Figure 4.4. We denote $\mathcal{D}_l$ the direct index created for nodes at level l, and $\mathcal{D}_l(t) = \{\langle i_1, \mathcal{F}_{t,i_1} \rangle, \langle i_2, \mathcal{F}_{t,i_2} \rangle, \ldots\}$ the direct index entry for image I_t,

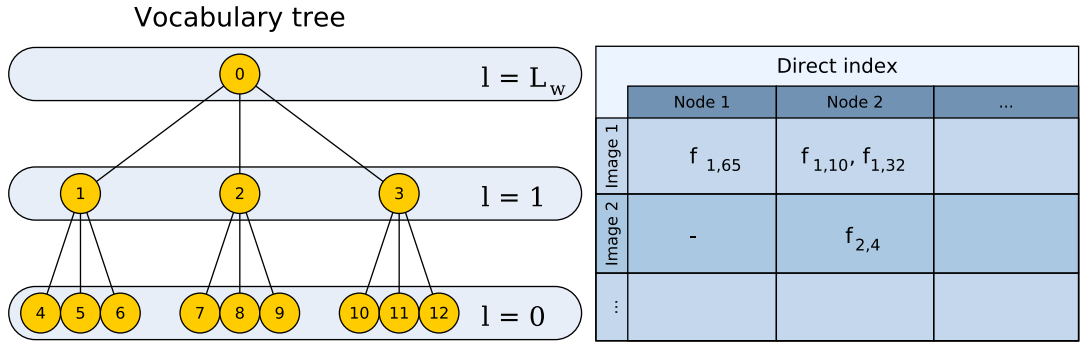


Figure 4.4: Example of vocabulary tree and direct index for level $l = 1$. The direct index stores the features of the images in the database and their associated nodes at coarseness level l in the vocabulary tree.

which is an ordered set of pairs of node indexes $(i_1 < i_2 < \dots)$ and sets of local features of I_t associated to those nodes $(\mathcal{F}_{t,i_1}, \mathcal{F}_{t,i_2}, \dots)$.

We take advantage of the direct index and the bag-of-words tree to use them as a means to approximate nearest neighbors in the BRIEF descriptor space. This is useful during the geometrical verification stage, when corresponding points are needed. The direct index allows to speed up the geometrical verification by computing correspondences only between those features that belong to the same words, or to words with common ancestors at level l. The direct index is updated when a new image is added to the database, and accessed when a candidate matching is obtained and geometrical check is necessary.

4.3.3 Computation of corresponding points

All the geometrical verification algorithms proposed in the previous chapter need to compute correspondences between images. The simplest method needs at least corresponding points between the loop candidate images I_t and $I_{t'}$. To compute these correspondences, we must obtain all the pairwise distances between feature descriptors of the query image and those of the matched one. There are several approaches to perform this. The easiest and slowest one is the exhaustive search, that consists in measuring the distance of each feature of I_t to the features of $I_{t'}$ in the descriptor space, to select correspondences later according to the neighbor-ratio condition (Lowe 2004). This is a $\Theta(n^2)$ operation in the number of features per image, where Θ stands for exact asymptotically bound.

A second technique consists in calculating approximate nearest neighbors by arranging the descriptor vectors of images individually in randomized k-d trees (Silpa-Anan & Hartley 2008). A k-d tree (Friedman et al. 1977) is a binary tree that arranges k-dimensional data in a hierarchical manner by separating them by one of the dimensions at each tree level. In a randomized k-d tree the chosen dimension is a random one from those that present the greatest variance; in contrast, the original k-d tree algorithm selects the maximum always. In order to approximate neighbors to compute corresponding features, several randomized k-d trees are built with their descriptors, obtaining one leaf node per feature. Then, given an input descriptor, the trees are traversed from the root by selecting child nodes according to the corresponding dimension. This is done at the same time for all the trees, maintaining a priority queue of visited nodes to skip those with

Algorithm 2: Correspondence computation with direct index

Input : Timestamps of candidate loop images t, t',
Direct index $\mathcal{D}_l$ for tree level l, s.t. $\mathcal{D}_l(t) = \{\langle i_1, \mathcal{F}_{t,i_1}\rangle, \langle i_2, \mathcal{F}_{t,i_2}\rangle, \dots\}$
Output: Set $\mathcal{C}$ of corresponding features between images at t and t'

$\mathcal{C} \leftarrow \varnothing$
$i \leftarrow \text{firstNode}(\mathcal{D}_l(t))$
$j \leftarrow \text{firstNode}(\mathcal{D}_l(t'))$
while neither i nor j reach the end of $\mathcal{D}_l(t)$ nor $\mathcal{D}_l(t')$ **do**
  if $i = j$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \text{correspondences}(\mathcal{F}_{t,i}, \mathcal{F}_{t',j})$
    $i \leftarrow \text{nextNode}(\mathcal{D}_l(t), i)$
    $j \leftarrow \text{nextNode}(\mathcal{D}_l(t'), j)$
  else if $i < j$ **then**
    $i \leftarrow \text{nextNode}(\mathcal{D}_l(t), i)$
  else if $i > j$ **then**
    $j \leftarrow \text{nextNode}(\mathcal{D}_l(t'), j)$
  end
end
return $\mathcal{C}$

higher distance than nodes in other trees. When a leaf node is reached, the feature represented by this leaf is taken as the first best neighbor. The search is then refined by checking for nearer neighbors in other leaf nodes.

Following the k-d tree idea, we take advantage of our single bag-of-words vocabulary tree and reuse it to approximate nearest neighbors, for any image, by exploiting the role of the direct index. To obtain correspondences between I_t and $I_{t'}$, we look up them in the direct index and perform comparisons only between those features that are associated to the same nodes at level l in the vocabulary tree. This condition speeds up the correspondence computation because we expect features that are close in the descriptor space to follow a similar path along the vocabulary tree when they are clustered as words. Since corresponding features do not always result in the same word, we use the parameter l, which is fixed beforehand and entails a trade-off between the number of correspondences obtained between I_t and $I_{t'}$ and the time consumed for this purpose. When $l = 0$, only features belonging to the same word are compared, so that the highest speed-up is achieved, but fewer correspondences can be obtained. This makes the recall of the complete loop detection process decrease due to some correct loops being rejected because of the lack of corresponding points. On the other hand, when $l = L_w$, recall is not affected but execution time is not improved either. Since direct index entries are implemented as sets ordered by node indexes, traversing the common nodes of two entries is very fast. Algorithm 2 outlines this process to compute correspondences between features of images I_t and $I_{t'}$, where function `correspondences` accounts for equation (3.12).

4.4 Experimental evaluation

We conducted several experiments of our current approach, including a detailed analysis of the relative merits of the different parts in our algorithm. We present comparisons between the effectiveness of BRIEF and two versions of SURF features, the descriptor most used for loop

closing. We also analyze the performance of the temporal consistency test when increasing the working frequency, and the impact of the direct index to compute correspondences for a simple geometrical verification based on the monocular epipolar constraint. We then present the results achieved by our technique after evaluating it in seven public datasets with 0.7–4Km long trajectories, and compare it with our previous proposal and the FAB-MAP 2.0 software. We demonstrate that we can run the whole loop detection procedure, including the feature extraction, in 52ms in 26300 images (22ms on average), outperforming previous techniques by more than one order of magnitude.

4.4.1 Methodology

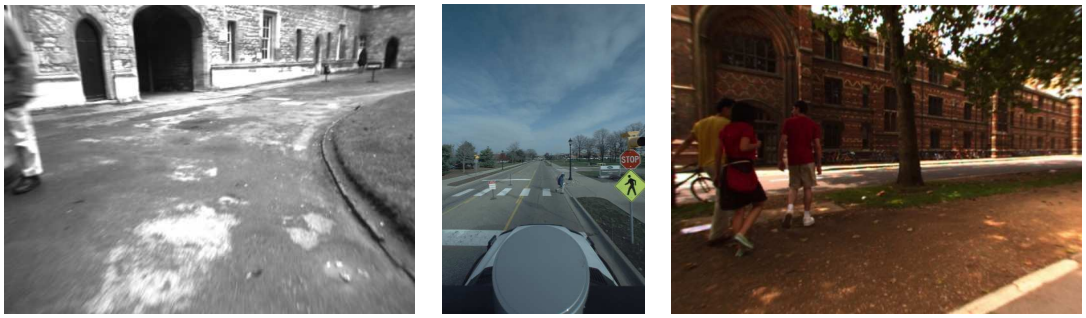
4.4.1.1 Datasets

Dataset	Description	C	T (m)	R (m)	S (m/s)	I (px × px)
Bicocca 2009-02-25b	Indoors, static	F	760	113	0.5	640 × 480
Bovisa 2008-10-04	Outdoor, static	F	1718	208	0.75	640 × 480
Bovisa 2008-10-06	Mixed, dynamic	F	1892	268	0.88	640 × 480
New College	Outdoors, dynamic	F	2260	1570	1.5	512 × 384
Ford Campus 2	Urban, ∼dynamic	F	4004	280	6.9	600 × 1600
Malaga 2009 Parking 6L	Outdoors, ∼dynamic	F	1192	162	2.8	1024 × 768
City Center	Urban, dynamic	L	2025	801	-	640 × 480

Table 4.1: Datasets used for evaluation. Legend: C, (F)rontal or (L)ateral camera; T, total dataset length; R, revisited length; S, average speed of the robot; I, image size.

We tested our enhanced loop detector in seven datasets, four of them already presented in the experimental evaluation section of the previous chapter. The three additional datasets used here are the New College campus (Smith et al. 2009), the Ford Campus 2 (Pandey et al. 2011) and the sequence City Center used by Cummins & Newman (2008). Table 4.1 summarizes them and Figure 4.5 shows some image examples.

The Bicocca25b, Bovisa04 and Bovisa06 datasets belong to the Rawseeds project (Rawseeds 2007-2009), and the Malaga6L (Blanco et al. 2009) dataset was collected with a camera on a



(a) NewCollege (b) Ford2 (c) CityCenter

Figure 4.5: Examples of datasets

car, in a parking area, as we detailed in Section 3.4.1. NewCollege (Smith et al. 2009) is a well-known challenging dataset, composed of stereo images captured at 20Hz along a 2.2Km path through a college's grounds and adjoining parks. It presents a long trajectory, with some people walking around and similar-looking places revisited several times whose appearance is likely to produce perceptual aliasing. The imagery of Ford2 (Pandey et al. 2011) was collected by a Point Grey Ladybug 3 omnidirectional camera system mounted on a car while driving around the Ford Research campus in Dearborn, Michigan during November-December 2009. We used the images from the frontal camera only. Its difficulty lies in the high speed of the platform and the restrictive horizontal field of view of the images, which produces views with mainly distant features. These six datasets provide images acquired at high frequency (between 7.5 and 20Hz). Unlike them, CityCenter (Cummins & Newman 2008) is a collection of images gathered along a 2Km path at low frequency and perpendicular to the motion of the robot. This makes consecutive images overlap very little, so they can be considered as a set of disjoint images instead of a sequence. This may be detrimental to the normalization of the score η_{L_1} of our method when the database is queried.

4.4.1.2 Training and evaluation datasets

The behavior of loop detection systems is sensitive to their parameters. It is common practice in the literature to tune system parameters according to the datasets where the system is evaluated. Nevertheless, to test our final loop detection system, we decided to use different data to choose the configuration of our algorithm and to evaluate it, in order to demonstrate the robustness of our approach.

We then separate the datasets shown in Table 4.1 into two groups. We use five of them (Bicocca25b, Bovisa04, Bovisa06, NewCollege and Ford2) as *training* datasets to find the best set of parameters of our algorithm. These present heterogeneous environments with many difficulties: lack of natural features indoors, perceptual aliasing, dynamic entities, distant scenes, narrow field of view, etc. The other two datasets (CityCentre and Malaga6L) are used as *evaluation* data to validate our final configuration. In these cases, we only use our algorithm as a black box with a predefined configuration.

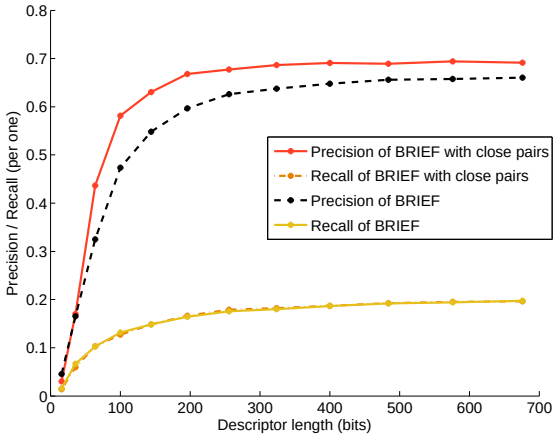
4.4.1.3 Settings

To use our algorithm we defined some settings that remained the same throughout all our experiments. A single vocabulary tree was used to process all the datasets. This was built with $k_w = 10$ branches and $L_w = 6$ depth levels, which yielded one million words. Its training was carried out with 9M features acquired from 10K images of another dataset from (Rawseeds 2007-2009) (*Bovisa 2008-09-01*). In all the experiments, we used a threshold of 10 units in the response function of FAST, and 500 in the Hessian response of SURF. For each processed image, we kept only the 300 features with highest response.

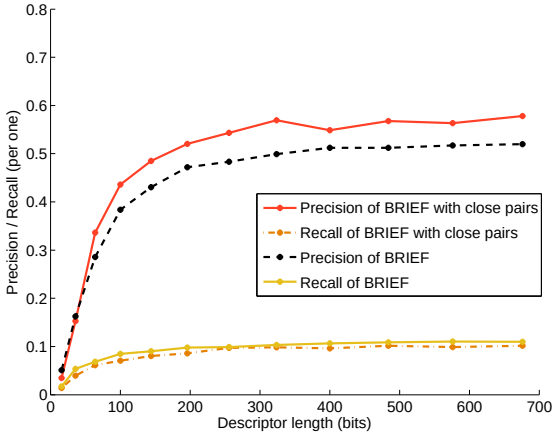
4.4.2 Selecting BRIEF parameters

We ran several tests to compare the performance of BRIEF by varying the descriptor length L_b, the patch size S_b and the method to select the test pairs $\mathbf{a}$ and $\mathbf{b}$ from equation (4.1). We tried two methods to choose the test pairs. The first one consisted in choosing both $\mathbf{a}$ and $\mathbf{b} \in \mathbb{R}^2$ from a normal distribution $\mathcal{N}_2(\mathbf{0}, \frac{1}{25}S_b^2)$, and the second one, in choosing close pairs such that $\mathbf{a} = \mathcal{N}_2(\mathbf{0}, \frac{1}{25}S_b^2)$, $\mathbf{b} = \mathcal{N}_2(\mathbf{a}, \frac{4}{625}S_b^2)$.

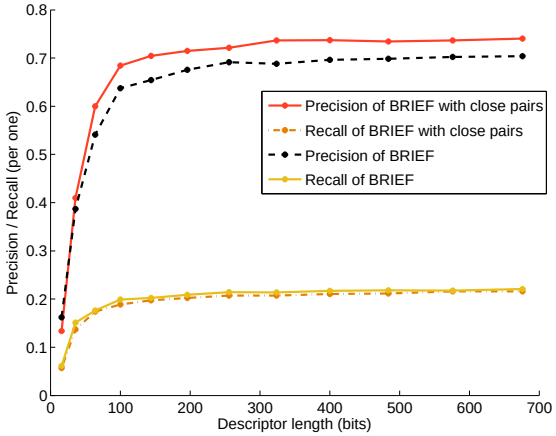
To do this test, we took several images (between 14 and 23) of three different scenes. By means of bundle adjustment (Triggs et al. 2000), we reconstructed the 3D geometry of the scenes



(a) Scene 1



(b) Scene 2



(c) Scene 3

Figure 4.6: Precision and recall in 3 test scenes of BRIEF and BRIEF with close pairs for several descriptor length L_b values, with patch size $S_b = 48$.

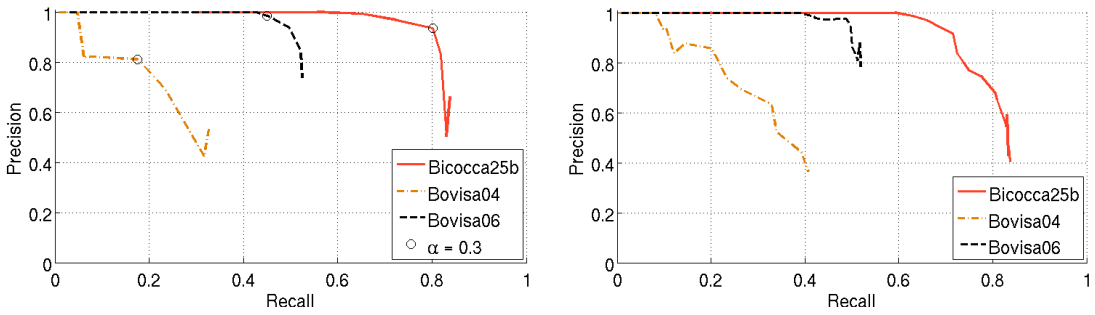
and obtained the ground-truth correspondences of every pair of images. We computed both versions of the BRIEF descriptor for patches of size S_b = 24, 48, 64 and 80 pixels around a dense set of FAST key points in each image, and matched them. A match was set if the distance between two descriptors was minimum, and the ratio between this and the distance to the second closest descriptor was lower than a threshold. Due to the random nature of the descriptor, we repeated this for 5 different test pair patterns. In Figure 4.6 we show the average precision and recall against the length of the descriptor when $S_b = 48$ of BRIEF and BRIEF with close pairs. The other patch sizes showed the same relation between both techniques, but $S_b = 48$ exhibited better results. We can see that the precision of BRIEF with close pairs is always higher than that of BRIEF with more general pairs for the same level of recall. This suggests the locality of the pairs provides more distinctiveness to the descriptor, and supports the idea behind new features such as the fast retina key points (FREAK) (Alahi et al. 2012), which build binary descriptors by comparing image intensities in areas which are smaller the closer to the key point center.

We also see that precision and recall improve when the number of pairs increases, up to some length. We finally chose BRIEF with close pairs, descriptor length $L_b = 256$ and patch size $S_b = 48$ pixels, for the rest of the experiments.

4.4.3 BRIEF and SURF effectiveness

SURF features are a common choice for loop detection (Cummins & Newman 2011, Paul & Newman 2010, Callmer et al. 2008). A BRIEF descriptor encodes much less information than a SURF descriptor, since BRIEF is not scale or rotation invariant. In order to check if BRIEF is reliable enough to perform loop detection, we compared its effectiveness with that of SURF.

We started by comparing our current loop detection system with that we presented and evaluated with SURF in the previous chapter, on the same training datasets: Bicocca25b, Bovisa04 and Bovisa06. For that, we processed their image sequences at f = 1Hz, deactivated the geometrical verification, fixed the required temporal consistency matches k to 3 and varied the value of the normalized similarity threshold α to obtain the precision-recall curves shown in Figure 4.7. On the left, we show the results of our system with FAST+BRIEF features, and on the right, the results presented in Section 3.4.4 that we obtained with SURF. We see that our system with FAST+BRIEF features performed very well in these three datasets, even without



(a) FAST + BRIEF (b) SURF

Figure 4.7: Precision-recall curves for several values of α and without geometrical checking. On the left, our current method using FAST+BRIEF, with the working point $\alpha = 0.3$ highlighted, and on the right, our previous algorithm using SURF on the same datasets. BRIEF features attain similar results than SURF, but require much less computational resources.

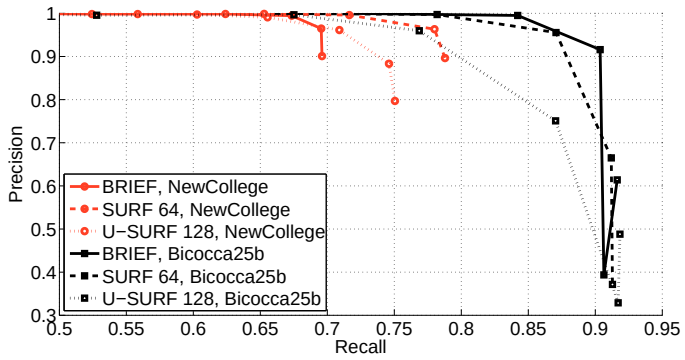


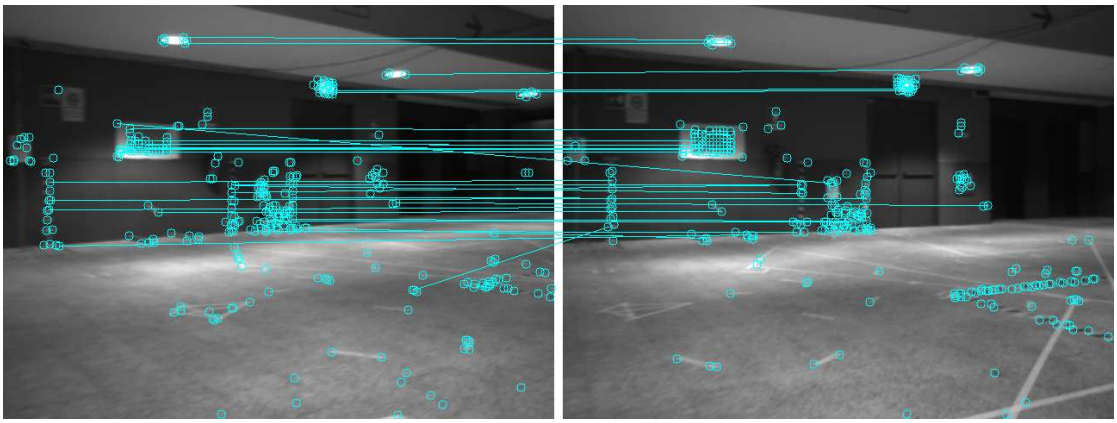
Figure 4.8: Precision-recall curves achieved by BRIEF, SURF64 and U-SURF128 in the training datasets, without geometrical verification.

geometrical verification. It obtained high recall, except for the outdoor dataset, without false positives. These results are very similar to those obtained with our previous system with SURF features. In the indoor Bicocca25b and mixed Bovisa06 datasets, BRIEF achieves a similar maximum recall for full precision than SURF, and its curve behaves like the SURF one. On the other hand, the curve of the outdoor Bovisa04 dataset of BRIEF is slightly below that of SURF. In both cases, the recall in this dataset is lower than that shown in the other two datasets. As we noticed before, this outdoor dataset is particularly challenging because the depth of the scenes produced very similar-looking images. In Section 3.4.3 we also showed that other techniques as FAB-MAP (Cummins & Newman 2008) did not reach 100% precision on this dataset.

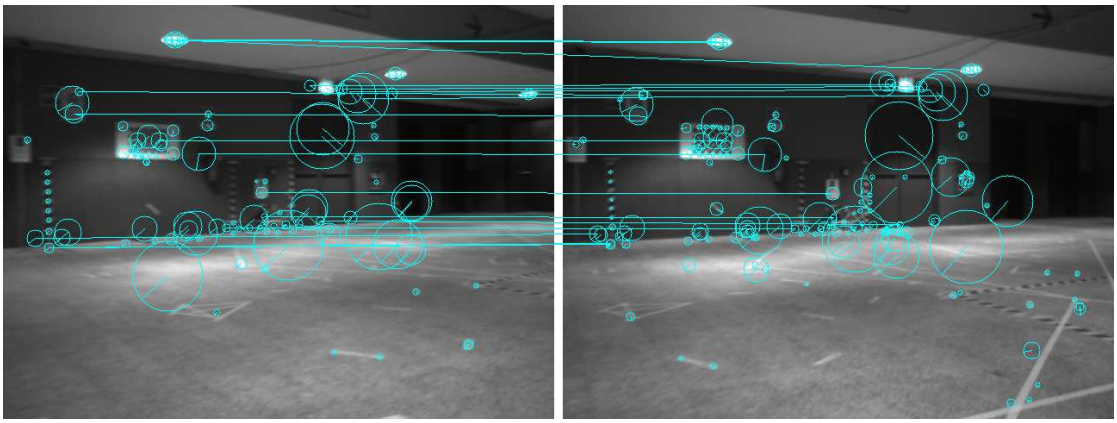
The previous comparison was done with vocabularies from different settings; the one used for BRIEF was built with branching factor $k_w = 10$ and $L_w = 6$ depth levels, yielding 1M words, whereas the one used for SURF contained 530K words. To look deeper into the differences between the two kinds of features, we performed a second test with a vocabulary with the same structure. Since SURF is rotation invariant and our training sequences hardly present in-plane rotation, it is possible that this invariance produces some perceptual aliasing that BRIEF can avoid. To study this factor, we selected two versions of SURF features for this test: 64-dimensional descriptors with rotation invariance (SURF64) and 128-dimensional descriptors without rotation invariance (U-SURF128), as those used by Cummins & Newman (2011).

We created vocabulary trees for SURF64 and U-SURF128 in the same way we built it for BRIEF, with 1M words, and ran our system on Bicocca25b and NewCollege, since these datasets do not present any overlap with the training images. We processed the image sequences at frequency $f = 2$Hz for several values of threshold α, we no geometrical verification and $k = 3$ again, and obtained the precision-recall curves shown in Figure 4.8. The first remark is that the curve of SURF64 dominates that of U-SURF128 on both datasets. We can also see that BRIEF offers again a very competent performance compared with SURF. In Bicocca25b, BRIEF outperforms U-SURF128 and is slightly better than SURF64. In NewCollege, SURF64 achieves better results than BRIEF, but BRIEF still gives very good precision and recall rates. On sight of these results, we can conclude the rotation invariance has little impact on the loop detection capability of the system.

To better illustrate the different abilities of BRIEF and SURF64 to find correspondences, we have selected some loop events from the previous experiments. In Figure 4.9, Figure 4.10 and Figure 4.11 the features that are associated to the same word of our vocabulary are connected with lines. These are the only matches taken into account to compute the normalized similarity



(a) BRIEF

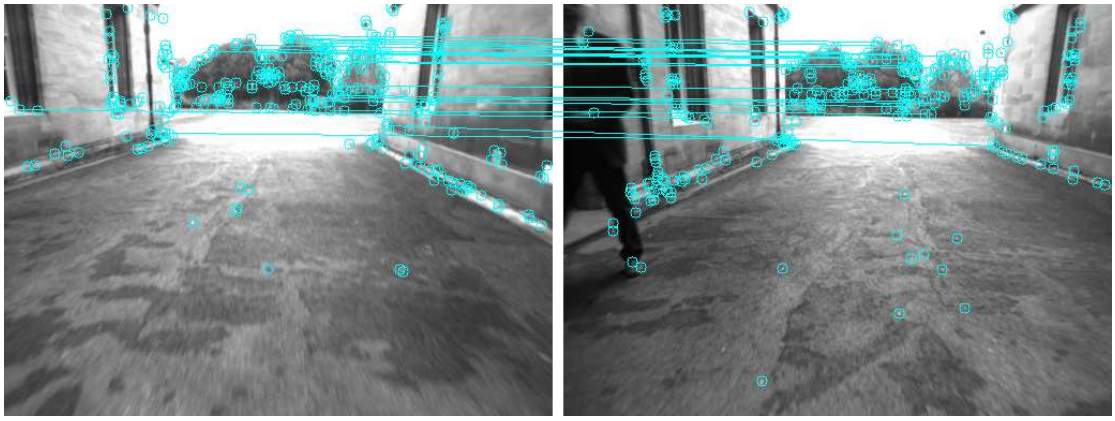


(b) SURF64

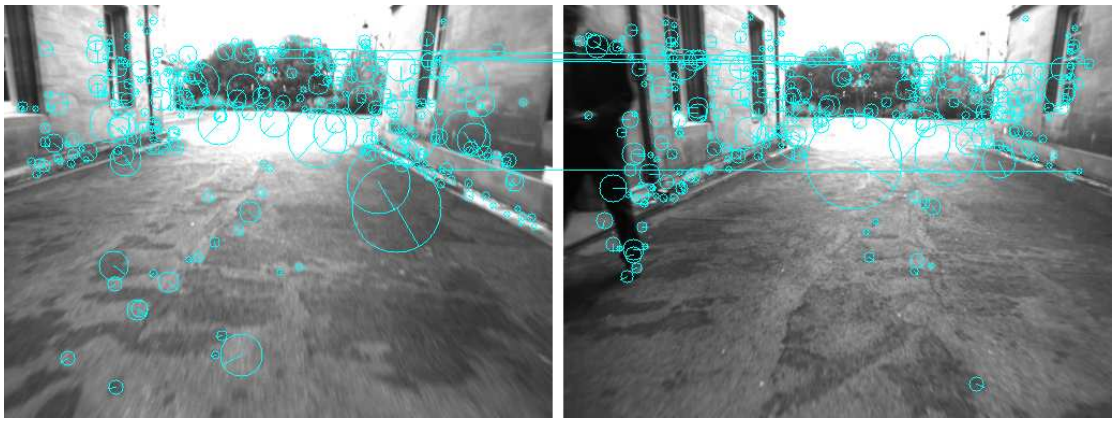
Figure 4.9: Example of words matched by BRIEF and SURF64. In the general case, as in this example, BRIEF provides as many word correspondences as SURF64.

score. In most cases, BRIEF obtains as many correct word correspondences as SURF64, in spite of the slight perspective changes, as shown in the first example in Figure 4.9. In the second example in Figure 4.10, only BRIEF is able to close the loop, since SURF64 does not obtain enough word correspondences. These two examples show that BRIEF finds correspondences in objects that are at a middle or large distance, such as the signs on the wall or the trees in the background. In general, distant objects are present in most of the imagery of our datasets. Since the scale of the key points extracted from distant objects hardly varies, BRIEF is suitable to match their patches. In cases where objects are close to the camera, SURF64 is more suitable because of its invariance to scale changes. However, we observed very few cases where this happened. In the third example in Figure 4.11, the camera tilted, making the image appear rotated in some areas. This along with the scale change prevented BRIEF from obtaining word correspondences. In this case, SURF64 overcame these difficulties and detected the loop.

Our results show that FAST features with BRIEF descriptors, despite of lacking scale and rotation invariance, are almost as reliable as SURF features for loop detection problems with in-plane camera motion. As advantages, not only they are much faster to obtain (13ms per image



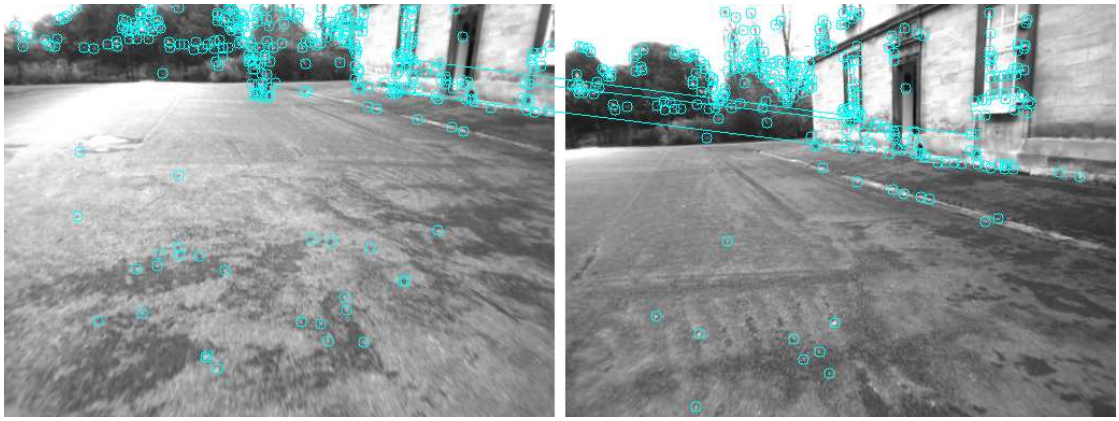
(a) BRIEF



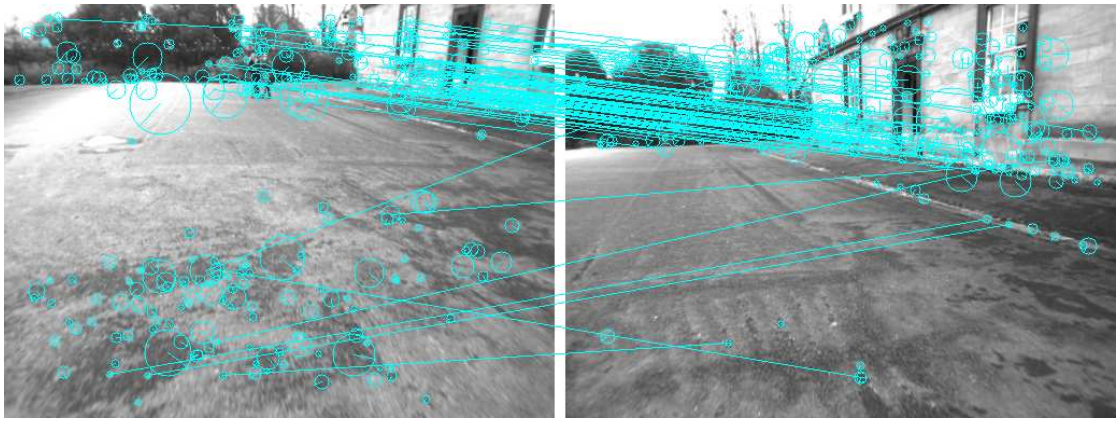
(b) SURF64

Figure 4.10: Example of words matched by BRIEF and SURF64. Only BRIEF is able to close the loop because of the features in the trees. On the other hand, SURF64 do not obtain enough word correspondences.

instead of 100–400ms), but they also occupy less memory (32MB instead of 256MB to store a 1M word vocabulary) and are faster to compare, speeding up the use of the hierarchical vocabulary.



(a) BRIEF



(b) SURF64

Figure 4.11: Example of words matched by BRIEF and SURF64. The scale and orientation change prevent BRIEF from obtaining enough word correspondences, missing the loop. SURF64 manages to provide enough correspondences to detect the loop.

4.4.4 Temporal consistency to match islands

After selecting the features, we tested the number k of temporally consistent matches required to accept a loop closure candidate. For this, we ran our system in the Bicocca25b, NewCollege and Ford2 training datasets with $f = 2$ Hz, for several values of k and α and without any geometrical constraint. We tested k for values between 0 (i.e., disabling the temporal consistency) and 4. We observed a big improvement between $k = 0$ and $k > 0$ for all the working frequencies. As k increases, a higher recall is attained with 100% precision, but this behavior does not hold for very high values of k, since only very long closures would be found. We chose $k = 3$ since it showed a good precision-recall balance in these three datasets. We repeated this test in Bicocca25b for frequencies $f = 1$ and 3Hz as well, to check how dependent parameter k is on the processing frequency. We show in Figure 4.12 the precision-recall curves obtained in Bicocca25b by varying the parameter α; for clarity, only $k = 0$ and 3 are shown. This shows the temporal consistency of islands is a valuable mechanism to avoid mismatches. We can also see that $k = 3$ behaves well even for different frequency values, so that we can consider this parameter stable. In Section 4.4.7 we show that this holds even with a very large working frequency f.

4.4.5 Geometrical verification with direct index

To verify the geometrical consistency of the candidate loop images, we implement the epipolar constraint for single images presented in Section 3.3.1. To accept a loop, we must find with RANSAC a well-conditioned fundamental matrix, supported by at least 12 points, between the two images, computed by means of the direct index. Note that we could use any other method of those presented in the previous chapter, but we decided to use this one to show that with our current approach we are able to obtain robust results even with a weak geometrical verification.

According to Figure 4.12, we could select a restrictive value of α to obtain 100% precision, but this would require to tune this parameter for each dataset. Instead, we set a generic value and verify matches with the geometrical constraint between the two images I_t and $I_{t'}$ of a loop candidate. In view of our tests in the training datasets, we selected $\alpha = 0.3$, because it offered a balanced trade-off between recall increment and precision loss. Note that the precision is improved by the geometrical verification discarding all the false positive cases. As an example, Figure 4.7(a) shows this working point in the Rawseeds datasets. We could ease the α threshold with a lower value, for instance, the 0.15 used previously, but we would require a stronger

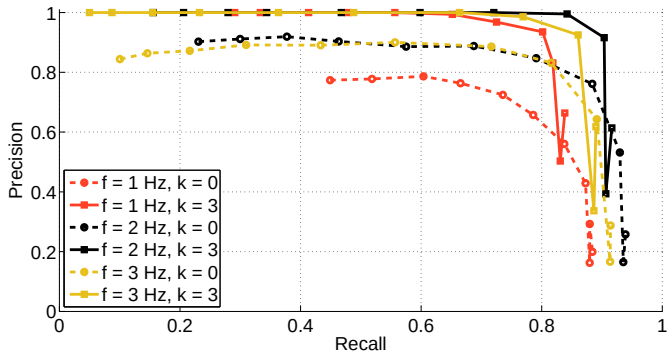


Figure 4.12: Precision-recall curves in Bicocca25b with no geometrical check, for several values of similarity threshold α, number of temporally consistent matches k and working frequency f.

Technique	Recall (%)	Execution time (ms/query)		
		Median	Min	Max
DI_0	38.3	0.43	0.25	16.50
DI_1	48.5	0.70	0.44	17.14
DI_2	56.1	0.78	0.50	19.26
DI_3	57.0	0.80	0.48	19.34
FLANN	53.6	14.09	13.79	25.07
Exhaustive	61.2	14.17	13.65	24.68

Table 4.2: Performance of different approaches to obtain correspondences in NewCollege

geometrical verification, such as the CRF-Matching geometrical verification.

Computing the corresponding points between I_t and $I_{t'}$ is the most time-consuming step of this stage. We compared our proposal of using the direct index to compute correspondences, coined DI_l, with the exhaustive search and a k-d tree approach. The parameter l stands for the level in the vocabulary tree at which the ancestor nodes are checked. In the k-d tree approach, the FLANN library (Muja & Lowe 2009), as implemented in the OpenCV library (OpenCV 2009), is used to build a set of randomized k-d trees with the feature descriptors of I_t. This allows to obtain for descriptors of $I_{t'}$ the approximate nearest neighbors in I_t. After computing distances with any of these methods, the nearest neighbor distance ratio, with a threshold of 0.6 units, was applied. Although both the FLANN and the vocabulary approaches rely on a tree structure, they are conceptually different here: our vocabulary tree was created with training data, so that the neighbor search is based on independent data, whereas the randomized k-d trees are tailored to each $I_{t'}$.

We ran each of the methods in the NewCollege dataset with $f = 2$ Hz, $k = 3$, $\alpha = 0.3$. We selected this dataset because it presents the longest revisited trajectory and many perceptual aliasing cases. In Table 4.2 we show the execution time of the geometrical check per query, along with the recall of the loop detector in each case. The precision was 100% in all the cases. The time includes the computation of corresponding points, the RANSAC loops and the computation of the fundamental matrices. The highest execution time of all the methods was obtained when the maximum number of RANSAC iterations was reached. The exhaustive search achieves higher recall than the other methods, which are approximate, but exhibits the highest execution time as well. We see that the FLANN method takes nearly as long as the exhaustive search method. Around the 60% of this time is dedicated to create the k-d trees, 38% to search for neighbors, and 2% to RANSAC iterations. Although this search time is smaller than that taken by the exhaustive search, we can conclude that the speed-up obtained when computing the correspondences is not worth the cost of building a FLANN structure per image. On the other hand, DI_0 presents the smallest execution time, but also the lowest recall level. This behavior is not surprising, because selecting correspondences only from features belonging to the same word is very restrictive when the vocabulary is big (one million words). Regarding this, DI_1, DI_2 and DI_3 methods offer a good balance between execution time and recall level. Just with DI_1 we are able to obtain a great improvement in the recall with respect to DI_0, with a slight increment in the execution time. DI_2 increases more the recall, requiring similar time. Furthermore, DI_2 outperforms the FLANN approach both in recall and execution time. For $l \geq 3$, the improvement of DI_l in the recall level is not so striking. We finally chose the method DI_2 for our geometrical check since it showed a good balance between recall and execution time.

4.4.6 Execution time

To measure the execution time, we ran our system in the NewCollege dataset with $k = 3$, $\alpha = 0.3$ and DI_2. By setting the working frequency to $f = 2$ Hz, a total of 5266 images were processed, yielding a system execution time of 16 ms per image on average and a peak of less than 38 ms. However, in order to test the scalability of the system, we set the frequency to $f = 10$ Hz and obtained 26292 images. Even with $k = 3$, the system yielded no false positives. This shows that the behavior of the temporal consistency parameter k is stable even for high frequencies.

The execution time consumed per image in that case is shown in Figure 4.13. This was measured on a Intel Core i7 @ 2.67GHz machine. We also show in Table 4.3 the required time of each stage for this amount of images. The *features* time involves computing FAST key points and removing those with low corner response when there are too many, as well as smoothing the image with a Gaussian kernel and computing BRIEF descriptors. The *bag-of-words* time is split into four steps: the conversion of image features into a bag-of-words vector, the database query to retrieve similar images, the creation and matching of islands, and the insertion of the current image into the database (this also involves updating the direct and inverse indexes). The *verification* time includes both computing correspondences between the matching images, by means of the direct index, and the RANSAC loop to calculate fundamental matrices.

We see that all the steps are very fast, including extracting the features and the maintenance of the direct and inverse indexes. This allows to obtain a system that runs in 22ms per image, with a peak of less than 52ms. The feature extraction stage presents the highest execution time; most of it, due to the overhead produced when there are too many features and only the best 300 ones must be considered. Even so, we have achieved a reduction of more than one order of magnitude with respect to other features, such as SIFT or SURF, removing the bottleneck of these loop closure detection algorithms. In the bag-of-words stage, the required time of managing the islands and the indexes is negligible, and the conversion of image features into bag-of-words vectors takes as long as the database query. Its execution time depends on the number of features and the size of the vocabulary. We could reduce it by using a smaller vocabulary, since we are using a relatively big one (1M words, instead of 10–60K as used by other authors, as Paul & Newman (2010)). However, we found that a big vocabulary produces more sparse inverse indexes associated to words. Therefore, when querying, fewer database entries must be traversed to obtain the results. This reduces the execution time strikingly when querying, trading off, by far, the time required when converting a new image. We conclude that big vocabularies can improve the computation time when using large image collections. Furthermore, note that querying a database with more than 26K images takes 9 ms only, suggesting this step scales well with tens of thousands images. The geometrical verification exhibits a long execution time in the worst case, but as we saw in the previous section, this rarely occurs, whereas the 75% of the cases require less than 1.6ms.

Our results show that we can reliably detect loops against databases with 26K images in 52ms (22ms on average). This represents an improvement of one order of magnitude with respect to the 300–700ms required by algorithms based on SIFT or SURF, as our system presented in Chapter 3, or the proposals by Cummins & Newman (2011), Paul & Newman (2010) or Angeli et al. (2008). For example, the state-of-the-art algorithm FAB-MAP 2.0 (Cummins & Newman 2011) needs 423ms for extracting SURF, 60ms for conversion into bag of words, 10ms for retrieving matching candidates against 25K images, and 120ms (worst case) for RANSAC geometric verification. Our algorithm also outperforms the extremely efficient loop detector developed by Konolige et al. (2010), based on compact randomized tree signatures. According to their figure 6, the method requires around 300ms to perform the complete loop detection against a database with 4K images.

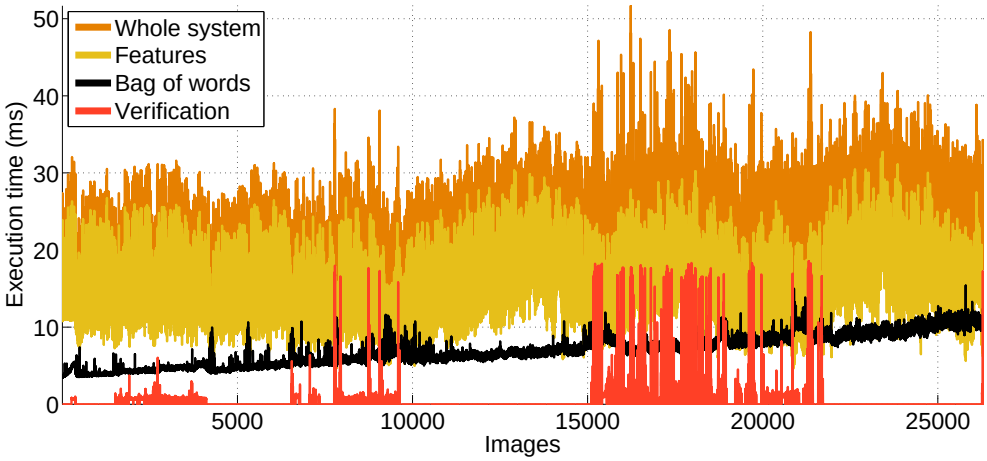


Figure 4.13: Execution time in NewCollege with 26292 images.

		Execution time (ms/query)			
		Mean	Std	Min	Max
Features	FAST	11.67	4.15	1.74	30.16
	Smoothing	0.96	0.37	0.79	2.51
	BRIEF	1.72	0.49	1.51	4.62
Bag of words	Conversion	3.59	0.35	3.27	8.81
	Query	3.08	1.91	0.01	9.19
	Islands	0.12	0.04	0.08	0.97
	Insertion	0.11	0.02	0.06	0.28
Verification	Correspondences and RANSAC	1.60	2.64	0.61	18.55
Whole system		21.60	4.82	8.22	51.68

Table 4.3: Execution time in NewCollege with 26292 images

4.4.7 Performance of the final system

In previous sections we showed the effect of the parameters of our system in the correctness of the results. For our algorithm we chose the generic parameters $k = 3$, $\alpha = 0.3$, and the DI_2 method for computing correspondences, since they proved effective under several kinds of environments in the training datasets. We ran our complete system in all the datasets but those with some overlap with the training images (Bovisa04 and Bovisa06). A summary with the parameters of the algorithm and the vocabulary is shown in Table 4.4. In Figure 4.14 we show the precision-recall curves obtained in these datasets with these parameters, processing the sequences at $f = 2$Hz. In Table 4.5 we show the figures of those curves with the final configuration. We achieved a high recall rate in the three datasets with no false positives. The results obtained in datasets NewCollege and Bicocca25b are shown in Video 1.a. and Video 1.b., listed in Section 1.4.2.

In order to check the reliability of our algorithm with new datasets, we used Malaga6L and CityCenter as evaluation datasets. For these, we used our algorithm as a black box, with the default configuration given above and the same vocabulary. For Malaga6L, we processed the sequence at $f = 2$ Hz, and for CityCenter, we used all the images, since these are already taken

FAST threshold	10
BRIEF descriptor length (L_b)	256
BRIEF patch size (S_b)	48
Max. features per image	300
Vocabulary branch factor (k_w)	10
Vocabulary depth levels (L_w)	6
Disallow local (τ_c)	20s
Min. score with previous image ($s(\mathbf{v}_t, \mathbf{v}_{t-\Delta t})$)	0.005
Temporally consistent matches (k)	3
Normalized similarity score threshold (α)	0.3
Direct index level (l)	2
Min. matches after RANSAC	12

Table 4.4: General parameters of our system

far apart. We also compared our algorithm with the state-of-the-art FAB-MAP 2.0 algorithm (Cummins & Newman 2011), configured by default as it is available in its authors' website. Given a query image, FAB-MAP returns a vector with the probability p of being at the same place than some previous image. Only those matches with p higher than a threshold are accepted. This parameter must be set by the user. We chose $p \geq 98\%$ because it showed the highest recall for 100% precision in these datasets. Table 4.5 and Table 4.6 show the results in the evaluation datasets. For sake of fairness, we remark on how this comparison was performed: FAB-MAP 2.0 software does not apply any geometrical constraint to the returned matches by default, so we applied a verification stage similar to ours, consisting in computing a fundamental matrix with the exhaustive search method. The input for FAB-MAP 2.0 must be a sequence of disjoint images. For Malaga6L, we fed it with images taken at frequency 1Hz. We also tried 0.25 and 0.5Hz, but 1Hz yielded better results. For CityCenter, we used all the available images. Finally, FAB-MAP 2.0 provides a vocabulary of 11K words of 128 float values, built from outdoor disjoint images, whereas our vocabulary contains 1M words of 256 bits, created from a sequence of images.

As shown in Table 4.5, our algorithm with the parameters by default is able to achieve large recall with no false positives in both evaluation datasets. Our recall level is similar to that yielded by FAB-MAP 2.0, but with lower execution time. In the Malaga6L dataset, all the

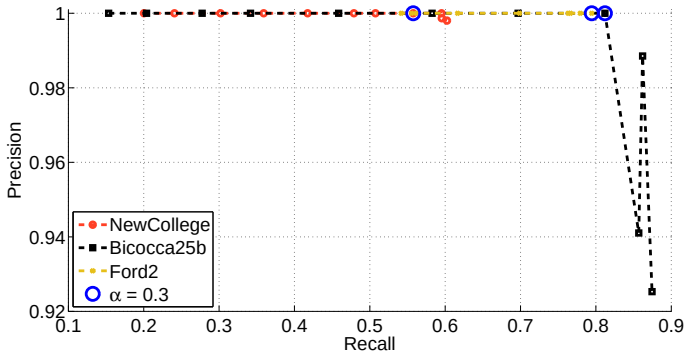


Figure 4.14: Final precision-recall curves in the training datasets with $f = 2$Hz, with the selected working point $\alpha = 0.3$.

Dataset	# Images	Precision (%)	Recall (%)
NewCollege	5266	100	55.92
Bicocca25b	4924	100	81.20
Ford2	1182	100	79.45
Malaga6L	869	100	74.75
CityCentre	2474	100	30.61

Table 4.5: Precision and recall of our system

Dataset	# Images	Min. p	Precision (%)	Recall (%)
Malaga6L	462	98%	100	68.52
CityCentre	2474	98%	100	38.77

Table 4.6: Precision and recall of FAB-MAP 2.0

loops are correct in spite of the illumination difficulties and the depth of the views. The results in CityCenter differ between our method and FAB-MAP 2.0 because the change between loop closure images is bigger than that in other datasets. This hinders the labor of the DI_2 technique because features are usually more distinct and are separated in early levels in the vocabulary tree. Note that this highlights the little invariance of BRIEF, since others as SURF may be able to produce more similar features between the images. Anyhow, we see that our method is still able to find a large amount of loop events in this dataset, and that the normalized similarity score η_{L_1} behaves correctly in spite of the little overlap between consecutive images. This test shows that our method can work fine out of the box in many environments and situations, and that it is able to cope with sequences of images taken at low or high frequency, as long as they overlap. We can also remark that the same vocabulary sufficed to process all the datasets. This suggests that the source of the vocabulary is not so important when it is big enough.

We show in Figure 4.15 the detected loops in each dataset. No false detections were fired. The trajectory in NewCollege is based on partially corrected GPS data, so that some paths are inaccurately depicted. Note that part of the vehicle where the camera is mounted is present in all the images of Ford2; we removed the features that lay on it. We see that detecting 55.92% of the loop events is enough to, for example, widely cover all the loop areas in a long trajectory as that of NewCollege. On the right hand side of Figure 4.15, we show examples of correct loop detections in the training and evaluation datasets, with the final corresponding features. These examples make the limited scale invariance of BRIEF descriptors apparent. Most of the features matched are distant, as we noticed in Section 4.4.3. The scale change that BRIEF tolerates is shown in the correspondences that are close to the camera in NewCollege and Bicocca25b, and those on the cars in Malaga6L. However, BRIEF cannot handle such a large scale change as that produced on the car in CityCenter, where all correspondences were obtained from distant features. On the other hand, whenever features are matched in background objects, a loop can be detected despite medium translations. This is visible in CityCenter and Ford2, where the vehicle moved along different lanes of the road.

4.5 Discussion

In this chapter, we presented several enhancements for our technique to detect loops in monocular sequences. The main conclusion of our work is that binary features are very effective and extremely efficient in the bag-of-words approach.

It is worth mentioning that our vocabulary tree is one method among others to discretize binary spaces, and that these have drawbacks due to the thick boundaries of binary Voronoi regions (Trzcinski et al. 2012). In short, the problem arises because in Hamming spaces, unlike Euclidean ones, the points which are equidistant to any other two points occupy a large part of the space, making clustering more difficult. In the next chapter, we modify our vocabulary tree to deal with this issue. Despite this difficulty, our results demonstrate that FAST+BRIEF features are as reliable as SURF (either with 64 dimensions or with 128 and without rotation invariance) for solving the loop detection problem with in-plane camera motion, the usual case in mobile robots. In addition, the execution time and memory requirements are one order of magnitude smaller, without requiring special hardware.

We showed the reliability and efficiency of our proposal on seven very different public datasets depicting indoor, outdoor, static and dynamic environments, with frontal or lateral cameras. Departing from most previous works, to avoid over-tuning, we restricted ourselves to present all results using the same vocabulary, obtained from an independent dataset, and the same parameter configuration, obtained from a set of training datasets, without peeking on the evaluation datasets. So, we can claim that our system offers robust and efficient performance in a wide range of real situations, without any additional tuning.

In particular, we concluded that a single big vocabulary (we used 1 million words) is enough to process very different datasets. In fact, a big vocabulary tree does lower the execution time when accessing large image databases with an inverse index. We also shown a novel way of using a direct index along with the vocabulary tree to speed up the geometrical verification stage. We decrease the execution time by computing correspondences only between features with common nodes in the vocabulary tree, trading off the execution time of this stage with the recall rate of our system. Furthermore, this technique performed better than an approximate nearest neighbor approach (Muja & Lowe 2009) for individual images. Our experiments showed that the quantitative loss of recall is small when using the direct index, but it is even less significant if we think of the meaning behind the figures. A lower execution time allows to increase the working frequency of the loop detector. Even if the number of loops correctly fired drops in relative terms, the total amount of detections can be higher than that of higher recall at a lower latency. In addition, acquiring images faster raises the chance of finding a challenging loop closure, since more images of the same place are queried.

The main limitation of our technique is the use of features that lack rotation and scale invariance. It is enough for place recognition in indoor and urban robots, but surely not for all-terrain or aerial vehicles, humanoid robots, wearable cameras, or object recognition. However, our demonstration of the effectiveness of the binary bag-of-words approach paves the road for the use of new and promising binary features such as FREAK (Alahi et al. 2012), ORB (Rublee et al. 2011), or BRISK (Leutenegger et al. 2011), which outperform the computation time of SIFT and SURF, maintaining rotation and scale invariance.

A fast loop detector is not just useful for urban robots, but for other computer vision applications as well. For instance, hand-held camera SLAM can be performed in low-capability devices (Klein & Murray 2009); these algorithms can profit from real-time data association provided by a fast loop detector. A fast SLAM algorithm and loop detector, along with fast object recognition can provide real-time semantic mapping. Aiming at this purpose, in the next chapter, we propose a system to perform video-rate 3D object recognition.

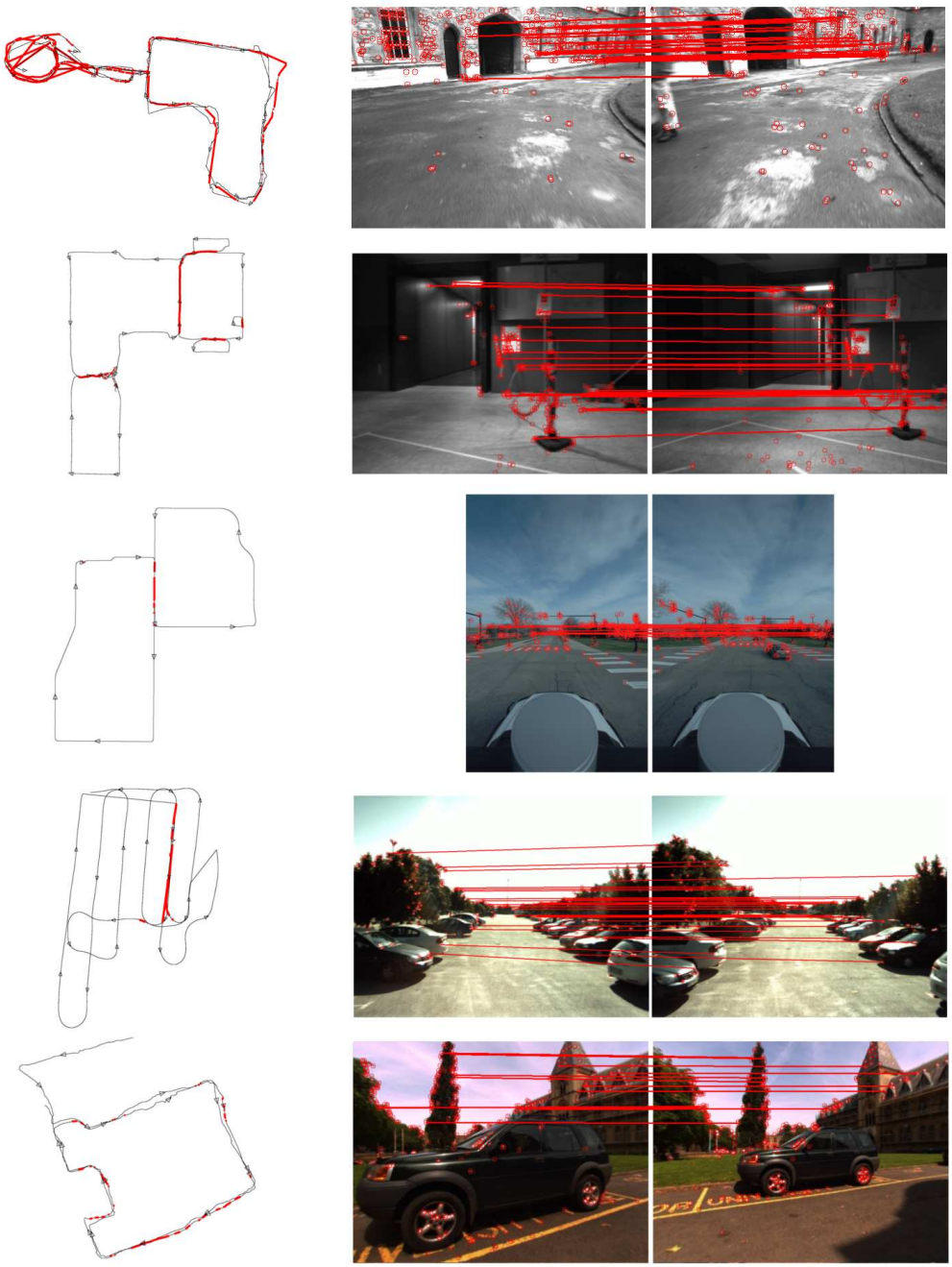


Figure 4.15: Loops detected by our system in the five datasets (from up to down: NewCollege, Bicocca25b, Ford2, Malaga6L, CityCentre), with some examples of correct loops detected in scenes with motion blur and slight scale and perspective change. On the right hand side, lines depict final corresponding features. On the left hand side, the trajectory of the robot is depicted with thin black lines in new places, and with thick red lines in revisited areas. There are no false positives in any case.

Chapter 5

3D Object recognition for visual SLAM

5.1 Object recognition pipeline

In this chapter we present our object detection and recognition algorithm for visual SLAM. The goal of our work is to provide a reliable detector, able to achieve real-time performance to place objects as high-level entities in a monocular SLAM map, producing *semantic maps*.

We make use of two approaches to address this problem. First we introduce a baseline technique in which different views of objects are modeled individually with SURF features. We presented this standard recognition approach used together with a monocular SLAM system to build 3D semantic maps in Civera et al. (2011). This baseline technique, although effective, has some lacks: it is not scalable before a large number of objects and views, and it does not achieve video-rate execution time. To overcome these shortcomings, we fit the object recognition problem in our large-scale image matching framework. For that, we propose a second technique in which all the views of the objects are modeled with a single bag of binary words, indexed and retrieved later from an efficient database with inverted and direct indexes.

Our two approaches work in an appearance recognition basis, for which object surface must be textured to some extent. The appearance matching is performed by means of local images features, so that it is robust to partial occlusion and some image transforms. Objects are recognized from single gray-scale images. As a result of detecting an object in an image, its pose in the 3D space with respect to the camera is obtained. The 3D pose permits placing the object in a 3D SLAM map; furthermore, this information together with the camera calibration suffices to locate the bounds of the object in the 2D image.

The algorithms of the two approaches can be outlined in a similar way, by defining two stages: first, batches of training images are processed offline to build models of individual objects, and second, input images gathered online from a video stream are searched for recognizable objects. This pipeline is illustrated in Figure 5.1 and detailed next:

1. Object modeling:

   (a) Training image processing: sets of training images depicting several points of view of an object are searched for local features, which are matched across views.

   (b) Geometric and appearance modeling: matching features are used to triangulate covisible points to obtain a 3D point cloud of the object. Points are associated to

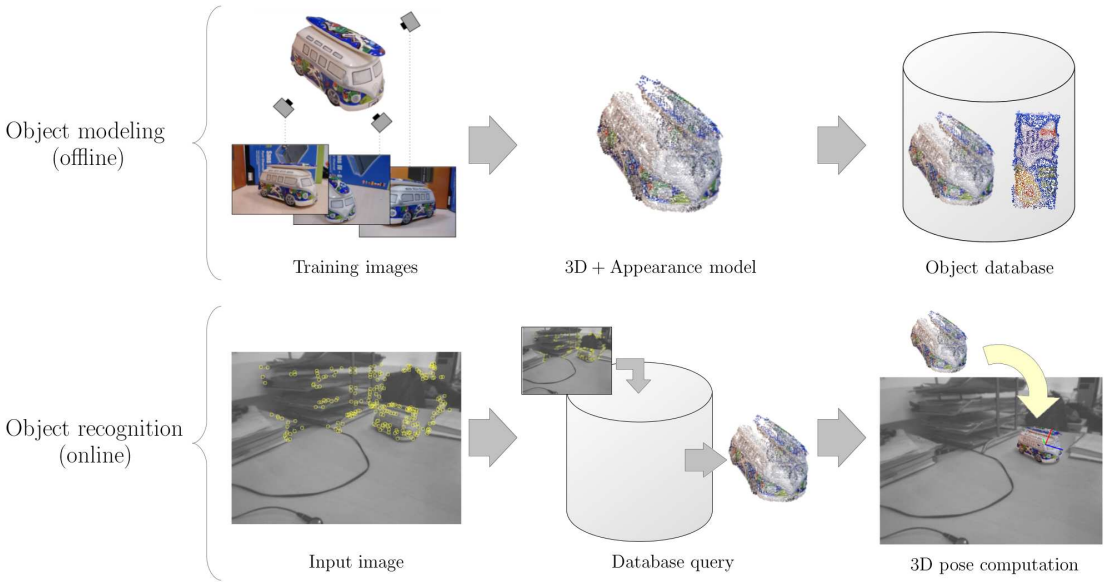


Figure 5.1: Object recognition pipeline. There are two main stages: object modeling (top row), performed offline only once per object, and object recognition (bottom row), performed online for each video frame.

appearance information acquired from image feature descriptors.

(c) Object database: objects models are stored together in a single database.

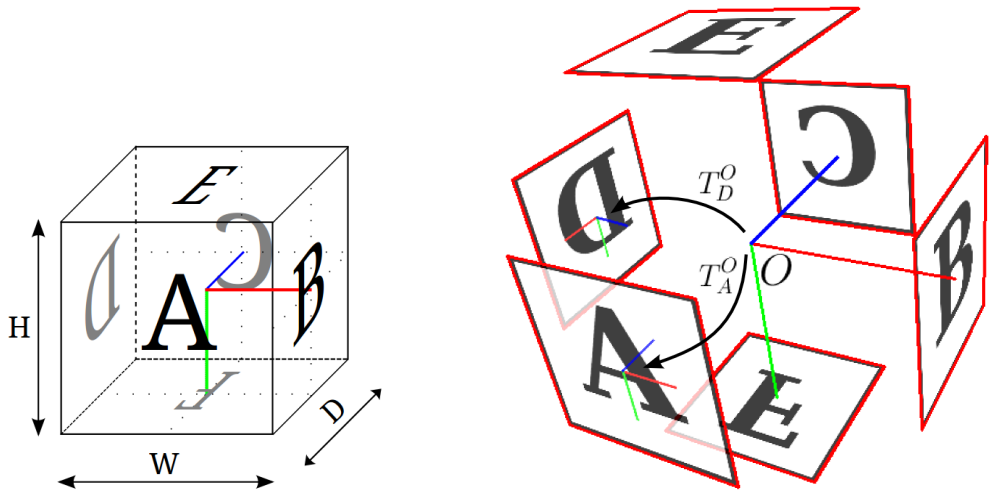
2. Object recognition:

(a) Input image: geometrical and appearance data, in form of image features, is obtained from a video frame. Optionally, features can be separated into regions of interest to focus on local regions individually.

(b) Database query: the object database is queried with the appearance data of the input image, and one or several object models are returned as candidates according to their similarity with the input.

(c) Model verification and 3D pose computation: the geometrical data of candidate models are checked for consistency with the location of the input features. If there is consistency, the pose of the object in the scene is obtained. Otherwise, the candidate is rejected.

The two approaches we consider differ in how the appearance of the objects is modeled and, hence, in how the object retrieval is carried out when the object database is queried. After explaining how we segment regions of interest in the next Section 5.4.4.1, we introduce both modeling techniques in Section 5.2, and detail the two object retrieval algorithms in Section 5.3 and Section 5.4.



(a) Cuboid model (b) Frames of reference

Figure 5.2: A planar cuboid-shaped model is defined by six ordered views and the dimensions of the object in metric units, as shown in (a). The cuboid condition determines exactly the position of any pixel in the 3D space, expressed in the object frame of reference O, located in the center of the cuboid.

5.2 Object modeling

Our object models are composed of geometrical and appearance data. The geometric information is given by the 3D coordinates of points belonging to the surface of the object. Appearance is related to local image feature descriptors associated to the 3D points, so that models are fully described by image descriptors with 3D coordinates attached to them. Depending on the recognition process, models can be further detailed, as we explain in the next section.

5.2.1 3D point cloud model

The geometry of the object is extracted from sets of training images that depict different points of view of the object to model. As a result, the geometry information is represented as a cloud of 3D points of coordinates expressed in the local object frame of reference.

We manage two different methods to create point clouds that determine the number and source of training images. The first method imposes a planarity restriction to the surfaces, so that just one image per planar view of the object suffices to determine the 3D coordinates of all its points. This method is suitable for planar or box-shaped objects, such as cards, posters, cartons, boxes, etc. The second method does not constrain the shape of the object and estimates it by multi-view geometry (Hartley & Zisserman 2004). In this case several unordered pictures of the object are necessary to capture as much structure as possible. This method is utilized to model non-planar and generic objects.

5.2.1.1 Planar geometry

We define a planar object as that shaped as a cuboid hexahedron, i.e. with six planar views which are pairwise equally-sized, as illustrated in Figure 5.2(a). To create such a model we

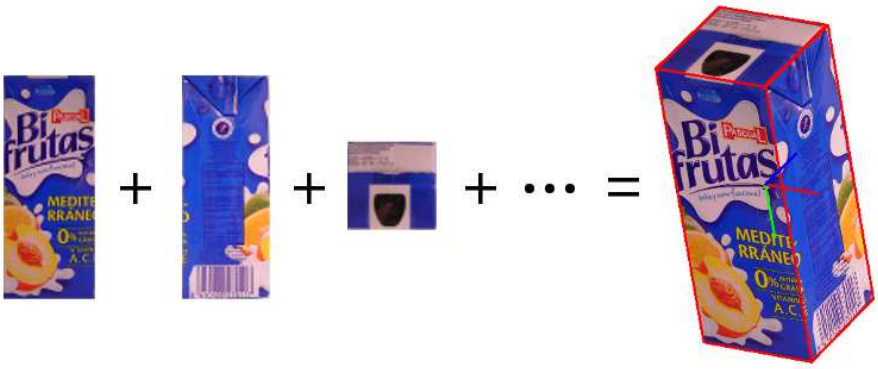


Figure 5.3: A juice box modeled as a planar object

require six training images, one per surface, from A to F, and the width, height and depth of the final object, denoted W, H and D, respectively, in metric units. By imposing the planarity restriction, we can compute the 3D coordinates of any pixel in the training images expressed in a local frame of reference, located in the center of the image, as shown in Figure 5.2(b). Later, using the cuboid condition, we can transform that point to the object frame of reference, denoted O and located in the center of the cuboid. The cuboid condition fixes the location of views with respect to the object frame, so that the transformations $\mathbf{T}_A^O$, ..., $\mathbf{T}_F^O$ are known. Therefore, given a pixel $\mathbf{p} = (u, v)^\top$ from surface A, without loss of generality, its 3D point $\mathbf{p}^O$ is given by

$$\mathbf{p}^O = \mathbf{T}_A^O \cdot \begin{pmatrix} s_1^A/w_A & 0 & 0 & 0 \\ 0 & s_2^A/h_A & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u - w_A/2 \\ v - h_A/2 \\ 0 \\ 1 \end{pmatrix}, \tag{5.1}$$

where w_A and h_A stand for the width and height of the image in pixels, and s_1 and s_2 are two of the real dimensions of the object in metric units, W, H or D depending on the surface index. For surface A, $s_1^A = W$ and $s_2^A = H$. All the pixels of the training images, transformed into the object frame of reference, form the point cloud that comprises the geometrical information of the object model.

A juice box modeled planar is shown in Figure 5.3. We can also build cuboid models ignoring some views. This is useful if we cannot provide some training images of the object, as for example the back part of a wardrobe, or if the object is just a planar surface, as a postcard. In such a case, we can define the point cloud just with an image and the width and height of the object.

5.2.1.2 General geometry

Objects that have an arbitrarily complex shape can be modeled with a general technique that triangulates points present in several images by means of multi-view geometry. For that, we require several images that cover different points of view of the object. The only condition to satisfy is that there is enough overlap between images to be able to obtain 3D points by triangulation. General images with background can be used. In that case, image masks are required to segment out the object. Note that the process of separating object and background could be performed in an autonomous way, for example by finding salient regions in the depth 3D points provided by a RGBD camera or a laser sensor, or by rotating the object on a calibrated

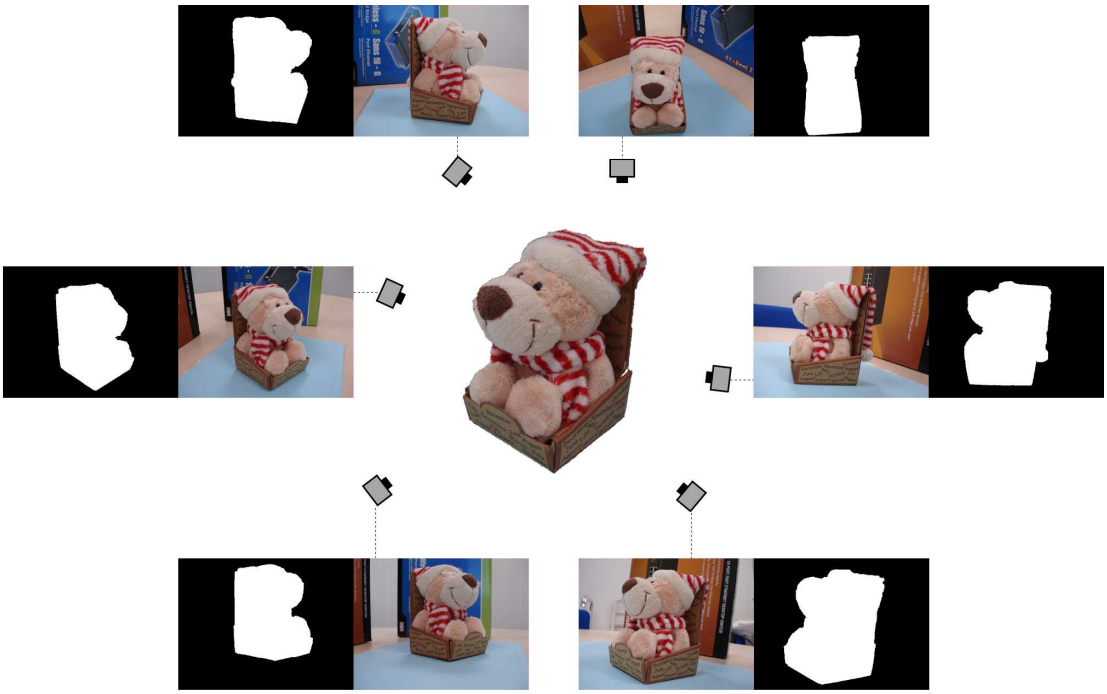


Figure 5.4: A set of images taken around the object are used to build the model. Images are masked to remove background. Any camera and any set of images can be used, as long as there is enough overlap between views.

platform at an known position (Pangercic et al. 2011). Nevertheless, we segmented images manually because it does not require a complicated set-up and makes the process less error-prone. In fact, our training images can be acquired by any customer camera, independently of those images processed online later by visual SLAM. Figure 5.4 shows an example of some images used to create the model of a toy lion.

We compute the 3D point cloud of the surface of objects making use of Bundler software (Snavely et al. 2006). Bundler runs a structure-from-motion algorithm over an unordered collection of images to estimate the geometry of the scene between pairs of images using robust bundle adjustment algorithms (Triggs et al. 2000). A non-linear optimization step finally produces the 3D reconstruction of a sparse set of scene points, as well as the relative motion between the training cameras. Bundler starts by extracting SIFT features (Lowe 2004) from the images to match them and obtain corresponding points between views, which are the input of the bundle adjustment. Although Bundler uses SIFT, this does not determine the features we use to represent the appearance of our models in the end. At this stage, we do not consider the masks of the training images and use the full images. The background is useful in this case because it encompasses a richer scene structure that provide more information for the optimization step, minimizing the 3D reconstruction uncertainty.

The 3D points yielded by Bundler cover the object surface sparsely. We use then the multi-view stereo reconstruction algorithm by Furukawa & Ponce (2010), implemented by the PMVS software, to obtain a dense point cloud that fully encompasses the object. Given the collection of training images and the extrinsic parameters of their cameras, provided by Bundler, the

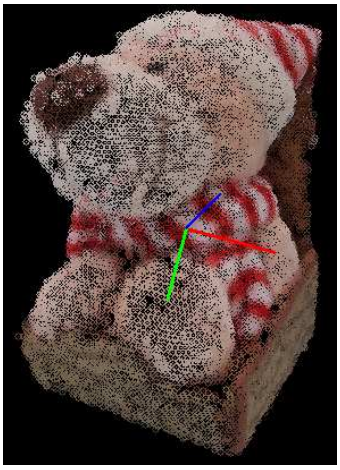


Figure 5.5: A toy modeled as a general 3D object

PMVS algorithm starts by matching small 7×7 patches yielded by a Harris detector (Harris & Stephens 1988) across the training images. By means of these correspondences, it then spreads these patches to nearby pixels in an iterative manner, obtaining a dense patch collection. In a final step, PMVS applies visibility constraints to filter out incorrect matches. PMVS outputs a collection of small rectangular patches that cover the scene comprehensively with covisibility information. Due to matching and reconstruction errors, the 3D coordinates of some patches are not well estimated. We impose a simple condition to avoid those cases and dismiss the patches which are not seen from at least 3 training images. We consider the centers of the resulting patches as the initial set of 3D points.

We use now the mask of the training images to filter out those 3D points that belong to the background. For that, we take the 3D coordinates of each point, given by PMVS, and the extrinsic camera parameters of each training image from which the point is seen, given by Bundler, and project the point on the image. If the projected 2D point is not inside the corresponding masks, the 3D point is considered background and removed. The surviving points form the point cloud that comprises the geometrical information of the object model. Finally, an arbitrary local frame of reference in the 3D space is created for the model. It is set in the centroid of the point cloud, with the three orthogonal axes oriented along its principal directions. Figure 5.5 shows the point cloud obtained for the training images of the toy lion.

Since only monocular information is considered as input to create object models, the scale of the observed scenes cannot be fixed. This means that the 3D coordinates of the points are at an arbitrary scale, so the translation of the computed pose of a recognized object is up to scale. The 2D area occupied by an object in an image can be retrieved even in those cases. However, we set the real scale manually by measuring the real distance between two points of the object. This enables the object models to be used in visual SLAM applications.

5.2.2 Approaches for modeling appearance

In addition to geometrical information, object models contain appearance data, used for detecting objects in images. We define object appearance by image local features, as we did to address the place recognition problem. We apply two different methods in our research to use image features, starting from the same basis. We extract features from the training images, masking out those

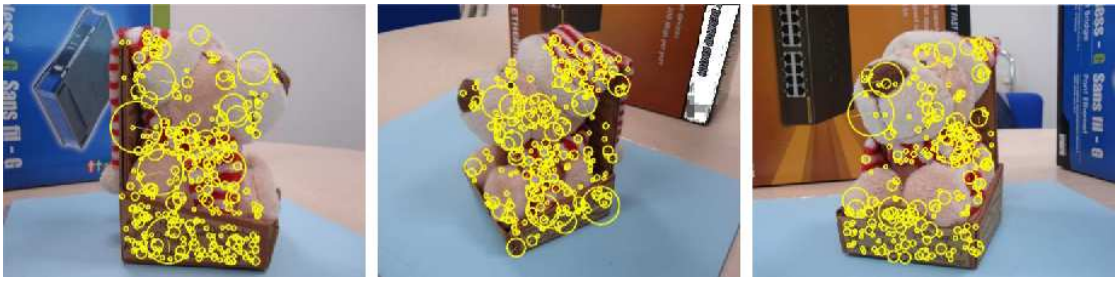


Figure 5.6: Some of the images used to build a model of a lion toy. Notice the SURF features used for recognition superimposed over the images. Sets of k-d trees are built over them to provide correspondence computation.

which lie in the background or too close to the object bounds, if necessary. Later, we will use the same kind of features in the online recognition process. For each training image, we look for the visible 3D point whose back projection minimizes the 2D distance to each feature. If a feature is at most at 3 pixels from a 3D point, the feature is associated to its spatial coordinates; otherwise, it is dismissed. This operation results in a set of image features associated to 3D coordinates in the object frame. This information suffices to compute a transformation from the camera of an input image to the object after establishing feature correspondences, as we explain in Section 5.5. It should be remarked that the number of image features is commonly lower than the number of 3D points in the geometrical model, so that there are points with no appearance information and the point cloud is denser than necessary. Although it is not used for recognition, we keep the complete point cloud in our applications for visualization purposes. Furthermore, we think that a dense model like this one could be of importance for other robotic applications, like grasping.

The way in which the model image features are structured defines how the models are stored in the object database and how the recognition process is carried out. We researched two approaches to model object appearance. Our first proposal is a recognition based on independent views of the object, i.e. individual sets of covisible SURF features (Bay et al. 2008). In this, the SURF set extracted from each training image is stored individually in the object model, so that when the database is queried, sets of correspondences with each view of each object are computed. In our second proposal, we used binary features to jointly describe the whole surface of the object, which is finally represented by a bag of binary words, fusing this method with the approach we used in the previous chapter to address the place recognition problem. This involves that all the views of the object are treated as a single entity with a general appearance. Next sections go into these approaches in depth.

5.3 Recognition based on independent views

In the independent view approach, the appearance of an object is represented as disjoint sets of SURF features. Then, for recognition, SURF correspondences are obtained sequentially for each view to select the candidate object model.

Figure 5.7: Database of independent views of objects

5.3.1 Appearance as disjoint sets of SURF features

As standard cameras usually do not capture a whole object in an image –you never see the front and the back of an object at the same time–, the object model is divided into *views*. We name *view* V the tuple $\langle \mathcal{F}_V, \mathcal{X}_V^O, \mathcal{K}_V \rangle$, composed of a set of SURF features $\mathcal{F}$ obtained from the training image, the set of the 3D coordinates $\mathcal{X}_V^O$ associated to each SURF feature, in terms of the object frame reference O, and a set of randomized k-d trees built over the features $\mathcal{F}_V$. Then, in this approach, the appearance of an object is modeled as a set of views

$$\mathcal{O} \equiv \{\langle \mathcal{F}_1, \mathcal{X}_1^O, \mathcal{K}_1 \rangle,\ \langle \mathcal{F}_2, \mathcal{X}_2^O, \mathcal{K}_2 \rangle,\ \ldots\} \tag{5.2}$$

created from each training image. These will be used independently to perform the object recognition in a per-view basis. When a query image is compared with a view V, corresponding SURF features with $\mathcal{F}_V$ are computed. An exhaustive search to measure distances between descriptors can be slow, as we showed in our previous chapter. To speed up this process, we build a set of randomized k-d trees as created by the FLANN library (Muja & Lowe 2009) with the SURF features of each view to retrieve those that approximately minimize the distance in the descriptor space to given query descriptors. Figure 5.6 shows three of the views that compose the object model of the toy lion presented above. These three images are a subset of the twenty that were used to construct the model, covering several points of view. The SURF features are drawn in the images as yellow circles. A set of k-d trees is built over each set of SURF features.

The object database of this approach is composed of models $\mathcal{O}_1$, $\mathcal{O}_2$, ..., which, in turn, are composed of their views, as illustrated in Figure 5.7. All these will be accessed individually in a sequential matching to perform recognition.

Algorithm 3: Object recognition by sequential view matching

Input : Query image I, Object database $\mathcal{B} = \{\mathcal{O}_1, \mathcal{O}_2, \dots\}$

Output: Objects recognized $\mathcal{R} = \left\{\langle \mathcal{O}_{i_1}, \mathbf{T}_C^{O_{i_1}} \rangle, \langle \mathcal{O}_{i_2}, \mathbf{T}_C^{O_{i_2}} \rangle, \dots\right\}$

$\mathcal{R} \leftarrow \varnothing$;
$\mathcal{F}_I \leftarrow$ features(I);
foreach $\mathcal{O} \in \mathcal{B}$ **do**
  $\mathcal{C} \leftarrow \varnothing$
  foreach $\langle \mathcal{F}_i, \mathcal{X}_i^O, \mathcal{K}_i \rangle \in \mathcal{O}$ **do**
    $\mathcal{C}_i \leftarrow$ correspondences($\mathcal{K}_i$, $\mathcal{F}_I$)
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle i, \mathcal{C}_i \rangle\}$
  end
  Arrange $\mathcal{C}$ by number of correspondences in descending order
  foreach $\langle i, \mathcal{C}_i \rangle \in \mathcal{C}$ **do**
    $\langle$inliers, $\mathbf{T}_C^O\rangle \leftarrow$ transformation($\mathcal{F}_I$, $\mathcal{F}_i$, $\mathcal{X}_i^O$, $\mathcal{K}_i$)
    if there are enough inliers **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \left\{\langle \mathcal{O}, \mathbf{T}_C^O \rangle\right\}$
      Skip the remaining sets of correspondences in $\mathcal{C}$
    end
  end
end
return $\mathcal{R}$

5.3.2 Sequential object view model recognition

The method to recognize objects with sequential matching is depicted in Algorithm 3. Given a query image I from a video sequence, it starts by extracting SURF features from it. For each object $\mathcal{O}$ in the database, putative correspondences are calculated between I and its views V by applying the neighbor ratio described in equation (3.12) in page 27. For this, distances between I features and each set $\mathcal{F}_V$ are computed by means of the k-d trees $\mathcal{K}_V$. These correspondences are then checked to be geometrically consistent, testing first those views with higher number of initial correspondences.

As explained later in Section 5.5, the RANSAC algorithm is run to find a subset of at least 5 correspondences between different SURF features that describe a valid transformation between the image I and the object view. The Perspective-n-Point (PnP) problem is solved to estimate the transformation $\mathbf{T}_C^O$ from the camera C that acquired I to the object frame of ference O. However, for planar objects, a homography is estimated instead, using the DLT algorithm (Hartley & Zisserman 2004). The correspondences which are not consistent with the transformation or the homography are rejected. If we obtain a valid transformation between I and a view, we stop searching the rest of the views of the current object $\mathcal{O}$, as it has already been found.

5.4 Real-time recognition based on a single surface point cloud

In the single surface approach, the appearance of an object is represented as a single set of binary features that cover all the potential points of view of the object. These are then compacted as

a bag-of-words vector of binary words, so that it is possible to create a database with inverted and direct indexes to speed up object retrieval and recognition, just as we tackled the place recognition problem in Chapter 4. We also improve the process by separating input features into regions of interest (ROI) by performing a novel two-stage clustering, querying the object database with each ROI individually. Finally, a transformation is computed with RANSAC and PnP .

5.4.1 Rotation-aware BRIEF features

We use Rotation-aware BRIEF features (ORB) (Rublee et al. 2011) as key points to perform object recognition in this approach. ORB features are built over FAST key points and provide binary descriptors, as BRIEF.

The BRIEF descriptor is computed according to a fixed pattern of pixel comparisons that does not have image rotations into account, so it is not robust against these transforms. In an object recognition scenario, this lack of rotation invariance becomes a drawback, since we do not know beforehand the pose of objects in the scene. ORB overcomes this shortcoming by calculating a dominant orientation between the center of the key point and the intensity centroid (Rosin 1999) of its patch. The comparison pattern is then rotated accordingly so that the descriptor computed later is little prone to vary when the image rotates in the bidimensional plane.

The ORB descriptor is a binary string of 256 bits. Each bit is given a value depending on the result of comparing two pixels around the key point. ORB differs from BRIEF in the way the comparison pattern is obtained. BRIEF selects random pairs from the patch area in advance, imposing a condition in the distance between the comparison points in some cases. To be able to compare two BRIEF features, the random pattern must be retained. ORB avoids that because it makes use of a fixed comparison pattern obtained by learning the pairs that provide the highest bit variance in a set of training patches. Maximizing this variance produces less correlated bits, enhancing feature descriptiveness. Scale invariance is not directly addressed, but Rublee et al. (2011) suggest to ease this by extracting ORB features after scaling the image at several levels.

These properties make ORB a robust binary feature, fast to compute and with a discriminative power comparable to that of SIFT or SURF in some cases (Rublee et al. 2011).

5.4.2 Bag-of-words surface model

In order to define the model appearance, we extract ORB features from all the training images and associated them with 3D points from the point cloud, as previously explained.

Since objects can appear at any scale and point of view during recognition, we initially associate each 3D point to several ORB descriptors to mitigate their lack of scale invariance. These are extracted at different scale levels (up to 2 octaves) from several training images, as shown in Figure 5.8. It is possible to get several similar descriptors on the same point if the camera did not move enough between two training images. To prevent a 3D point from being over-represented, we merge those descriptors which are very close in the descriptor space and, therefore, are not likely to represent a different enough point of view. We achieve this by converting features into visual words and keeping only one descriptor per 3D point and visual word. When several descriptors satisfy this condition, we keep the median binary vector, as this has shown good results (Sattler et al. 2011). Therefore, a 3D point p is finally described by its 3D coordinates $\mathbf{X}^O$ in the object frame of reference O and several pairs of words w and descriptors d:

$$p \equiv \langle \mathbf{X}^O, \{\langle w_1, d_1\rangle, \langle w_2, d_2\rangle, \dots\}\rangle. \tag{5.3}$$

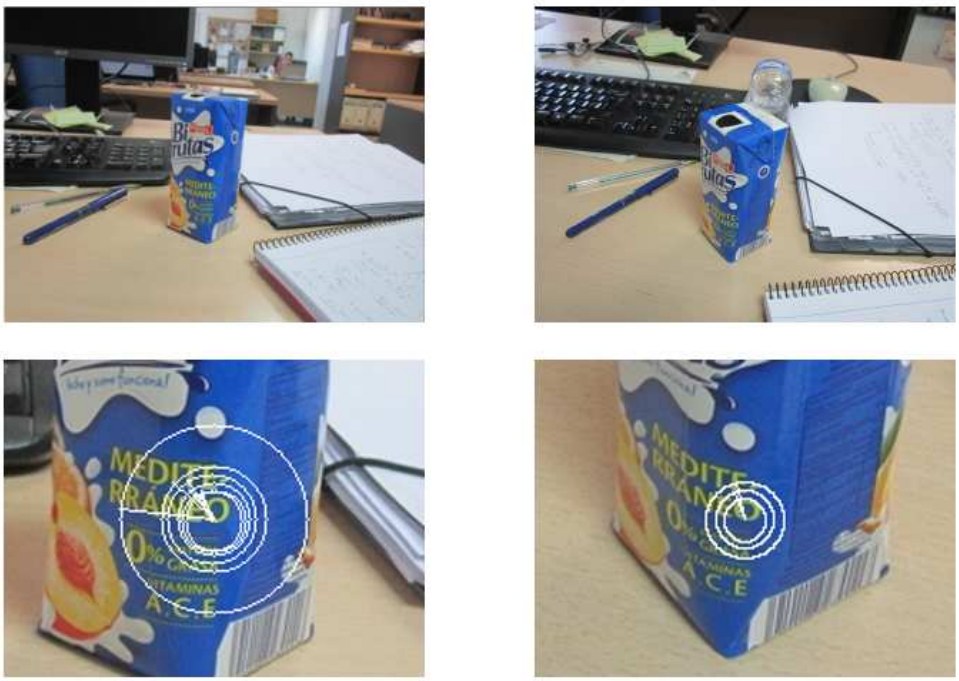


Figure 5.8: The description of a 3D point is obtained from several ORB descriptors when creating the model. Top row: two training images of a juice box. Bottom row: zoom in showing the ORB features obtained at several scales from the same 3D point seen in the two training images. All these descriptors are converted into visual words. Descriptors that result in the same word are merged to disregard redundant information.

The words used for this purpose are given the *term-frequency – inverse document frequency* (*tf-idf*) weight (cf. Section 3.1.3), so that we obtain the bag-of-words vector $\mathbf{v}$ that represents the complete appearance of the surface of the object in a compact manner. This together with the set $\mathcal{P} = \{p_1, p_2, \dots\}$ form the object model $\mathcal{O}$, so we can write

$$\mathcal{O} \equiv \langle \mathbf{v}, \mathcal{P} \rangle. \tag{5.4}$$

This model provides information of all the object surface, so that a single comparison of its bag-of-words vector can yield a similarity measurement independently of the viewpoint and the scale of the object in the query image.

5.4.3 Visual vocabulary object database

We define our object database similarly to the place database in Chapter 4. It is composed of a visual vocabulary, an inverted index and a direct index, as shown in Figure 5.9.

As we introduced in Section 4.2.3, there is a thick boundary problem when discretizing binary spaces that causes a large amount of points to be equidistant to some clusters, so it is not possible to match them with a cluster unambiguously. This difficulty was pointed out by Trzcinski et al. (2012), who measured the loss of matching performance and proposed several structures to overcome this problem. One of them are *parc-trees*, which are a set of random k-means trees

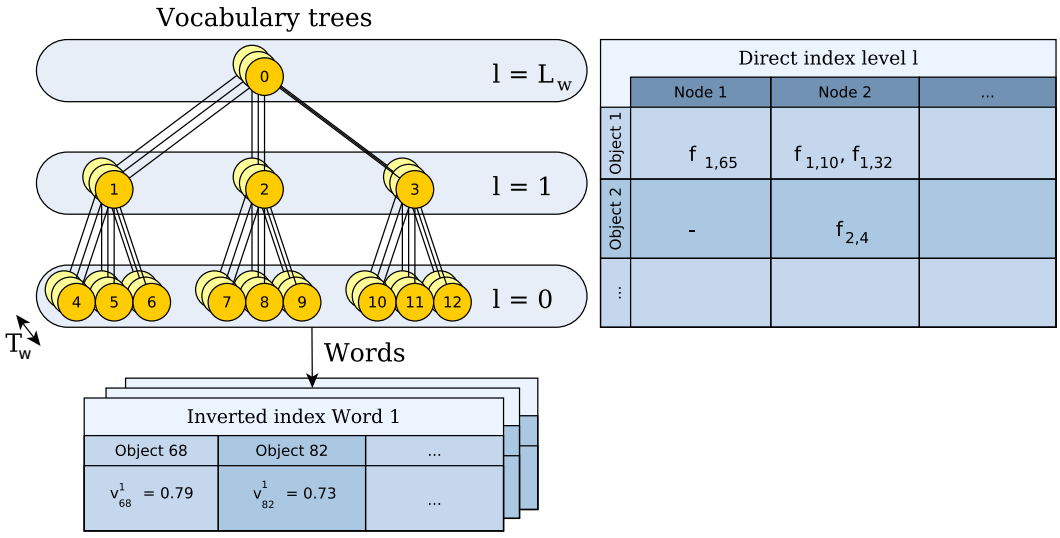


Figure 5.9: Components of the database to represent the whole surface of objects as single bags of binary words.

that produces several hierarchical clusterization of the binary space, so that the discretization error produced by a tree is likely to be mitigated by other. Different types of tree collections are becoming popular to deal with binary matching (Muja & Lowe 2012). Following this idea, instead of relying on a single tree, our vocabulary builds T_w random trees from a set of training ORB features obtained offline, with the k-medians technique. This increases the chances of finding a unique correct cluster for a given point in the descriptor space. The leaves of these trees compose the words of the visual vocabulary. We use 12M descriptors obtained from 30607 independent images from Caltech-256 (Griffin et al. 2007) to build a vocabulary of $T_w = 12$ trees with $k_w = 10$ branches and $L_w = 4$ depth levels, which yields 120K words. When a ORB feature is given, its descriptor vector traverses each tree from the root to the leaves, selecting at each level the node which minimizes the Hamming distance. From the T_w final leaves, the one with lowest distance is selected as word. By concatenating the corresponding words of a set of ORB features, we obtain a vector of words. When all the chosen descriptors $d_1, d_2, \ldots$ of training images are converted into words, these are weighted with the term frequency – inverse document frequency (*tf-idf*) value, and normalized with the L_1-norm. This way, we obtain the bag-of-words vector $\mathbf{v}$ of an object model $\mathcal{O}$.

The inverted index stores for each word in the vocabulary the objects where it is present, along with its weight in those objects. When a query image is given, this structure allows to retrieve the objects that have some word in common with it, so that similarity must be computed only for these models. In addition, the inverted index provides fast access to the common words between the query bag-of-words vector and the model one. The direct index stores for each object model the words (or some other tree node) it contains and its features associated with that word. This is used to access fast to those features which belong to the same word when we have to compute correspondences between them for the geometrical check. As we showed in our previous chapter, we can increase the amount of correspondences if we use the direct index to store nodes at other tree levels (coarser discretization levels), without worsening the execution

time. In this work, for each object and feature, we store in the direct index the node which is closest to the feature descriptor at the first discretization level of the vocabulary trees.

5.4.4 Bag-of-words object recognition process

The bag-of-words recognition approach works in three steps:

1. ORB features are obtained at a single scale from a query image I. These are separated into regions of interest (ROIs) by its 2D location in the image.

2. The object database is queried with each region of interest individually. The inverted index is used to compute a similarity score for all the objects, by using the BC-matching algorithm (Naik et al. 2009).

3. Correspondences between 2D points from the region of interest in the query image and the 3D points of the best-scored object are computed by means of the direct index. These will be used later to verify the recognition of the object and compute its 3D pose.

The process is detailed in the next sections and illustrated by Algorithm 4. It starts by obtaining all the ORB features $\mathcal{F}_I$ from the query image I. The two-stage clustering returns a set of clusters $\mathcal{Q}$ containing subsets of features $\mathcal{F}_q \subseteq \mathcal{F}_I$ that belong to different regions of interest in the query image. Each subset of ORB features is converted into a bag-of-words vector $\mathbf{v}_q$ by traversing the trees of the visual vocabulary V. This vector is compared with all the bag-of-words vectors of the objects in the database and a set of similarity scores $\mathcal{S}$ is obtained. This is done efficiently by accessing directly the data stored by the inverted index $\mathcal{I}$. The object $\mathcal{O}$ with the highest score is retrieved and looked up in the direct index $\mathcal{D}$ created for a certain tree level l. The entry $\mathcal{D}(\mathcal{O})$ is used to compute a set of correspondences $\mathcal{C}$ between the 2D points $\mathcal{F}_q$ of the ROI, and the 3D points $\mathcal{X}^O$ of the object in the object frame O. The correspondences $\mathcal{C}$ are finally used to compute the transformation $\mathbf{T}_C^O$ between the camera and the object with a RANSAC process. The transformation, and hence the recognition, is accepted only if RANSAC successes with a minimum number of supporting correspondences (inliers).

5.4.4.1 Region of interest segmentation

Unlike the sequential view detection, where all the objects are checked for correspondences with the query image, in the bag-of-words recognition approach correspondences are just computed for the best retrieved candidate. This leads cluttered background to hinder the detection of objects which occupy a small part of the image, since spurious similarity scores are produced when querying the database. To ease this difficulty, we propose to cluster the image features by their 2D position and consider each cluster as a region of interest, so that detection can be performed on each individually.

Since the number of regions of interest in the image is not known, we use the nonparametric Medoidshifts technique (Sheikh et al. 2007) to obtain a varying number of clusters. The core of the Medoidshifts algorithm is based on the computation of the distances between each pair of features and on a two $N \times N$ matrix multiplication, being N the number of features. As this operation is $\mathcal{O}(N^{2.38})$ (Coppersmith & Winograd 1987), in practice it can take a long time when N is around a few hundreds. In order to speed up this step, we use a two-stage clustering method. When the number of features is high, first we compute a fixed number of 2D clusters with k-means++ (Arthur & Vassilvitskii 2007). The seeding technique of k-means++ tends to produce 2D clusters which are well distributed around all the image area covered by the features. Then, we perform Medoidshifts using the initial clusters as input points. This produces a varying

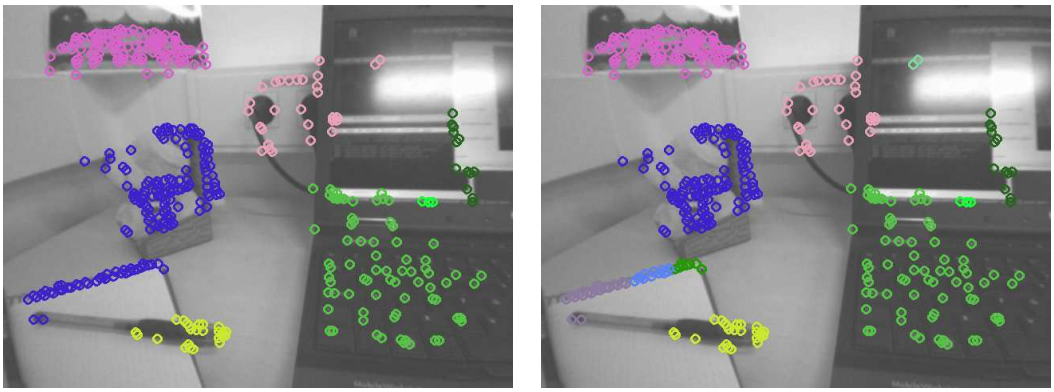
Algorithm 4: Object recognition by bags of words

Input : Query image I, Object database $\mathcal{B} = \{\mathcal{O}_1, \mathcal{O}_2, \dots\}$,
Visual vocabulary V,
Inverted indexes $\mathcal{I}(i)$ for each i-th word,
Direct index $\mathcal{D}_l$ for tree level l, s.t. $\mathcal{D}_l(\mathcal{O}) = \{\langle i_1, \mathcal{F}_{i_1}, \mathcal{X}^O_{i_1}\rangle, \langle i_2, \mathcal{F}_{i_2}, \mathcal{X}^O_{i_2}\rangle, \dots\}$

Output: Objects recognized $\mathcal{R} = \left\{\langle \mathcal{O}_{i_1}, \mathbf{T}_C^{O_{i_1}}\rangle, \langle \mathcal{O}_{i_2}, \mathbf{T}_C^{O_{i_2}}\rangle, \dots\right\}$

$\mathcal{R} \leftarrow \varnothing$;
$\mathcal{F}_I \leftarrow$ features(I);
$\mathcal{Q} \leftarrow$ two_stage_clustering($\mathcal{F}_I$);
foreach $\mathcal{F}_q \in \mathcal{Q}$ **do**
  $\mathbf{v}_q \leftarrow$ conversion($\mathcal{F}_q$, V)
  $\mathcal{S} \leftarrow$ query_BC_matching($\mathbf{v}_q$, $\mathcal{I}$)
  $\mathcal{O} \leftarrow \underset{\mathcal{O}}{\text{argmax}}\ s_{\chi^2}(\mathbf{v}_q, \mathbf{v}_\mathcal{O}) \in \mathcal{S}$
  $\mathcal{C} \leftarrow$ correspondences($\mathcal{D}_l(\mathcal{O})$, $\mathcal{F}_q$, $\mathbf{v}_q$);
  $\langle$inliers, $\mathbf{T}_C^O\rangle \leftarrow$ transformation($\mathcal{F}_q$, $\mathcal{X}^O$, $\mathcal{C}$)
  if there are enough inliers **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{\langle \mathcal{O}, \mathbf{T}_C^O\rangle\}$
  end
end
return $\mathcal{R}$

number of final 2D clusters which fit the distribution of image features better than the initial clusters yielded by k-means++. Image features are associated to their nearest final cluster. This results in a good approximation of the clusters yielded by Medoidshifts, but at a reduced computational effort. Figure 5.10 shows the resulting clusters in an image with 400 features when we create 100 initial clusters with k-means++. Both produce similar results, the only noticeable difference between the techniques are the clusters on the rings of the notebook, on the left-hand



(a) Medoidshifts (b) Two-stage clustering

Figure 5.10: Clusters obtained by Medoidshifts and our two-stage clustering (best viewed in color).

side of the image. Table 5.1 shows the time consumed by each method, measured in a Intel Core i7 @ 2.67GHz machine. We obtain an improvement of one order of magnitude in the execution time.

5.4.4.2 Database query

After separating image features into regions of interest, we transform each image ROI into a bag-of-words vector and query the database to obtain object matching candidates.

In our previous chapter, we used the L_1-norm distance to match images of places when using the whole image as input, since Nister & Stewenius (2006) and our own results showed its reliability. However, a ROI contains just a fraction of the features of the input image, much fewer than the thousand of points comprised by an object model. We ran some initial tests and found out that the L_1-norm distance was not so effective in this case, whereas the Bhattacharyya coefficient or the χ^2 distance obtained better results. This supports the observations by Sivic & Zisserman (2009), who found that the L_1-norm distance performed poorly in their datasets when the bags-of-words vectors to compare had a few words in common only.

A bag-of-words vector encodes an histogram of occurrences of words. This means each histogram bin can be modeled as a Poisson-distributed random variable. This enables the χ^2 distance as a measurement of dissimilarity between bag-of-words vectors (Aherne et al. 1998). However, although this metric is accurate over short statistical distances, it is not well suited for large distances, where the Bhattacharyya coefficient is more convenient (Aherne et al. 1998). Thus, we use a variation of the BC-matching technique proposed by Naik et al. (2009), which makes use of both metrics together. This consists in using a Bhattacharyya-based score (s_B) to obtain the best candidates and selecting from these the one which maximizes a χ^2-based score (s_{χ^2}). Naik et al. (2009) selected a fixed number N of top-ranked Bhattacharyya matches. In our case, we found that a single value of N does not behave well for all the models, so we consider a variable number of Bhattacharyya candidates. We select those whose score is higher than the best Bhattacharyya score obtained multiplied by a factor β.

As well as the L_1-norm score s_{L_1}, the Bhattacharyya and χ^2 scores are very convenient because they can be formulated in terms of the common elements of two bag-of-words vectors, and this information is directly provided by our inverted index. Let $\mathbf{v}$ and $\mathbf{w}$ be two vectors such that $\|\mathbf{v}\|_1 = \|\mathbf{w}\|_1 = 1$, where v^i denotes the i-th element of vector $\mathbf{v}$. Let $\mathcal{V} = \{i \mid v^i \neq 0\}$, $\overline{\mathcal{V}} = \{i \mid v^i = 0\}$ be the sets of indexes of words of $\mathbf{v}$ which are non-empty and empty respectively, and analogously for $\mathcal{W}$ and $\overline{\mathcal{W}}$. The Bhattacharyya score is defined as the Bhattacharyya coefficient:

$$s_B(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V} \cap \mathcal{W}} \sqrt{v^i \; w^i}. \tag{5.5}$$

We expand the χ^2 distance to obtain the χ^2 score s_{χ^2}. The χ^2 distance is defined as

$$\chi^2(\mathbf{v}, \mathbf{w}) = \sum_{v^i + w^i \neq 0} \frac{(v^i - w^i)^2}{v^i + w^i}. \tag{5.6}$$

	Medoidshifts	Two stages
k-means++	-	5.13
Distance computation	0.91	0.04
Medoidshifts	67.76	0.81
Total	70.60	6.24

Table 5.1: Clustering execution time (ms)

Algorithm 5: Query with BC matching

Input : Vector $\mathbf{v}$, Object database $\mathcal{B} = \{\mathcal{O}_1, \mathcal{O}_2, \dots\}$,
Inverted indexes $\mathcal{I}(i)$ for each i-th word
Output: Set of best similarity scores $\mathcal{S} = \{s_{\chi^2}(\mathbf{v}, \mathbf{v}_\mathcal{O}) \mid \mathcal{O} \in \mathcal{B}\}$

$\mathcal{S}_B \leftarrow \{s_B(\mathbf{v}, \mathbf{v}_\mathcal{O}) \leftarrow 0 \mid \mathcal{O} \in \mathcal{B}\}$
$\mathcal{S}_{\chi^2} \leftarrow \{s_{\chi^2}(\mathbf{v}, \mathbf{v}_\mathcal{O}) \leftarrow 0 \mid \mathcal{O} \in \mathcal{B}\}$
foreach word $i \in \mathbf{v}$ **do**
  foreach $\langle \mathcal{O}, v^i_\mathcal{O} \rangle \in \mathcal{I}(i)$ **do**
    $s_B(\mathbf{v}, \mathbf{v}_\mathcal{O}) \leftarrow s_B(\mathbf{v}, \mathbf{v}_\mathcal{O}) + \sqrt{v^i\, v^i_\mathcal{O}}$
    $s_{\chi^2}(\mathbf{v}, \mathbf{v}_\mathcal{O}) \leftarrow s_{\chi^2}(\mathbf{v}, \mathbf{v}_\mathcal{O}) + 2\dfrac{v^i\, v^i_\mathcal{O}}{v^i + v^i_\mathcal{O}}$
  end
end
$\mathcal{S} \leftarrow \{s_{\chi^2}(\mathbf{v}, \mathbf{v}_\mathcal{O}) \in \mathcal{S}_{\chi^2} \mid s_B(\mathbf{v}, \mathbf{v}_\mathcal{O}) \geq \beta \cdot \max(s_B \in \mathcal{S}_B)\}$
return $\mathcal{S}$

If we assume all the elements of the vectors are non-negative (this holds when using the *tf-idf* weight), we can write

$$\chi^2(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V} \cap \overline{\mathcal{W}}} v^i + \sum_{\overline{\mathcal{V}} \cap \mathcal{W}} w^i + \sum_{\mathcal{V} \cap \mathcal{W}} \frac{(v^i - w^i)^2}{v^i + w^i}. \tag{5.7}$$

Given that

$$\sum_{\mathcal{V} \cap \overline{\mathcal{W}}} v^i = \sum_{\mathcal{V}} v^i - \sum_{\mathcal{V} \cap \mathcal{W}} v^i, \tag{5.8}$$

$$\sum_{\overline{\mathcal{V}} \cap \mathcal{W}} w^i = \sum_{\mathcal{W}} w^i - \sum_{\mathcal{V} \cap \mathcal{W}} w^i, \tag{5.9}$$

we get

$$\chi^2(\mathbf{v}, \mathbf{w}) = \sum_{\mathcal{V}} v^i + \sum_{\mathcal{W}} w^i + \sum_{\mathcal{V} \cap \mathcal{W}} \left[\frac{(v^i - w^i)^2}{v^i + w^i} - v^i - w^i \right]. \tag{5.10}$$

Since $\mathbf{v}$ and $\mathbf{w}$ are normalized, we obtain the expression

$$\chi^2(\mathbf{v}, \mathbf{w}) = 2 - 4 \sum_{\mathcal{V} \cap \mathcal{W}} \frac{v^i\, w^i}{v^i + w^i}. \tag{5.11}$$

We finally scale the χ^2 distance to obtain a value in the range $[0, 1]$:

$$s_{\chi^2}(\mathbf{v}, \mathbf{w}) = 1 - \frac{1}{2}\chi^2(\mathbf{v}, \mathbf{w}) = 2 \sum_{\mathcal{V} \cap \mathcal{W}} \frac{v^i\, w^i}{v^i + w^i}. \tag{5.12}$$

Algorithm 5 depicts the process of querying the database applying the BC-matching technique. Given a query bag-of-words vector $\mathbf{v}$, we look up its words in the inverted index $\mathcal{I}$ to get the objects whose appearance vectors $\mathbf{v}_\mathcal{O}$ contain them. These are retrieved together with their word values $v^i_\mathcal{O}$, which allow to compute the Bhattacharyya and the χ^2 scores s_B and s_{χ^2}. The

	Times objects are first candidates (%)		
Metric	Segmented	Erosion 25px	Erosion 50px
Bhattacharyya	57	50	43
χ^2	67	73	73
BC matching $N = 12$	67	77	77
BC matching $\beta = 0.5$	77	77	77

Table 5.2: Accuracy of similarity metrics

objects with a highest Bhattacharyya score are selected, and their corresponding χ^2 scores are returned in descending order.

We ran an experiment to measure the accuracy of each metric when selecting matching candidate models. We selected 30 images from a real webcam-quality video sequence where some objects were shown. To avoid image clustering from influencing the results, we disabled it and masked the background manually. We extracted features and retrieved the nearest model from a database with 20 objects. The accuracy in this experiment is defined as the percentage of times the correct object model is retrieved in first position. The results are shown in Table 5.2. We show results for BC matching with the best N we obtained and the β we selected. To check the effect of noise, we ran the same test after shrinking the background masks with morphological erosions of 25 and 50 pixels, making features from background be used as well. As noticed by Naik et al. (2009), noise in the measurements makes the Bhattacharyya score lose accuracy, while the χ^2 one behaves better. The combination of the two metrics together outperforms the results of them alone. Using β instead of N yields slightly better results.

5.4.4.3 Correspondence computation

The candidate object of each image cluster is finally checked for geometrical consistency to achieve its 3D pose, or to discard the cluster if no transformation is obtained. This process, described in depth in Section 5.5, requires correspondences between features in the query image and 3D points in the object model.

In this approach, corresponding descriptors are required to belong to the same vocabulary tree node and to satisfy the nearest neighbor ratio condition. This way, we take advantage of our vocabulary trees as a means to divide features into groups which are expected to be close in the descriptor space. This is efficiently done with our direct index. When an object is added to the database, one of the nodes its descriptors traverse is stored in the direct index. We select the closest node in the first discretization level $l = 1$ of all the trees. The same is done when the image features are converted into words, so that we can select features with share a common node.

Although this can reduce the amount of correspondences obtained, as we shown in Section 4.4.5, the overall number of comparisons between descriptors is lowered, dropping the execution time remarkably. We show in Table 5.3 the average execution time of computing correspondences with our direct index (DI) and an exhaustive method, where all the distances between the image

Method	Execution time (ms/cluster)
Exhaustive	105.2
DI	1.9

Table 5.3: Computation of correspondences

cluster and the model features are computed. These data were obtained by comparing 30 image clusters with some object models consisting in 700–17000 words.

5.5 3D Pose computation from corresponding points

The object retrieval algorithms presented in the previous sections provide a set of correspondences between the features of an image and the 3D points of an object candidate. The recognition finally successes if the matched points are geometrically consistent, allowing us to compute the 3D pose of the detected object; otherwise, the candidate is rejected. We make use of two standard methods to fulfill this: we impose a homography-based restriction to planar objects and apply a perspective-n-point algorithm to those of arbitrary shape. For the reader's convenience, we detail both techniques below.

5.5.1 Homography for planar objects

A homography is a 3×3 non-singular homogeneous matrix that represents a projective transformation between two views (Hartley & Zisserman 2004). Given 3 collinear points in a view, a projective transformation is that which preserves their collinearity in the other view. This transformation is characterized for resulting from a rotating camera or for being induced by a 3D plane that is seen by two cameras, as illustrated by Figure 5.11. The intersection between the ray that goes from the optical center of a camera C and a 3D point $\mathbf{x}_\pi$ in a plane Π produces a 2D point in the image plane of the camera $\mathbf{x}$. The projection of the $\mathbf{x}_\pi$ in the image plane of a second camera C', produces the 2D point $\mathbf{x}'$. Then, the homography $\mathbf{H}$ that is induced by the plane Π results in a direct mapping between the corresponding projections $\mathbf{x}$, $\mathbf{x}'$ of points that lie in the plane. With points in homogeneous coordinates, $\mathbf{x} = \lambda(x, y, 1)^\top$, $\mathbf{x}' = \lambda'(x', y', 1)^\top$, we can write

$$\mathbf{x}' = \mathbf{H}\mathbf{x}. \tag{5.13}$$

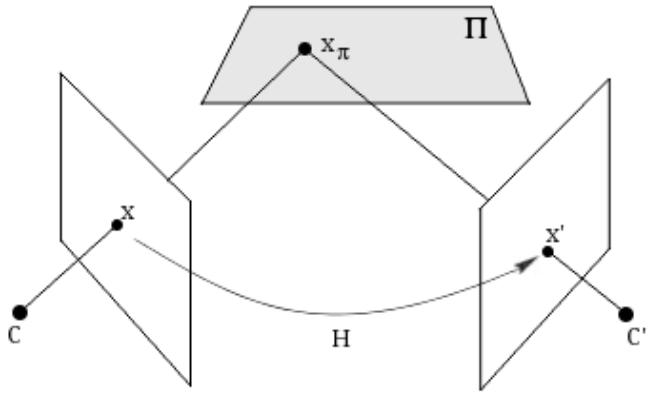


Figure 5.11: Homography induced by a plane. The ray corresponding to a point $\mathbf{x}$ in the image plane of camera C is extended to meet the plane Π in the 3D point $\mathbf{x}_\pi$. The projection of $\mathbf{x}_\pi$ in the image plane of camera C' yields point $\mathbf{x}'$. The map from $\mathbf{x}$ to $\mathbf{x}'$ is the homography $\mathbf{H}$ induced by the plane Π. Credit: Hartley & Zisserman (2004).

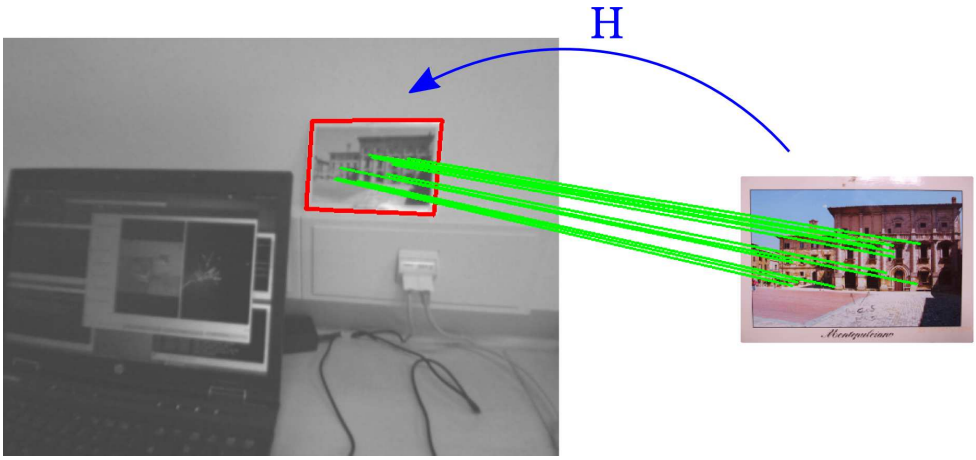


Figure 5.12: A homography between the view of a planar object model and a query image is computed from a set of corresponding points selected with RANSAC. The recognition is accepted if enough inliers are obtained.

Although the homography matrix contains 9 elements, it has 8 degrees of freedom because it is defined up to scale. This means that matrix $\mathbf{H}$ multiplied by any non-zero scalar value still provides the same 2D mapping between points $\mathbf{x}$ and $\mathbf{x}'$.

Homography as geometrical verification. Homographies present a very practical application in the object recognition pipeline when dealing with objects composed of planar surfaces. Regarding planar objects modeled as independent views $V = \langle \mathcal{F}_V, \mathcal{X}_V^O, \mathcal{K}_V \rangle$ (cf. Section 5.3.1), we can consider the object present in a query image as the result of applying a projective transformation to one of the training images that defined the object. Thus, there must exist a homography that relates both views. If we are able to find a homography in such a case, we accept the recognition of the planar object; otherwise, we reject the candidate detection.

Consider the example given in Figure 5.12. This depicts a real scene in which a postcard is attached to the wall in a desktop area. The postcard is modeled as a planar object with a single view, with the training image shown on the right hand side. The homography $\mathbf{H}$ describes the transformation from the view to the scene, locating the 2D area of the view in the query image. We can estimate the homography from 4 pairs of corresponding 2D points between the features $\mathcal{F}_I$ of the query image and those of the model view $\mathcal{F}_V$. If there are not 3 collinear points, the solution is unique. These are obtained by running RANSAC over the set of initial correspondences $\mathcal{C}$ computed during the view matching. In each RANSAC iteration, the direct linear transformation (DLT) algorithm (Hartley & Zisserman 2004) is used with 4 random points to compute a candidate homography matrix $\widetilde{\mathbf{H}}$. Then, the reprojection error of each correspondence is computed and those below a threshold ε_r are considered inliers. The number of inliers is given by

$$\widetilde{n} = \left| \{ \langle f_i,\, f'_j \rangle \in \mathcal{C} \mid \| \mathbf{x}'^{*}_j - \left(\widetilde{\mathbf{H}} \mathbf{x}_i \right)^{*} \| \leq \varepsilon_r \} \right|, \tag{5.14}$$

being * the normalization operator. If we are able to find a homography that is supported by a reliable enough number of inlier correspondences, the recognition of the candidate object is accepted.

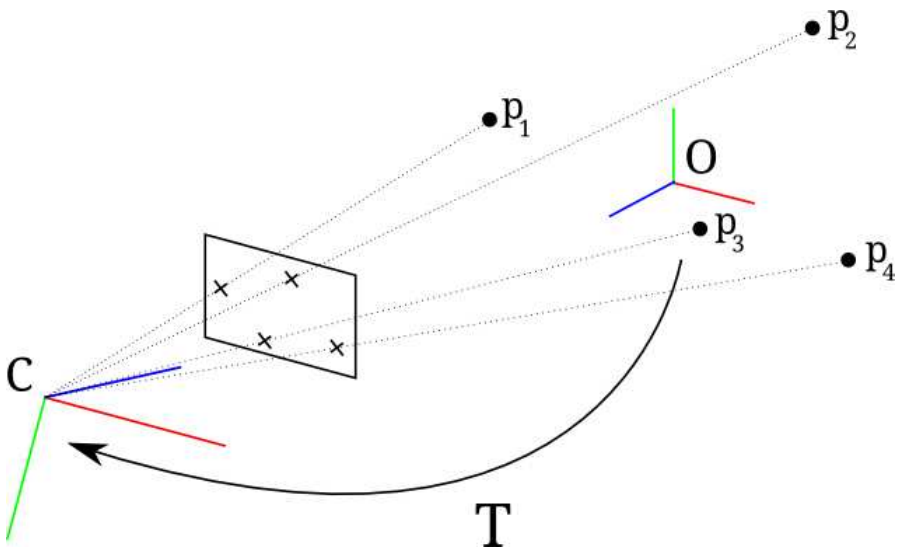


Figure 5.13: The perspective-n-point problem consist in finding the pose of a camera C in the frame of reference O given four 3D points and their projections in the image plane.

3D pose recovery. After a successful recognition, a final homography matrix $\mathbf{H}$ is optimized by using all the inliers returned by RANSAC. This homography encodes the motion between the cameras that acquired the query image and the object training image (Hartley & Zisserman 2004). We can write

$$\mathbf{H} = \mathbf{R} + \frac{1}{d}\mathbf{t}\mathbf{n}^{\top}, \tag{5.15}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{t} \in \mathbb{R}^3$ are the rotation and translation that define the spatial transformation between the cameras, and $\mathbf{n} = (n_x, n_y, n_z)^{\top}$ and d the normal vector and the distance to origin of the plane $\Pi \equiv n_x x + n_y y + n_z z + d = 0$ that induces the homography $\mathbf{H}$. It is possible to extract the plane and the motion between the cameras from the homography matrix by using a numerical method based on its the singular value decomposition (Faugeras & Lustman 1988, Zhang & Hanson 1996), or analytically (Vargas & Malis 2005). However, as a consequence of the scale ambiguity, neither the distance coordinate d of the plane nor the translation between cameras $\mathbf{t}$ can be extracted individually, but the relation $\mathbf{t}/d$ is obtained instead. We can solve this issue and find $\mathbf{t}$ by incorporating the real distance between two observed 3D points, available since we provide the dimensions of the planar views when creating the planar object model.

From $\mathbf{H}$ we can obtain the transformation $\mathbf{T}_C^T$ that relates the query camera C with the camera T that created the training image of the recognized object view. To find the pose of the object $\mathbf{T}_C^O$ we just concatenate $\mathbf{T}_C^T$ with the transformation $\mathbf{T}_T^V$ between the training camera and the object view, and the transformation $\mathbf{T}_V^O$ between the view and the object frame, both of them computed when creating the object model:

$$\mathbf{T}_C^O = \mathbf{T}_C^T \cdot \mathbf{T}_T^V \cdot \mathbf{T}_V^O. \tag{5.16}$$

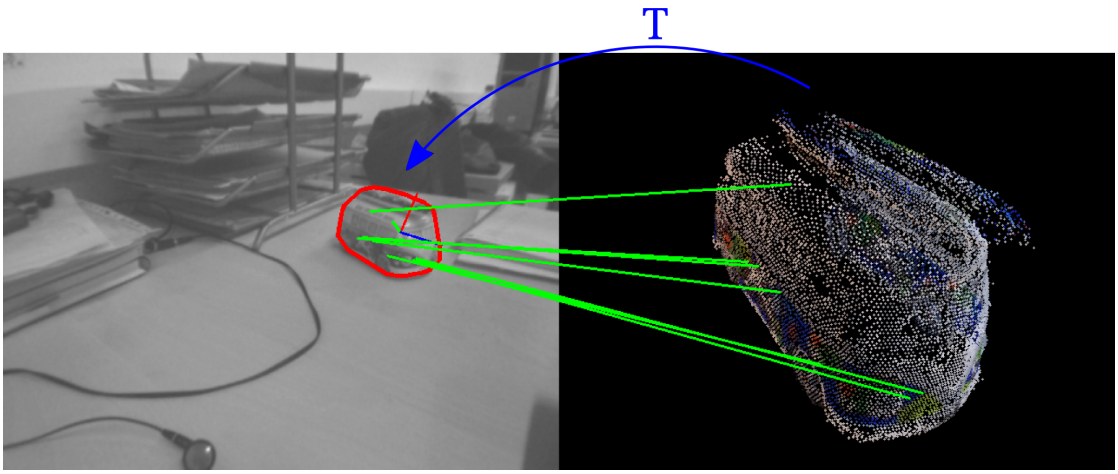


Figure 5.14: A transformation between a 3D model and a query image is computed with EPnP from a set of 2D-3D correspondences selected with RANSAC. The recognition is accepted if enough inliers are obtained.

5.5.2 General perspective-n-point problem

In the general case, after computing correspondences between a query image and an object model, composed of either independent views or a single bag-of-words vector, we obtain an association between 2D and 3D points in the object frame, as illustrated in Figure 5.13. The problem of finding the pose of the camera in the same frame of reference, which is equivalent to find the pose of the object with respect to the camera, is named perspective-n-point (PnP) problem. This can be solved with a unique solution when at least $n = 4$ points are provided; $n = 3$ points without restrictions result in multiple configurations, and fewer points, in infinite solutions (Fischler & Bolles 1981). This problem is usually solved in two steps. The length of the projection rays from the camera to the 3D points is firstly estimated, so that the coordinates of the points in the camera frame are obtained. This produces a direct map between two sets of 3D points expressed in different frames whose transformation is finally obtained by solving the absolute orientation problem (Horn et al. 1988). There exist several algorithms to estimate the depth of the 3D points in the first step. Lepetit et al. (2009) proposed EPnP, a non-iterative algorithm that finds the depths and the camera pose from 3 coplanar points or 4 general points. Their algorithm provides an accurate solution, and scales well for higher amounts of points.

PnP as geometrical verification. The standard approach to find the pose of an object is to compute a set of putative correspondences between the query image and the model, select a subset of inlier pairs with a robust technique and solve the perspective-n-point problem. In our approach, we use EPnP along with RANSAC to find the pose of the object in the camera frame $\mathbf{T}_C^O$ from a set of putative correspondences $\mathcal{C}$ obtained from view matching or by the direct index. In each RANSAC iteration, a candidate $\widetilde{\mathbf{T}}_C^O$ is computed from four randomly selected pairs. The reprojection error is then calculated for every pair and those which are below a threshold are considered inliers. A 3D point in the object frame, denoted $\mathbf{X}^O = (x, y, z, 1)^\top$ is projected to the 2D point $\mathbf{x}$ by a camera with calibration matrix $\mathbf{K}$ and a candidate $\widetilde{\mathbf{T}}_C^O$ by

$$\mathbf{x} = (\mathbf{K} \,|\, \mathbf{0}) \left(\widetilde{\mathbf{T}}_C^O\right)^{-1} \mathbf{X}^O. \tag{5.17}$$

The number of inliers of a candidate transformation is given by

$$\widetilde{n} = \left| \{ \langle f_i,\, f'_j \rangle \in \mathcal{C} \mid \| \mathbf{x}'^{*}_j - \left((\mathbf{K} \,|\, \mathbf{0}) \left(\widetilde{\mathbf{T}}^O_C \right)^{-1} \mathbf{X}^O_i \right)^{*} \| \leq \varepsilon_r \} \right|, \tag{5.18}$$

being ε_r the reprojection error threshold and * the normalization operator. If a transformation is supported by enough inliers, the recognition of the candidate object is accepted.

3D pose recovery. After a successful recognition, an initial pose of the object $\widetilde{\mathbf{T}}^O_C$ is obtained, created from 4 corresponding points. The final 3D pose of the object $\mathbf{T}^O_C$ is optimized by applying EPnP on all the inliers returned by RANSAC. Figure 5.14 shows a successful recognition, where the transformation obtained is used to project the 3D model in the query image, whose convex hull is drawn.

5.6 EKF monoSLAM augmented with objects

Once the object recognition algorithms have been defined, we can explain how they can be used together with a monocular SLAM system to be inserted into SLAM maps. Our proposal builds on a state-of-the-art monocular SLAM formulated in an extended Kalman filter (EKF) basis (Durrant-Whyte & Bailey 2006), enhanced to support the addition of objects in the map.

We follow the 1-point RANSAC EKF proposed by Civera et al. (2010). The state vector $\mathbf{x}$ includes the camera motion parameters (position, orientation and linear and angular velocities) at step k and the n map features, based on FAST key points matched by normalized cross-correlation. Map features are initialized using inverse depth parametrization (Civera et al. 2008), converted later into 3D Euclidean coordinates in a universal frame of reference.

Initially, we used this SLAM algorithm with the baseline recognition approach based on independent SURF views. The complete system is divided into two threads: one dedicated to monocular SLAM and other one to object recognition. Figure 5.15 gives a general overview of the algorithm using images from our experimental results. Let start at step $k - m$. The image I_{k-m} is used both in the monocular SLAM and the recognition threads. The monocular SLAM thread uses this image to update the state vector $\mathbf{x}$ from the previous step $k - m - 1$ to the current one $k - m$ using a standard EKF formulation. At the same time, the SLAM state vector is augmented with the current camera pose $\mathbf{T}^{C_{k-m}}_W$ in the world frame W. Such augmentation is necessary for a coherent object insertion every time the recognition thread starts, because the recognition results will arrive after some processing at step k, but the object insertion should be made with respect to the input image I_{k-m}. When the recognition process finishes for a given image, the past camera pose is marginalized out from the state vector.

After a successful recognition m steps later, the transformation from the past camera to the object $\mathbf{T}^O_{C_{k-m}}$ is obtained. A few points of those used to compute the transformation between image I_{k-m} and the object model are selected, in such a way that their projections in I_{k-m} are sufficiently spread. The 3D coordinates of these points $\mathbf{X}^O$ in the object frame are also returned by the recognition thread. The insertion of the object in the map takes place by initializing features of I_{k-m} with the given 3D coordinates. Each point has to be referred to the SLAM reference frame W for its insertion, which is done as follows:

$$\mathbf{X}^W = \left(\mathbf{T}^{C_{k-m}}_W \right)^{-1} \cdot \left(\mathbf{T}^O_{C_{k-m}} \right)^{-1} \cdot \mathbf{X}^O, \tag{5.19}$$

where $\mathbf{T}^{C_{k-m}}_W$ is the position and orientation of the SLAM camera when the object recognition was initiated, stored until $\mathbf{x}_{k-1}$. The obtained 3D points $\mathbf{X}^W$ are added to the state vector

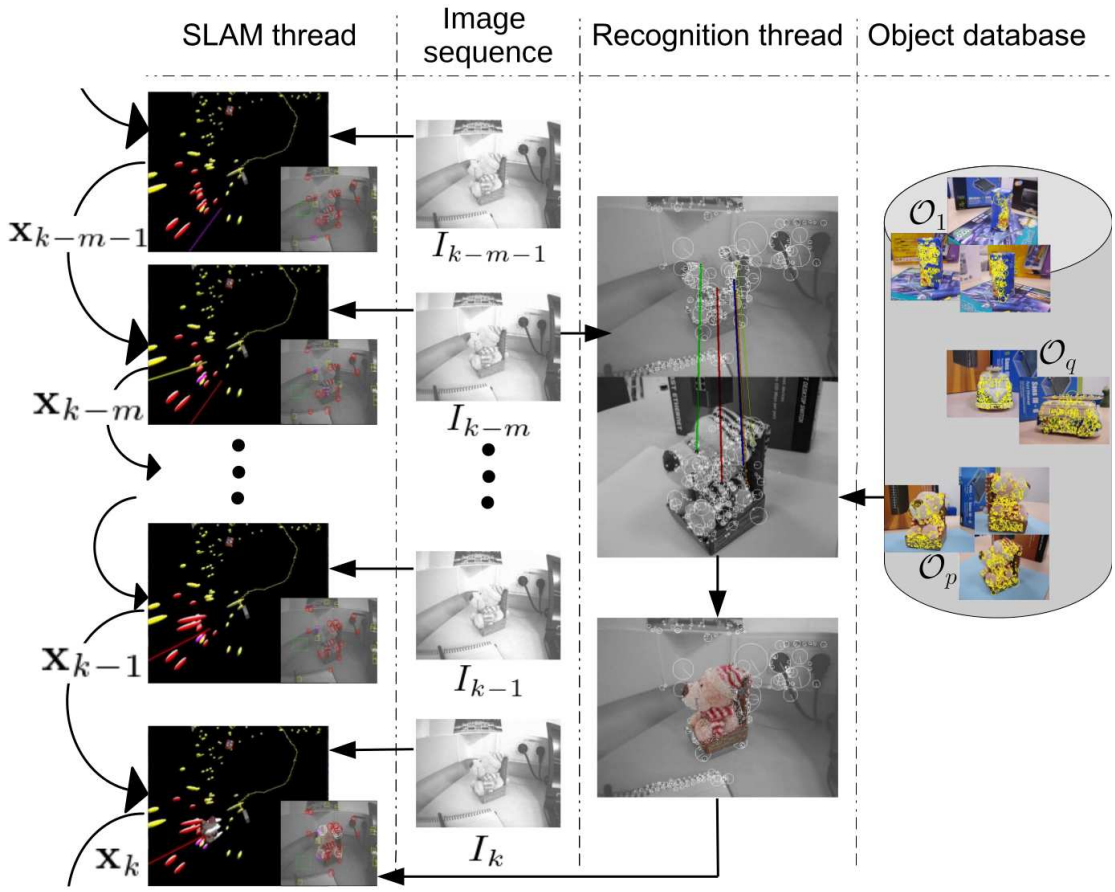


Figure 5.15: Overview of the monoSLAM algorithm fused with recognition of 3D objects (best seen in color).

$\mathbf{x}$ from step k and labeled as objects. They are then tracked as other map points, and hence its position is refined by the standard EKF monocular SLAM formulation (Civera et al. 2008), refining as well the pose of the object in the SLAM map.

5.7 Experimental evaluation

We tested our two object recognition approaches in two real video sequences. Firstly, we tested the algorithm based on independent views along with our proposed monocular SLAM system. The experiments, initially presented in Civera et al. (2011), show results in a desktop environment and in a hospital room, and were carried out under the framework of the European project RoboEarth (Waibel et al. 2011). Then, we used the bag-of-words method to show the performance improvement of the recognition process obtained in the desktop scenario.

5.7.1 Independent view recognition for monoSLAM

We tested the independent view approach working together with the monocular SLAM algorithm by building a map in two different indoor environments. The two experiments presented here were recorded with the same camera, a low-cost black-and-white Unibrain camera with a resolution of 320×240 pixels. The model of the objects were built from images taken with a standard consumer digital camera. The image sequences for both experiments were gathered by the Unibrain camera at 30 frames per second and used at this frequency as the input to a monoSLAM algorithm. As the computational cost of the proposed algorithm is higher than 33 milliseconds for the map sizes used, some of the frames may be skipped by the visual SLAM.

5.7.1.1 Hospital room environment

Before going into detail about the performance of the object recognition in the visual SLAM system, the general aim of this experiment should be stated. This experiment is framed into the RoboEarth project (Waibel et al. 2011); dedicated to construct a giant network and database repository for robots to share knowledge by uploading and downloading actions and sensory data. In this specific experiment a robot provided with an arm enters a hospital room and downloads from the RoboEarth database 1) the recognition models for the objects it may encounter in the hospital room, and 2) an action recipe consisting in the task of serving a tetra brik of juice to a patient in the bed. The object recognition algorithm and the visual SLAM system proposed here, using the downloaded recognition models, estimated the partially annotated map that allowed the robot to successfully grasp the brik and serve it to a patient.

The object models considered in this experiment were the tetra brik of juice, the cabinet where it was located and the bed, depicted in Figure 5.16. The three of them were modeled as box-shaped objects composed of planar surfaces, in a per-view basis and described by SURF features. The homography geometrical check was used for verification, requiring at least 6 corresponding points and computing the pose with EPnP. The cabinet and the bed models were created just with 3 and 1 views due to the occlusion or the lack of texture of the rest of their surface. For visualization purposes, they were depicted as colored prisms in the visual SLAM application. As an example, we show in Figure 5.17 the planar model of the brik.

The sequence for this experiment has 6003 frames. Figure 5.18 shows several frames extracted from the SLAM experiment. Figure 5.18(a) (top) shows the estimation at frame #34, where the cabinet has been already recognized and inserted in the map. Notice the accuracy of the insertion by the overlap between the model reprojection and the real cabinet. Features are color-coded: white stands for points successfully tracked in recognized objects, red for the map points

(a) Cabinet (b) Bed (c) Tetra brik

Figure 5.16: Cabinet, bed and tetra brik present in the RoboEarth experiment.

Figure 5.17: Planar model for the tetra brik object.

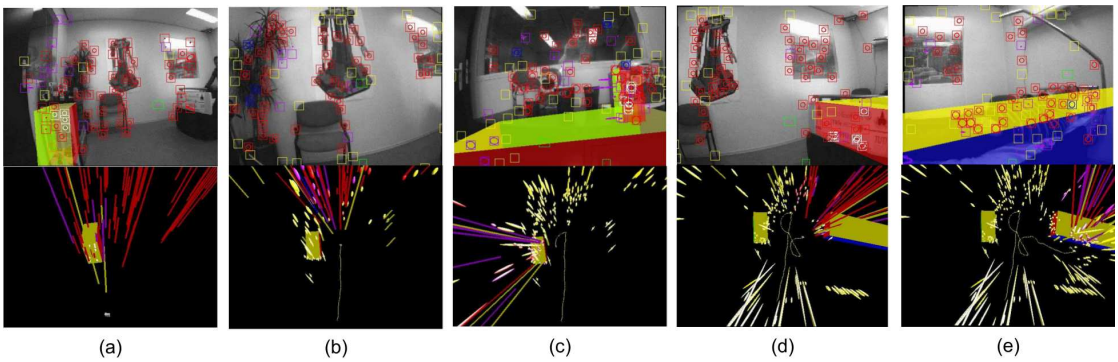


Figure 5.18: Representative images (at the top) and 3D estimation at their respective times (at the bottom) for the hospital room experiment. The tracked features are displayed in the images as circles. The inserted objects, represented as colored prismatic solids, are also reprojected at the images. The 3D views show the camera trajectory up to this frame as a yellow line, the point feature uncertainties as ellipses and the inserted objects. (a), frame #34, the cabinet has been already recognized and inserted. (b), frame #241, the robot goes forward. Notice that, although the cabinet is not seen in the image, its 3D position remains registered. (c), frame #912, the robot turns left and faces the cabinet and the object bio, which is detected and registered. (d), the robot turns, recognizing and registering the bed. (e) Robot location at the end of the experiment.

successfully tracked, and blue and magenta for rejected points. In the bottom row it can be seen a top view of the 3D map: the ellipses stand for the uncertainty regions of the salient point features and the colored prism is the cabinet model. The other images in Figure 5.18 stand for other frames of the sequence temporally ordered. Notice that in Figure 5.18(c) the tetra brik was already recognized and inserted. In Figure 5.18(d) the bed has also been recognized and registered. Finally, Figure 5.18(e) shows the final frame of the sequence along with the final estimation results For a better visualization, Figure 5.19 shows a detail of the final map estimation and the camera trajectory. Notice the accurate registration of the tetra brik over the cabinet in Figure 5.19(b). The complete RoboEarth experiment can be watched in Video 2.a., listed in Section 1.4.2.

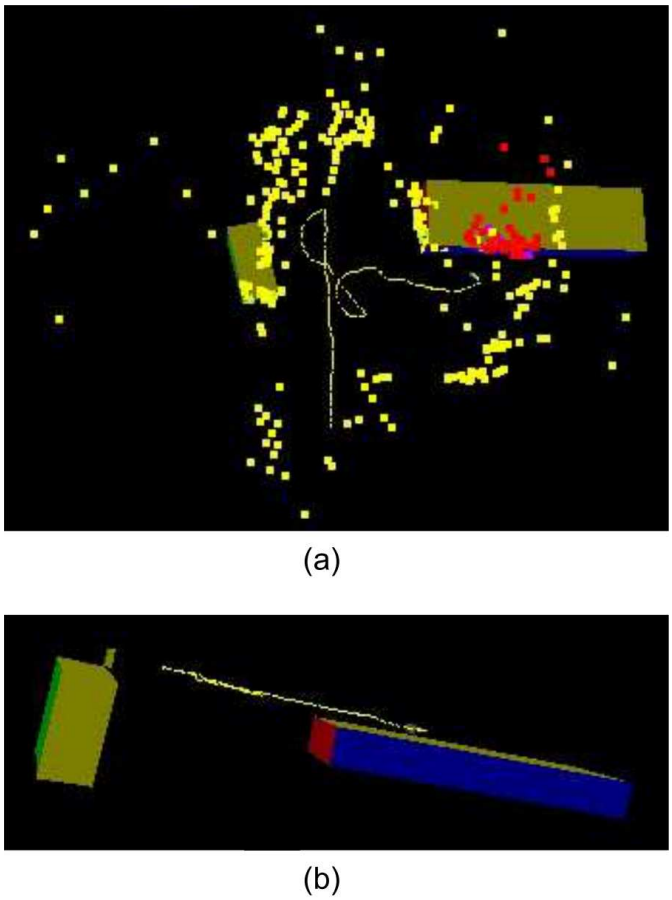


Figure 5.19: SLAM results at the end of the hospital room experiment. (a), top-view of the camera trajectory, estimated salient point features and recognized objects. (b), side-view of the camera trajectory and recognized objects: the tetra pack over the cabinet, both at the left; and the bed at the right.

5.7.1.2 Desktop environment

The sequence for this experiment had 8951 frames and was recorded moving a hand held camera over a desktop in one of our laboratories for about 5 minutes.

We built 6 models with SURF features for the objects shown in Figure 5.20. The selected objects are a toy van, a toy lion, a tetra brik (named *bio*), a chewing gum box (*orbit*) and two postcards (*card1*, *card2*). We modeled the postcards as planar objects with a single image, imposing the actual size of the objects to compute the 3D coordinates. The geometry of the rest of objects were modeled as generic 3D point clouds, constructed from several training images taken around each object (5, 14, 15 and 20, respectively). The object bio is the same brik used in the previous experiment. Unlike then, we used here a generic 3D model instead of a planar one. Since the brik is not completely planar, the generic structure-from-motion technique is able to capture more accurate 3D information when creating the point cloud. The same applies to the object orbit. The homography geometrical check was used for the verification of the

Figure 5.20: Evaluation objects

cards, and the generic perspective-n-point algorithm for the rest of objects, requiring at least 6 corresponding points.

The lighting conditions were particularly bad in this sequence, as can be observed in Figure 5.22. In spite of this fact, all the six objects that composed our database were detected along the sequence (with no false positives), inserted in the 3D map and tracked the rest of the sequence. Only the first object recognition was considered. Figure 5.21 shows the results for the recognition thread at the specific frames where the six objects were recognized. The top row shows the frame in the sequence where they were recognized; the middle row the object view that was recognized; and the colored lines stand for the correspondences between the two. In the bottom row it is displayed the reprojection of the dense point cloud model over the top row images.

Figure 5.22 summarizes the results of this experiment for several steps of the estimation. For each step it is shown the current frame and a 3D view of the estimation. The 3D view show the uncertainty ellipses for each point in the 3D map, the dense point clouds modeling the objects and the camera trajectory as a yellow line. Figure 5.22(a) shows the estimation results at step #610, when no object has been inserted yet. Figure 5.22(b) shows frame #1359, just after bio is detected and inserted. Figures 5.22(c), 5.22(d) and 5.22(e) show respectively that van, orbit and card1 have been inserted in the map and are being tracked. Their correspondent frames in the sequence are $\#2764, \#4062, \#4725$. Figure 5.22(f) shows the results at step #7102, when the two latest objects –lion and card2– have been inserted. In Figure 5.22(g) the camera has gone back to the starting point (frame #8538), imaging again objects bio and van. Finally, Figure 5.22(h) is the last frame of the sequence, showing only the objects in the SLAM map. The complete desktop experiment can be watched in Video 2.b., listed in Section 1.4.2.

5.7.2 Bag-of-words recognition

To evaluate the bag-of-words object recognition, we applied this approach on the same desktop sequence used above. We focus our results on the recognition performance, so we do without the

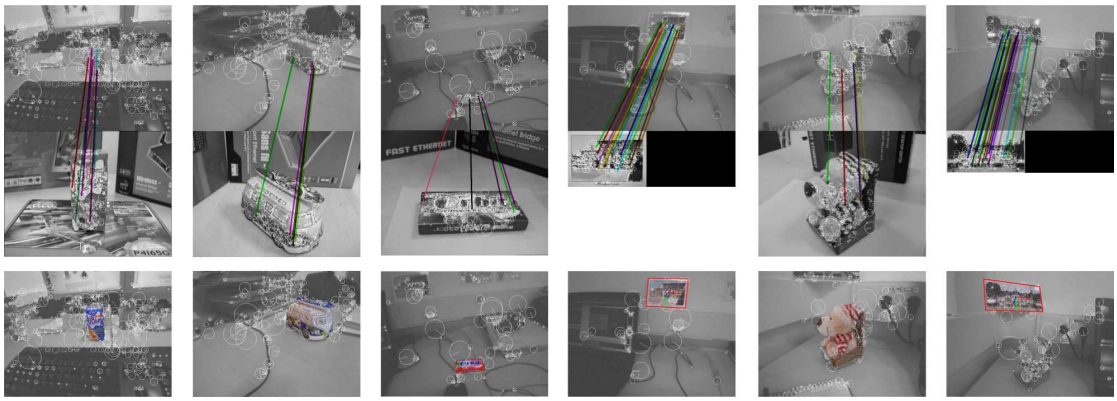


Figure 5.21: Object recognition thread results, showing column-wise the specific frames where the six objects were detected. The top row shows the image in the monocular sequence input to the monocular SLAM, the middle row shows the view of the object model, and the colored lines are the matches. The object is inserted in the map based on those correspondences. The bottom row shows the dense point cloud of the object over the image, proving the correct alignment.

visual SLAM system.

We extracted 12M ORB descriptors from 30607 independent images from Caltech-256 (Griffin et al. 2007) to build a vocabulary of $T = 12$ trees with $k = 10$ branches and $L = 4$ depth levels, which yields 120K words. Apart from the 6 objects used before, shown in Figure 5.20, we inserted 14 more in the database, adding up to 20 objects in total for this experiment. The object models were created with a varying number of training images (from 11 to 25 for general 3D objects, 1 for two planar ones), obtained from different consumer cameras. These images were 640×480 px and showed the object occupying most of the image area. Ten of these objects were obtained from the dataset provided by Hsiao et al. (2010). All of them are objects which can be usually found in real scenarios (boxes, cans, mugs, toys). The produced models have a very different number of words, from 743 to 35077, depending on how rich their texture is and how much surface is covered by the training images. Only the six objects said above are present in the video sequence, while the rest of them were used as distractors. Since no SLAM is run in this experiment, we processed the video sequence at 15Hz. The results of this experiment can be watched in Video 2.c., listed in Section 1.4.2.

Method	Max. iterations	Successful cases	Execution time (ms/cluster)
Exhaustive	2000	20	51.6
DI	2000	19	77.7
DI	1000	18	40.2
DI	100	17	4.3
DI	50	11	2.3
DI	30	7	0.8

Table 5.4: RANSAC and EPnP

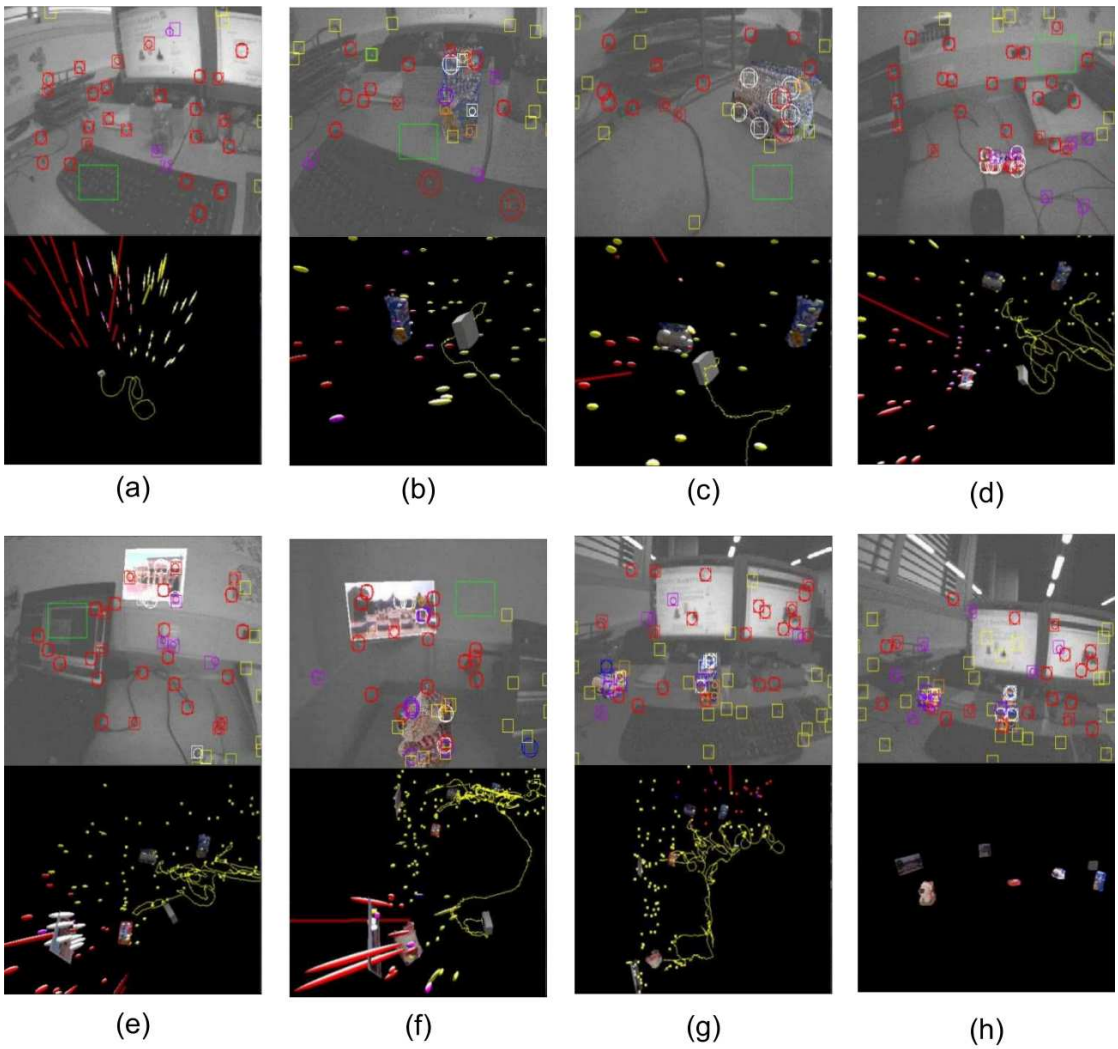


Figure 5.22: Representative images (at the top) and 3D estimation at their respective times (at the bottom) for the desktop experiment. (a) Initial map, still with no recognized objects. (b) Bio has been recognized, inserted and is being tracked. (c), (d) and (e): van, orbit and card1 have been inserted. (g) The two remaining objects –card2 and lion– are inserted. (h) The camera moves back close to the starting position, revisiting all previous objects. (h) Final frame of the sequence and 3D view of the objects registered in a common reference frame.

5.7.2.1 Geometrical verification performance

After querying the database and obtaining the best object model that matches an image region of interest, we compute the 2D-3D correspondences to verify the recognition with RANSAC and EPnP . RANSAC computes iteratively candidate spatial transformations from random subsets of the available correspondences and selects the one which maximizes the number of consistent points. If no transformation with at least 6 inliers is found, we reject the detection. The execution time of RANSAC (and its effectiveness) can be reduced by limiting the number of iterations. We show in Table 5.4 the effect of limiting this parameter. We ran RANSAC and EPnP on 147 clusters obtained from 30 images. From 30 clusters containing objects, the database query returned the correct object as the first candidate in 22 cases, which is the maximum number of successful cases that could be obtained in Table 5.4. We show also the results when computing correspondences with the exhaustive method and running RANSAC with a high number of iterations. We consider this is the best result we can obtain in these images with ORB descriptors. Note that computing correspondences exhaustively takes around 105ms per cluster, as we shown in Table 5.3 in page 101, whereas the direct index reduces this to 2ms per cluster. We see in Table 5.4 that if we use the direct index instead, we miss one detection only (from 20 to 19), but halve the total average time per image cluster when taking into account the time of computing correspondences. We set the number of RANSAC iterations to 100 for the rest of experiments because it offers a balanced trade-off between detection rate and execution time.

5.7.2.2 Detection performance

We measured the detection rate of our system. No false detections were fired all along the sequence. This means that those cases where a wrong object was returned after querying the database were successfully rejected in the geometrical check stage. Figure 5.23 shows two examples of successful recognitions of bio and the van. Detections are shown by projecting the points of the model in the image.

Table 5.5 shows the times each object is recognized and the number of frames in which it is visible. Note that the presence of an object in a frame was decided by manual inspection of the video sequence, so that although it may be easy for a person to say whether an object is present in a frame, it may be a challenge for many computer vision algorithms. All the objects are correctly recognized several times, yielding a total average recall of 26%. We can see there are detections missing in some frames, specially in the case of objects lion and van. These two are particularly difficult to recognize because they present a lot of similar looking patterns, such as the striped white and red scarf and cap of the lion, and the similar picture which appears on top and on both sides of the van. This produces perceptual aliasing, making correspondences fail the neighbor ratio condition and be rejected. Most of the rest of missed cases are due to the

Object	Detections	Occurrences	Recall (%)
Bio	251	471	53.3
Van	26	262	9.9
Orbit	193	464	41.6
Lion	2	483	0.4
Card1	183	681	26.9
Card2	141	590	23.9
Average			26.0

Table 5.5: Detection performance

Figure 5.23: Examples of correct recognitions of bio and the van

(a) Detection missed (b) Incorrect transformation

Figure 5.24: Examples of unsuccessful detections. In (a) the van and bio are not detected because of their small size in the image. In (b) an incorrect transformation is obtained because ill-conditioned point correspondences.

small size of the object in the query image. These are challenging cases. Fewer features can be extracted on the object, which are more likely to be clustered together with background features. This may make database queries fail when retrieving the candidate model. This can happen as well if the scale change is bigger than the 2 octaves we considered when creating the models, since features will not be converted into the same word than the corresponding feature in the model. Figure 5.24(a) shows an example where the bio carton and the van were present but were not detected. As a future work, our system is due to be compared with the state-of-the-art approach to show the effects of the described difficulties.

Although we obtained no false positives, in some cases the computed transformations were slightly inaccurate. The error in computing the transformation can be due to ill-conditioned point configurations for PnP , which makes it produce a imprecise transformation. Figure 5.24(b) shows an example of this case. RANSAC can also obtain an incorrect transformation if some correspondence produced by the background is considered as inlier. This could be ease by increasing the number of inliers to accept the transformation. In the general case, the features extracted from consecutive frames are not the same. This leads to obtain correspondences

Figure 5.25: Example of different consecutive detections of bio

	Per image		Per cluster	
	Median	Max.	Median	Max.
ORB	4.0	10.5	-	-
Clustering	2.9	10.4	-	-
Conversion	11.9	26.1	1.0	16.5
Query	0.3	1.9	0.0	1.6
DI	6.4	19.4	1.1	10.8
RANSAC	2.0	9.6	1.9	6.1
Total	27.9	60.8	8.0	27.0

Table 5.6: System execution time (ms)

between different sets of model and image features which can produce different transformations. The effect of this is that the pose of an object found in consecutive frames varies, as in the example shown in Figure 5.25. This behavior does not entail a great drawback because this effect is mitigated by the SLAM algorithm which robustly tracks the features which belong to the object once it is recognized, as shown in our previous experiment.

5.7.2.3 Execution time

The execution time of this experiment is detailed in Table 5.6. We show the median and the maximum execution time of each step per cluster and per image. The overall approach can be executed in less than 30 ms on average per image, so working with a SLAM algorithm at 30Hz would be feasible. All the steps require a low execution time, including the conversion of ORB features into bag-of-words vectors and the database query. The conversion execution time depends mainly on the number of trees of the vocabulary and their branching factor. We could reduce their number, obtaining fewer words in the vocabulary, to speed up this step, but this would affect the performance of the feature descriptor discretization. The time to query the database is remarkably low, less than 1 ms on average. This indicates our approach is scalable with regard to the amount of objects. Since this query time grows linearly with the number of elements in the database, as we could see in Figure 4.13 in page 80, we may have a one order of magnitude larger database with little impact in the execution time. The geometrical check, which comprises the computation of correspondences by using the direct index (DI) and the RANSAC loop to solve the PnP problem, presents also a low time consumption.

	Bag-of-words approach	Independent view approach
Bio	34	51
Van	95	100
Orbit	152	147
Lion	238	237
Card1	169	171
Card2	229	243

Table 5.7: Elapsed time until first detection (s)

5.7.2.4 Comparison with previous approach

The low execution time of our approach allows to process a higher number of images per unit of time, increasing the recall level shown in previous section when the image sequence is considered as a whole instead of in a per image basis. In order to show this and check the feasibility of our object detector for visual SLAM applications in comparison with usual approaches using floating-point features, we compared this approach with the previous one based on views of SURF features. In the previous experiment, the detector skipped objects after their first recognition, because once their features were detected, the SLAM algorithm kept track of them to locate the object in the visual map. Because of this, we show in Table 5.7 the elapsed time in the tested video sequence from the beginning until the first detection of each object.

We can see that objects bio and card2 are found much earlier than in our previous work. The fast response of our system allows to perform recognition in a higher number of frames. Our previous system finds the object orbit 5 seconds before than our current approach. This case occurs because the SURF features obtain a successful match with a query image which present a scale change larger than the 2 octaves of our ORB features. In the rest of the cases there are no differences between both approaches. This shows that our current system with ORB features produces successful results, as a SURF-based system, but requiring much lower computation time, which allows to obtain a response in the same amount of time.

5.8 Discussion

We presented two different approaches to solve the problem of recognizing 3D textured objects and estimating its pose in the space. We successfully tested them and showed results working with a real-time visual SLAM system.

Object recognition based on independent view models constructed with structure-from-motion techniques and a state-of-the-art monocular SLAM algorithm were combined to allow the insertion of precomputed known objects into a standard point-based monocular SLAM map. The main input to the algorithm is visual information: a monocular sequence feeds the EKF SLAM algorithm; the appearance and geometric models for the known objects are precomputed from a set of sparse images; and also object recognition is driven by visual features. Experimental results show the feasibility of the algorithm and its real-time capabilities for room-sized scenarios.

Taking advantage of our previous research on bags of words based on binary descriptors to solve the place recognition problem, we applied these techniques to the object recognition in our second approach. This allows us to address two drawbacks present in the previous recognition approach. Firstly, representing an object as disjoints sets of features can result in redundant information resulting from overlapping views, increasing the matching time. Secondly, accessing object models and views must be done sequentially. By using a single bag of words to repre-

sent the whole surface of an object, we eliminate redundant information, and with an efficient database with inverted and direct indexes we speed up the model retrieval. Furthermore, binary features strikingly reduces the execution time of managing image features. Overall, this approach, aimed at SLAM applications for semantic mapping, is fast and scalable to perform 3D object recognition. Our tests have shown that our approach can successfully recognize several objects without yielding any false positive detection, running at 30Hz, allowing it to run along with a SLAM algorithm. The average recall of 26% presented in our experiments usually suffices for a SLAM application working at video rate, as we showed by comparing this system with the previous approach based on SURF features. This recall means that a recognition, providing the object pose, could be fired every 120ms. We could improve the recall at the expense of increasing the execution time of the algorithm (e.g. by increasing the number of RANSAC iterations). However, the working frequency of the algorithm would decrease, so the recall per unit of time would remain similar.

Regarding the cameras used: the camera used for building the models can be different from the one used to perform object recognition and visual SLAM. Any robot with a calibrated camera would be able to exploit then the precomputed models, what makes the system interoperable. The robot ends up with the location of the object it is supposed to interact with under quite general circumstances. We also showed that when using a SLAM algorithm, just by recognizing a part of an object once, the SLAM map can incorporate information about object regions that are not observed. This might be quite useful for tasks like robot navigation.

The work presented here was the first one introducing general 3D objects in a geometric SLAM map in real-time. There is a potential value residing on the concept underneath this work. On the one hand recent research on visual object recognition allows to robustly recognize a wide extent of objects from visual input; but 3D information is rarely considered in those approaches. On the other hand, monocular SLAM and structure from motion currently offer real-time camera motion and 3D scene estimation but without a semantic meaning. The combination presented here, providing a partially annotated local map and the current robot position, could be of high value for certain robotic tasks like grasping (as demonstrated in the RoboEarth experiment).

We pointed out our system always returns the correct object in the scene, but the transformation computed by the PnP algorithm may not be accurate enough. This could be enhanced in future work, for example, by performing query expansions (Chum et al. 2007) with the features matched when an object is detected. An expanded query may produce richer matches between the image and the model than a single query, resulting in more data to compute a more accurate object pose.

Other interesting lines for future work arise from the point of view of semantic mapping. First, if would be very interesting to increase the quality and density of the semantic annotations. For example, the object recognition based on instances could be upgraded to *category* recognition (Ferrari et al. 2010). This would allow to recognize generic categories (e.g., the category chair) instead of specific instantiations of a category (e.g., a specific chair). Context-based object detection (Murphy et al. 2003) or image segmentation (Gould et al. 2010) could also help to augment the density of the annotated objects. Second, monocular SLAM algorithms providing with denser geometric maps (Newcombe & Davison 2010, Strasdat et al. 2010) could be used in order to help robotic tasks like navigation or path planning.

Chapter 6

Conclusions

In this thesis we have presented our image matching framework in large-scale databases to address two related problems: place recognition for loop closing detection and object recognition for semantic mapping. We have shown how our pipeline can be adapted to these problems obtaining reliable results with video-rate efficiency.

6.1 Loop closing detection

Along this work we have detailed our algorithm to solve the loop closing detection problem in visual SLAM. We started by presenting our initial technique with SURF features, a visual vocabulary and an inverted index. We have proposed several novelties with respect to other approaches and have also shown their advantages, overcoming the state of the art.

Our work benefits from the sequentiality of the frames acquired with a camera. This is specially exploited during the database query and the process to obtain a candidate match, requiring consistency with previous loop detections. In contrast with other probabilistic algorithms, our technique successes working at low and high frequencies, despite the much or little overlap existing between consecutive images.

We have concluded that a posterior geometrical verification is necessary to filter out the false positives yielded by loop detection based on appearance. We have made use of several techniques to robustly discard those cases, for single and multiple cameras, both considering epipolar geometry and probabilistic inference with geometrical properties of reconstructed 3D points.

We have then shown how our algorithm is improved with binary features, which are clustered by a visual vocabulary of binary words. To the best of our knowledge, this has been the first time where the loop closing detection problem is addressed in such a way. We have also introduced direct indexes for visual words to speed up the computation of feature correspondences.

We have evaluated our system keeping the same configuration and vocabulary settings throughout an extensive evaluation with heterogeneous datasets, separating evaluation data from training data. We have obtained a video-rate loop closing detection algorithm able to fire a detection in 22ms on average with more than 26000 images. Therefore, we can claim that our loop detection system offers robust and efficient performance in a wide range of real situations, without any additional tuning.

6.2 3D Object recognition

We have first presented a 3D object recognition algorithm that, together with a visual SLAM algorithm, have composed one of the first systems that can create 3D maps with 3D objects, overcoming the limitations of other state-of-the-art systems.

We have shown how our framework can be adapted to carry out the recognition of 3D objects with visual vocabularies and binary features. This visual vocabulary approach allows objects to be modeled as a single bag of words representing the whole surface of the object, associating words with the 3D coordinates of a point cloud, allowing a very fast object candidate retrieval.

Our final object recognition algorithm also makes use of a novel two-stage clustering process to segment regions of interest in the query image. This enables to perform object detection in small areas of the image without checking exhaustively the whole image by sliding a window.

Putting all the pieces together, we obtain a 3D object recognition algorithm that is able to run at video-rate (around 28ms per image) with medium-sized object databases, so that it can run with any visual SLAM algorithm to take advantage of it to refine the pose of the recognized objects after inserting them in the map.

6.3 Future work

In the near future, the next step to enhance this work is to put together both faces of our image matching framework, by joining the real-time place and object recognition algorithms. Technically speaking, they fit nicely because both of them are constructed on FAST key points, the same feature used by other monocular SLAM system, such as the parallel tracking and mapping algorithm (PTAM) (Klein & Murray 2009).

Both approaches can benefit from each other, since the loop closing detection allows to improve the map that, at the same time, refines the pose of recognized objects. This is an immediate advantage; but others can be glimpsed. By recognizing objects we can notice image features that belong to objects and ignore them to perform SLAM or loop detection if we know those objects are movable. Semantic maps play an important role here by providing the necessary external information about objects.

A video-rate object recognition also makes it possible to track objects by detection. A first detection can serve as a position cue for consecutive detections, so that next recognitions can be focused on a single object and on a certain region of the image. Tracking objects and separating their image features in the mapping stage would allow to create maps that deal with a mobile camera and mobile objects at the same time, a problem that has not been addressed yet.

All this paves the road to new research for robotics and other areas as augmented reality, so that it is not difficult to imagine new applications. For example, in a factory environment, workers may wear a helmet or camera glasses where a 3D Object SLAM system is integrated to keep track of the motion of the user, whereas they obtain instructions and information about the places in which they are or the tools they use.

Appendixes

Appendix A

Inference with conditional random fields

This appendix describes the learning and inference steps in Cadena's (2011) CRF-Matching algorithm that we used together with our loop detection approach in Cadena et al. (2012).

A.1 CRF model

Conditional random fields (CRFs) are a case of Markov Random Fields (and thus satisfy the Markov properties) where there is no need to model the distribution over the observations (Koller & Friedman 2009, Bishop 2006). If the neighborhood of a node A (i.e. all nodes with edges to A) in the graph is known, the assignment to A is independent of the assignment to another node B outside the neighborhood of A. By definition, the minimum spanning tree, as those used in Section 3.3.3, connects points that are close in the measurement space, highlighting intrinsic localities in the scene. This implies: first, that the associations are jointly compatible within neighborhoods, and second, that the compatibility is enforced and propagated from neighborhood to neighborhood by the edge between them.

Instead of relying on the Bayes' rule to estimate the distribution over hidden states $\mathbf{x}$ from observations $\mathbf{z}$, CRFs directly model $p(\mathbf{x}|\mathbf{z})$, the *conditional distribution* over the hidden variables given observations. Due to this structure, CRFs can handle arbitrary dependencies between the observations. This makes them substantially flexible when using complex and overlapped attributes or observations. The nodes in a CRF represent hidden states, denoted $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \rangle$, observations are denoted $\mathbf{z}$. In Cadena's (2011) framework the hidden states correspond to all the possible associations between the n features in image I_t and the m features in image $I_{t'}$, i.e. $\mathbf{x}_i \in \{0, 1, 2, \ldots, m\}$, where the additional state 0 is the no-match state. Observations are the features extracted from the images and their disparity information. The nodes $\mathbf{x}_i$ along with the connectivity structure represented by the undirected graph define the conditional distribution $p(\mathbf{x}|\mathbf{z})$ over the hidden states $\mathbf{x}$. Let $\mathcal{Q}$ be the set of cliques (fully connected subsets) in the graph of a CRF. Then, a CRF factorizes the conditional distribution into a product of *clique potentials* $\phi_{\mathbf{q}}(\mathbf{x}_{\mathbf{q}}, \mathbf{z})$, where every $\mathbf{q} \in \mathcal{Q}$ is a clique in the graph, and $\mathbf{z}$ and $\mathbf{x}_{\mathbf{q}}$ are the observed data and the hidden nodes in such clique. Clique potentials are functions that map variable configurations to non-negative numbers. Intuitively, a potential captures the "compatibility" among the variables in the clique: the larger a potential value, the more likely the configuration. Using the clique potential, the conditional distribution over hidden states is

written as:

$$p(\mathbf{x}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})} \prod_{\mathbf{q} \in \mathcal{Q}} \phi_{\mathbf{q}}(\mathbf{x_q}, \mathbf{z}) \tag{A.1}$$

where $Z(\mathbf{z}) = \sum_{\mathbf{x}} \prod_{\mathbf{q} \in \mathcal{Q}} \phi_{\mathbf{q}}(\mathbf{x_q}, \mathbf{z})$ is the normalizing partition function. The computation of this function can be exponential in the size of $\mathbf{x}$. Hence, exact inference is possible for a limited class of CRF models only, e.g. in tree-structured graphs.

Potentials $\phi_{\mathbf{q}}(\mathbf{x_q}, \mathbf{z})$ are described by log-linear combinations of *descriptor functions* $\mathbf{f}$, i.e., the conditional distribution (A.1) can be rewritten as:

$$p(\mathbf{x}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})} \exp \left\{ \sum_{\mathbf{q} \in \mathcal{Q}} \mathbf{w}^T \cdot \mathbf{f}(\mathbf{x_q}, \mathbf{z}) \right\} \tag{A.2}$$

where $\mathbf{w}$ is a weight vector that represents the importance of different features for correctly identifying the hidden states. These weights can be learned from labeled training data.

A.2 Parameter learning

The goal of parameter learning is to determine the weights of the descriptor functions used in the conditional likelihood (A.2). CRFs learn these weights discriminatively by maximizing the conditional likelihood of labeled training data. We resort to maximizing the *pseudo-likelihood* of the training data, which is given by the product of all local likelihoods $p(\mathbf{x}_i|\mathrm{MB}(\mathbf{x}_i))$; where $\mathrm{MB}(\mathbf{x}_i)$ is the Markov Blanket of variable $\mathbf{x}_i$, which contains the immediate neighbors of $\mathbf{x}_i$ in the CRF graph. Optimization of this pseudo-likelihood is performed by minimizing the negative of its log, resulting in the following objective function:

$$L(\mathbf{w}) = -\sum_{i=1}^{n} \log p(\mathbf{x}_i|\mathrm{MB}(\mathbf{x}_i), \mathbf{w}) + \frac{\mathbf{w}^T\mathbf{w}}{2\sigma_{\mathbf{w}}^2} \tag{A.3}$$

The rightmost term in equation (A.3) serves as a zero-mean Gaussian prior, with variance $\sigma_{\mathbf{w}}^2$, on each component of the weight vector. The training data are labeled using RANSAC over the best rigid-body transformation in 6DoF (Olson 2008) for $G_{3\mathrm{D}}$ and over the fundamental matrix for G_I, after feature matching of two consecutive scenes.

A.3 Inference

Inference in a CRF estimates the marginal distribution of each hidden variable $\mathbf{x}_i$, and can thus determine the most likely configuration of the hidden variables $\mathbf{x}$ (i.e., the maximum a posteriori, or MAP, estimation). Both tasks can be solved using *belief propagation* (BP) (Pearl 1988), which works by transmitting messages containing beliefs through the graph structure of the model. Each node sends messages to its neighbors based on the messages it receives and the clique potentials. BP generates exact results in graphs with no loops, such as trees or polytrees.

Bibliography

Aherne, Frank J., Neil A. Thacker & Peter Rockett (1998), 'The Bhattacharyya metric as an absolute similarity measure for frequency coded data', *Kybernetika* **34**(4), 363–368.

Alahi, Alahi, Raphaël Ortiz & Pierre Vandergheynst (2012), FREAK: Fast retina keypoint, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', pp. 510–517.

An, Shan, Xin Ma, Rui Song & Yibin Li (2009), Face detection and recognition with SURF for human-robot interaction, *in* 'IEEE International Conference on Automation and Logistics', pp. 1946–1951.

Angeli, Adrien, David Filliat, Stéphane Doncieux & Jean-Arcady Meyer (2008), 'Fast and incremental method for loop-closure detection using bags of visual words', *IEEE Transactions on Robotics* **24**(5), 1027–1037.

Anguelov, Dragomir, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers & James Davis (2005), 'SCAPE: shape completion and animation of people', *ACM Transactions on Graphics* **24**(3), 408–416.

Arthur, David & Sergei Vassilvitskii (2007), k-means++: The advantages of careful seeding, *in* 'Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete algorithms', pp. 1027–1035.

Baeza-Yates, Ricardo A. & Berthier Ribeiro-Neto (1999), *Modern Information Retrieval*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bailey, Tim & Hugh Durrant-Whyte (2006), 'Simultaneous Localization and Mapping (SLAM): Part II', *IEEE Robotics Automation Magazine* **13**(3), 108–117.

Bao, Sid Y., Mohit Bagra, Yu-Wei Chao & Silvio Savarese (2012), Semantic structure from motion with points, regions, and objects, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 2703–2710.

Bay, Herbert, Andreas Ess, Tinne Tuytelaars & Luc Van Gool (2008), 'Speeded-Up Robust Features (SURF)', *Computer Vision and Image Understanding* **110**(3), 346–359.

Bay, Herbert, Vittorio Ferrari & Luc Van Gool (2005), Wide-baseline stereo matching with line segments, *in* 'IEEE Conference on Computer Vision and Pattern Recognition', Vol. 1, pp. 329–336.

Beetz, Michael, Ulrich Klank, Alexis Maldonado, Dejan Pangercic & Thomas Rühr (2011), Robotic roommates making pancakes - look into perception-manipulation loop, *in* 'IEEE International Conference on Robotics and Automation (ICRA), Workshop on Mobile Manipulation: Integrating Perception and Manipulation', pp. 529–536.

Bhattacharyya, A. (1943), 'On a measure of divergence between two statistical populations defined by their probability distributions', *Bulletin of the Calcutta Mathematical Society* **35**, 99–109.

Bishop, Christopher M. (2006), *Pattern Recognition and Machine Learning*, Springer, ISBN: 978-0-387-31073-2.

Blanco, José-Luis, Francisco-Angel Moreno & Javier González (2009), 'A collection of outdoor robotic datasets with centimeter-accuracy ground truth', *Autonomous Robots* **27**(4), 327–351.

Boutell, Matthew R., Jiebo Luo, Xipeng Shen & Christopher M. Brown (2004), 'Learning multi-label scene classification', *Pattern Recognition* **37**(9), 1757–1771.

Cadena, César (2011), Efficient Simultaneous Localisation And Mapping in Large and Complex Environments, Habilitation, Universidad de Zaragoza, Department of Computer Science and Systems Engineering, Zaragoza, Spain.

Cadena, César, Dorian Gálvez-López, Fabio Ramos, Juan D. Tardós & José Neira (2010), Robust place recognition with stereo cameras, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 5182–5189.

Cadena, César, Dorian Gálvez-López, Juan D. Tardós & José Neira (2012), 'Robust place recognition with stereo sequences', *IEEE Transactions on Robotics* **28**(4), 871–885.

Callmer, Jonas, Karl Granström, Juan Nieto & Fabio Ramos (2008), Tree of words for visual loop closure detection in urban SLAM, *in* 'Proceedings of the Australasian Conference on Robotics and Automation', Canberra, Australia, p. 8.

Calonder, Michael, Vincent Lepetit, Christoph Strecha & Pascal Fua (2010), BRIEF: binary robust independent elementary features, *in* 'European conference on Computer vision', ECCV'10, Springer-Verlag, Berlin, Heidelberg, pp. 778–792.

Calonder, Michael, Vincent Lepetit, Pascal Fua, Kurt Konolige, James Bowman & Patrick Mihelich (2010), Compact signatures for high-speed interest point description and matching, *in* '12th IEEE International Conference on Computer Vision', pp. 357–364.

Canny, John (1986), 'A computational approach to edge detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-8**(6), 679–698.

Castle, Robert O. & David W. Murray (2011), 'Keyframe-based recognition and localization during video-rate parallel tracking and mapping', *Image and Vision Computing* **29**(8), 524–532.

Chow, C. & C. Liu (1968), 'Approximating discrete probability distributions with dependence trees', *IEEE Transactions on Information Theory* **14**(3), 462–467.

Chum, O., J. Philbin, J. Sivic, M. Isard & A. Zisserman (2007), Total recall: Automatic query expansion with a generative feature model for object retrieval, *in* 'IEEE International Conference on Computer Vision', pp. 1–8.

Civera, Javier, Andrew Davison & J. M. M. Montiel (2008), 'Inverse depth parametrization for monocular SLAM', *IEEE Transactions on Robotics* **24**(5), 932–945.

Civera, Javier, Dorian Gálvez-López, Luis Riazuelo, Juan D. Tardós & J. M. M. Montiel (2011), Towards semantic slam using a monocular camera, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1277–1284.

Civera, Javier, Óscar G. Grasa, Andrew Davison & J. M. M. Montiel (2010), '1-point ransac for ekf filtering: Application to real-time structure from motion and visual odometry', *Journal of Field Robotics* **27**(5), 609–631.

Collet, Alvaro, Manuel Martinez & Siddhartha S. Srinivasa (2011), 'The MOPED framework: Object Recognition and Pose Estimation for Manipulation', *The International Journal of Robotics Research* **30**(10), 1284–1306.

Coppersmith, Don & Shmuel Winograd (1987), Matrix multiplication via arithmetic progressions, *in* 'Proceedings of the nineteenth annual ACM symposium on Theory of computing', STOC '87, ACM, New York, NY, USA, pp. 1–6.

Cummins, Mark & Paul Newman (2008), 'FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance', *The International Journal of Robotics Research* **27**(6), 647–665.

Cummins, Mark & Paul Newman (2011), 'Appearance-only SLAM at large scale with FAB-MAP 2.0', *The International Journal of Robotics Research* **30**(9), 1100–1123.

DARPA (2007), 'Darpa grand challenge'.
URL: *http://archive.darpa.mil/grandchallenge*

Davison, Andrew, Ian Reid, Nicholas D. Molton & Olivier Stasse (2007), 'MonoSLAM: Real-time single camera SLAM', *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **29**(6), 1052–1067.

Douze, Matthijs, Hervé Jégou, Harsimrat Sandhawalia, Laurent Amsaleg & Cordelia Schmid (2009), Evaluation of GIST descriptors for web-scale image search, *in* 'ACM International Conference on Image and Video Retrieval', CIVR '09, ACM, New York, NY, USA, pp. 19:1–19:8.

Durrant-Whyte, Hugh & Tim Bailey (2006), 'Simultaneous Localization and Mapping: Part I', *IEEE Robotics Automation Magazine* **13**(2), 99–110.

Eade, Ethan D. & Tom W. Drummond (2008), Unified loop closing and recovery for real time monocular slam, *in* 'Proceedings of the British Machine Vision Conference', pp. 6.1–6.10.

Ekvall, Staffan & Danica Kragic (2005), Receptive field cooccurrence histograms for object detection, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 84–89.

Ekvall, Staffan, Patric Jensfelt & Danica Kragic (2006), Integrating active mobile robot object recognition and slam in natural environments, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 5792–5797.

Faugeras, Olivier D. & Francis Lustman (1988), 'Motion and structure from motion in a piecewise planar environment', *International Journal of Pattern Recognition and Artificial Intelligence* **2**(3), 485–508.

Ferrari, Vittorio, Frederic Jurie & Cordelia Schmid (2010), 'From images to shape models for object detection', *International journal of computer vision* **87**(3), 284–303.

Fischler, Martin A. & Robert C. Bolles (1981), 'Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography', *Communications of the ACM* **24**(6), 381–395.

Friedman, Jerome H., Jon Louis Bentley & Raphael Ari Finkel (1977), 'An algorithm for finding best matches in logarithmic expected time', *ACM Transactions on Mathematical Software* **3**(3), 209–226.

Furukawa, Yasutaka & Jean Ponce (2010), 'Accurate, dense, and robust multiview stereopsis', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **32**(8), 1362–1376.

Galindo, Cipriano, Alessandro Saffiotti, Silvia Coradeschi, P. Buschka, Juan Antonio Fernandez-Madrigal & Javier Gonzalez (2005), Multi-hierarchical semantic maps for mobile robotics, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 2278–2283.

Gálvez-López, Dorian & Juan D. Tardós (2011), Real-time loop detection with bags of binary words, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 51–58.

Gálvez-López, Dorian & Juan D. Tardós (2012), 'Bags of binary words for fast place recognition in image sequences', *IEEE Transactions on Robotics* **28**(5), 1188–1197.

Gálvez-López, Dorian, Kristoffer Sjöö, Chandana Paul & Patric Jensfelt (2008), Hybrid laser and vision based object search and localization, *in* 'IEEE International Conference on Robotics and Automation', pp. 2636–2643.

Gauglitz, Steffen, Tobias Höllerer & Matthew Turk (2011), 'Evaluation of interest point detectors and feature descriptors for visual tracking', *International Journal of Computer Vision* **94**(3), 335–360. 10.1007/s11263-011-0431-5.

Gionis, Aristides, Piotr Indyk & Rajeev Motwani (1999), Similarity search in high dimensions via hashing, *in* '25th International Conference on Very Large Data Bases', VLDB '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 518–529.

Gould, Stephen, Richard Fulton & Daphne Koller (2010), Decomposing a scene into geometric and semantically consistent regions, *in* '12th IEEE International Conference on Computer Vision', pp. 1–8.

Grabner, Michael, Helmut Grabner & Horst Bischof (2006), 'Fast approximated SIFT', *Asian Conference on Computer Vision* pp. 918–927.

Griffin, Gregory, Alex Holub & Pietro Perona (2007), Caltech-256 object category dataset, Technical Report 7694, California Institute of Technology.
URL: *http://authors.library.caltech.edu/7694*

Grisetti, Giorgio, Cyrill Stachniss & Wolfram Burgard (2009), 'Nonlinear constraint network optimization for efficient map learning', *IEEE Transactions on Intelligent Transportation Systems* **10**(3), 428–439.

Grundmann, Thilo, Wendelin Feiten & Georg von Wichert (2011), A Gaussian measurement model for local interest point based 6 DOF pose estimation, *in* 'IEEE International Conference on Robotics and Automation (ICRA)', pp. 2085 –2090.

Harris, Chris & Mike Stephens (1988), A combined corner and edge detector, *in* 'In Proceedings of Fourth Alvey Vision Conference', pp. 147–151.

Hartley, Richard I. (1997), 'In defense of the eight-point algorithm', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(6), 580–593.

Hartley, Richard I. & Andrew Zisserman (2004), *Multiple View Geometry in Computer Vision*, second edn, Cambridge University Press, ISBN: 0521540518.

Heymann, S., K. Müller, A. Smolic, Bernd Froehlich & T. Wiegand (2007), SIFT Implementation and Optimization for General-Purpose GPU, *in* 'International Conference in Central Europe on Computer Graphics and Visualization'.

Horn, Berthold K. P., Hugh M. Hilden & Shahriar Negahdaripour (1988), 'Closed-form solution of absolute orientation using orthonormal matrices', *Journal of the Optical Society of America A* **5**(7), 1127–1135.

Hsiao, Edward, Alvaro Collet & Martial Hebert (2010), Making specific features less discriminative to improve point-based 3D object recognition, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 2653–2660.

Ke, Yan & R. Sukthankar (2004), PCA-SIFT: a more distinctive representation for local image descriptors, *in* 'Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 2, pp. II–506 – II–513 Vol.2.

Klein, Georg & David Murray (2009), Parallel tracking and mapping on a camera phone, *in* 'Proceedings of the Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)', Orlando.

Koller, Daphne & Nir Friedman (2009), *Probabilistic Graphical Models: Principles and Techniques*, MIT Press, ISBN: 9780262013192.

Konolige, Kurt, James Bowman, J.D. Chen, Patrick Mihelich, Michael Calonder, Vincent Lepetit & Pascal Fua (2010), 'View-based maps', *The International Journal of Robotics Research* **29**(8), 941–957.

Kullback, Solomon & Richard A. Leibler (1951), 'On Information and Sufficiency', *Annals of Mathematical Statistics* **22**(1), 79–86.

Lafferty, John, Andrew McCallum & Fernando Pereira (2001), Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data, *in* 'Proceedings of the 18th International Conf. on Machine Learning', Morgan Kaufmann, San Francisco, CA, pp. 282–289.

Lepetit, Vincent, Francesc Moreno-Noguer & Pascal Fua (2009), 'EPnP: An accurate $O(n)$ solution to the PnP problem', *International Journal of Computer Vision* **81**(2), 155–166.

Lepetit, Vincent & Pascal Fua (2006), 'Keypoint recognition using randomized trees', *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp. 1465–1479.

Leutenegger, Stephan, Margarita Chli & Roland Siegwart (2011), BRISK: Binary Robust Invariant Scalable Keypoints, *in* 'IEEE International Conference on Computer Vision', Barcelona, pp. 2548 –2555.

López-Nicolás, Gonzalo, J.J. Guerrero & Carlos Sagüés (2010), 'Multiple homographies with omnidirectional vision for robot homing', *Robotics and Autonomous Systems* **58**(6), 773–783.

Lowe, David G. (1999), Object recognition from local scale-invariant features, *in* 'The Proceedings of the Seventh IEEE International Conference on Computer Vision', Vol. 2, pp. 1150–1157.

Lowe, David G. (2004), 'Distinctive image features from scale-invariant keypoints', *International Journal of Computer Vision* **60**(2), 91–110.

Majdik, Andras, Dorian Gálvez-López, Gheorghe Lazea & Jose A. Castellanos (2011), Adaptive appearance based loop-closing in heterogeneous environments, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', pp. 1256–1263.

Meger, David, Per-Erik Forssén, Kevin Lai, Scott Helmer, Sancho McCann, Tristram. Southey, Matthew Baumann, James J. Little & David G. Lowe (2008), 'Curious George: An attentive semantic robot', *Robotics and Autonomous Systems* **56**(6), 503–511.

Mikolajczyk, Krystian & Cordelia Schmid (2005), 'A performance evaluation of local descriptors', *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp. 1615–1630.

Muja, Marius & David G. Lowe (2009), Fast approximate nearest neighbors with automatic algorithm configuration, *in* 'International Conference on Computer Vision Theory and Application', pp. 331–340.

Muja, Marius & David G. Lowe (2012), Fast matching of binary features, *in* 'Ninth Conference on Computer and Robot Vision (CRV)', pp. 404–410.

Murillo, Ana C., J. J. Guerrero & Carlos Sagüés (2007), SURF features for efficient robot localization with omnidirectional images, *in* 'IEEE International Conference on Robotics and Automation', pp. 3901–3907.

Murphy, Kevin P., Antonio Torralba & William T. Freeman (2003), *Using the forest to see the trees: a graphical model relating features, objects and scenes*, Vol. 16, MIT Press, Vancouver, BC.

Naik, Nikhil, Sanmay Patil & Madhuri Joshi (2009), A scale adaptive tracker using hybrid color histogram matching scheme, *in* '2nd International Conference on Emerging Trends in Engineering and Technology (ICETET)', pp. 279–284.

Newcombe, Ricahrd A. & Andrew Davison (2010), Live dense reconstruction with a single moving camera, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', IEEE, pp. 1498–1505.

Nister, David & Henrik Stewenius (2006), Scalable recognition with a vocabulary tree, *in* 'IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 2, pp. 2161–2168.

Oliva, Aude & Antonio Torralba (2001), 'Modeling the shape of the scene: A holistic representation of the spatial envelope', *International Journal of Computer Vision* **42**(3), 145–175.

Olson, Edwin (2008), Robust and Efficient Robotic Mapping, PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Olson, Edwin (2009), 'Recognizing places using spectrally clustered local matches', *Robotics and Autonomous Systems* **57**(12), 1157–1172.

OpenCV (2009), 'Open source computer vision library. URL: http://opencv.willowgarage.com'.
URL: *http://opencv.willowgarage.com*

Pandey, Gaurav, James R. McBride & Ryan M. Eustice (2011), 'Ford campus vision and lidar data set', *The International Journal of Robotics Research* **30**(13), 1543–1552.
URL: *http://robots.engin.umich.edu/SoftwareData/Ford*

Pangercic, Dejan, Vladimir Haltakov & Michael Beetz (2011), Fast and robust object detection in household environments using vocabulary trees with sift descriptors, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Workshop on Active Semantic Perception and Object Search in the Real World', San Francisco, CA, USA.

Paul, Rohan & Paul Newman (2010), FAB-MAP 3D: Topological mapping with spatial and visual appearance, *in* 'IEEE International Conference on Robotics and Automation', pp. 2649–2656.

Paz, Lina M., Pedro Piniés, Juan D. Tardós & José Neira (2008), 'Large-scale 6-DOF SLAM with stereo-in-hand', *IEEE Transactions on Robotics* **24**(5), 946–957.

Pearl, Judea (1988), *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann Publishers Inc., ISBN: 0-934613-73-7, San Francisco, CA, USA.

Piniés, Pedro & Juan D. Tardós (2008), 'Large-scale SLAM building conditionally independent local maps: Application to monocular vision', *IEEE Transactions on Robotics* **24**(5), 1094–1106.

Piniés, Pedro, Lina M. Paz, Dorian Gálvez-López & Juan D. Tardós (2010), 'CI-Graph SLAM for 3D Reconstruction of Large and Complex Environments using a Multicamera System', *International Journal of Field Robotics* **27**(5), 561–586.

Pretto, Alberto, Emanuele Menegatti & Enrico Pagello (2007), Reliable features matching for humanoid robots, *in* '7th IEEE-RAS International Conference on Humanoid Robots', pp. 532–538.

Pronobis, Andrzej, Óscar Martinez Mozos, Barbara Caputo & Patric Jensfelt (2010), 'Multimodal semantic place classification', *The International Journal of Robotics Research (IJRR)* **29**(2-3), 298–320.

Quinlan, J. Ross (1986), 'Induction of decision trees', *Machine Learning* **1**(1), 81–106.

Ramos, Fabio, Dieter Fox & Hugh Durrant-Whyte (2007), CRF-Matching: Conditional Random Fields for Feature-Based Scan Matching, *in* 'Robotics: Science and Systems (RSS)'.

Ramos, Fabio, Mohammed Waleed Kadous & Dieter Fox (2008), Learning to associate image features with CRF-Matching, *in* 'The 11th International Symposium of Experimental Robotics (ISER)', pp. 505–514.

Rawseeds (2007-2009), 'Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets (Project FP6-IST-045144).'.
URL: *http://www.rawseeds.org/rs/datasets*

Rosin, Paul L. (1999), 'Measuring corner properties', *Computer Vision and Image Understanding* **73**(2), 291–307.

Rosten, Edward & Tom Drummond (2006), 'Machine learning for high-speed corner detection', *European Conference on Computer Vision (ECCV)* pp. 430–443.

Rublee, Ethan, Vincent Rabaud, Kurt Konolige & Gary Bradski (2011), ORB: An Efficient Alternative to SIFT or SURF, *in* 'IEEE International Conference on Computer Vision', pp. 2564–2571.

Rusu, Radu B., Zoltan C. Marton, Nico Blodow, Andreas Holzbach & Michael Beetz (2009), Model-based and learned semantic object labeling in 3D point cloud maps of kitchen environments, *in* 'IEEE/RSJ International Conference on Intelligent Robots and Systems', pp. 3601–3608.

Sattler, Torsten, Bastian Leibe & Leif Kobbelt (2011), Fast image-based localization using direct 2D-to-3D matching, *in* 'IEEE International Conference on Computer Vision (ICCV)', pp. 667 –674.

Schaffalitzky, Frederik & Andrew Zisserman (2002), Multi-view matching for unordered image sets, or "How do I organize my holiday snaps?", *in* '7th European Conference on Computer Vision-Part I', ECCV '02, Springer-Verlag, London, UK, UK, pp. 414–431.

Sheikh, Yaser A., Erum A. Khan & Takeo Kanade (2007), Mode-seeking by Medoidshifts, *in* '11th IEEE International Conference on Computer Vision', pp. 1–8.

Silpa-Anan, Chanop & Richard Hartley (2008), Optimised KD-trees for fast image descriptor matching, *in* 'IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', pp. 1–8.

Sivic, Josef & Andrew Zisserman (2003), Video Google: A text retrieval approach to object matching in videos, *in* 'IEEE International Conference on Computer Vision', Vol. 2, pp. 1470–1477.

Sivic, Josef & Andrew Zisserman (2009), 'Efficient visual search of videos cast as text retrieval', *IEEE Transactions on Pattern Analysis and Machine Intelligence* **31**(4), 591 –606.

Sjöö, Kristoffer, Dorian Gálvez-López, Chandana Paul, Patric Jensfelt & Danica Kragic (2009), 'Object search and localization for an indoor mobile robot', *Journal of Computing and Information Technology* **17**(1), 67–80.

Smith, Mike, Ian Baldwin, Winston Churchill, Rohan Paul & Paul Newman (2009), 'The new college vision and laser data set', *The International Journal of Robotics Research* **28**(5), 595–599.
URL: *http://www.robots.ox.ac.uk/NewCollegeData*

Snavely, Noah, Steven M. Seitz & Richard Szeliski (2006), 'Photo tourism: exploring photo collections in 3d', *ACM Transactions on Graphics* **25**(3), 835–846.

Steder, Bastian, Giorgio Grisetti & Wolfram Burgard (2010), Robust place recognition for 3D range data based on point features, *in* 'IEEE International Conference on Robotics and Automation', pp. 1400 –1405.

Strasdat, Hauke, J. M. M. Montiel & Andrew Davison (2010), Scale drift-aware large scale monocular SLAM, *in* 'Proceedings of Robotics: Science and Systems (RSS)'.

Szeliski, Richard (2010), *Computer vision: Algorithms and applications*, Springer, ISBN 978-1-84882-934-3.

Tenorth, Moritz, Alexander Perzylo, Reinhard Lafrenz & Michael Beetz (2012), The RoboEarth language: Representing and exchanging knowledge about actions, objects, and environments, *in* 'IEEE International Conference on Robotics and Automation', pp. 1284–1289.

Torralba, Antonio, Kevin P. Murphy, William T. Freeman & Mark A. Rubin (2003), Context-based vision system for place and object recognition, *in* 'IEEE International Conference on Computer Vision', pp. 273–280 vol.1.

Triggs, Bill, Philip McLauchlan, Richard Hartley & Andrew Fitzgibbon (2000), Bundle adjustment – A modern synthesis, *in* 'Vision Algorithms: Theory and Practice', LNCS, Springer Verlag, pp. 298–375.

Trzcinski, Tomasz, Vincent Lepetit & Pascal Fua (2012), 'Thick boundaries in binary space and their influence on nearest-neighbor search', *Pattern Recognition Letters* **33**(16), 2173–2180.

Tuytelaars, Tinne & Krystian Mikolajczyk (2008), 'Local invariant feature detectors: a survey', *Foundations and Trends in Computer Graphics and Vision* **3**(3), 177–280.

Tuytelaars, Tinne & Luc Van Gool (2004), 'Matching widely separated views based on affine invariant regions', *International Journal of Computer Vision* **59**(1), 61–85.

Valgren, Christoffer & Achim J. Lilienthal (2010), 'SIFT, SURF & seasons: Appearance-based long-term localization in outdoor environments', *Robotics and Autonomous Systems* **58**(2), 149–156.

Vargas, Manuel & Ezio Malis (2005), Visual servoing based on an analytical homography decomposition, *in* '44th IEEE Conference on Decision and Control, and European Control Conference. CDC-ECC '05.', pp. 5379–5384.

Vasudevan, Shrihari, Stefan Gächter, Viet Nguyen & Roland Siegwart (2007), 'Cognitive maps for mobile robots-an object based approach', *Robotics and Autonomous Systems* **55**(5), 359–371.

Viola, Paul & Michael Jones (2001), Rapid object detection using a boosted cascade of simple features, *in* 'IEEE Computer Society Conference on Computer Vision and Pattern Recognition', Vol. 1, pp. I–511 – I–518 vol.1.

Waibel, Markus, Michael Beetz, Raffaello D'Andrea, Rob Janssen, Moritz Tenorth, Javier Civera, Jos Elfring, Dorian Gálvez-López, Kai Haussermann, J.M.M. Montiel, Alexander C. Perzylo, Bjorn Schiessle, Oliver Zweigle & Rene van de Molengraft (2011), 'Roboearth', *IEEE Robotics Automation Magazine* **18**(2), 69–82.

Werner, Felix, Joaquin Sitte & Frederic Maire (2012), 'Topological map induction using neighbourhood information of places', *Autonomous Robots* **32**(4), 405–418.

Williams, Brian, Georg Klein & Ian Reid (2007), Real-time SLAM relocalisation, *in* '11th IEEE International Conference on Computer Vision', pp. 1–8.

Williams, Brian, Mark Cummins, José Neira, Paul Newman, Ian Reid & Juan D. Tardós (2009), 'A comparison of loop closing techniques in monocular SLAM', *Robotics and Autonomous Systems* **57**(12), 1188–1197.

Yang, Jun, Yu-Gang Jiang, Alexander G. Hauptmann & Chong-Wah Ngo (2007), Evaluating bag-of-visual-words representations in scene classification, *in* 'International Workshop on Multimedia Information Retrieval', MIR '07, ACM, New York, NY, USA, pp. 197–206.

Zender, H., O. Martínez Mozos, P. Jensfelt, G.J.M. Kruijff & W. Burgard (2008), 'Conceptual spatial representations for indoor mobile robots', *Robotics and Autonomous Systems* **56**(6), 493–502.

Zhang, Zhongfei & Allen R. Hanson (1996), 3d reconstruction based on homography mapping, *in* 'ARPA Image Understanding Workshop', pp. 0249–6399.