TESIS DE LA UNIVERSIDAD DE ZARAGOZA
2021
83

Berta Bescós Torcal

Visual slam in dynamic environments

Director/es
Neira Parra, José

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

VISUAL SLAM IN DYNAMIC ENVIRONMENTS

Autor

Berta Bescós Torcal

Director/es

Neira Parra, José

UNIVERSIDAD DE ZARAGOZA
Escuela de Doctorado

Programa de Doctorado en Ingeniería de Sistemas e Informática

2020

Universidad Zaragoza

PhD Thesis

Visual SLAM in Dynamic Environments

Autora

Berta Bescós Torcal

Director

José Neira Parra

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2020

Visual SLAM in Dynamic Environments

Supervisor

José Neira Parra	Universidad de Zaragoza, Spain

Dissertation Committee

Ana Cristina Murillo Arnal	Universidad de Zaragoza, Spain
Mariano Jáimez Tarifa	Facebook Reality Labs, Switzerland
Stefan Leutenegger	Imperial College London, UK
José Luis Blanco Claraco	Universidad de Almería, Spain
José Ángel Castellanos Gómez	Universidad de Zaragoza, Spain

External Examiners

César Cadena Lerma	ETH Zürich, Switzerland
Luca Carlone	MIT, USA

Funding

Beca FPI (BES-2016-077836)	Ministerio de Educacion, Spain
Proyecto DPI2015-68905	Direccion General de Investigacion, Spain
Proyecto PID2019-108398GB-I00	Direccion General de Investigacion, Spain
Grupo DGA T45-17R	Gobierno de Aragón, Spain

Agradecimientos

En primer lugar, quiero agradecer a mi supervisor José Neira por su ayuda y consejo, así como por confiar en mí durante todos estos años. En segundo lugar, estoy muy agradecida por los compañeros de departamento con los que he tenido la oportunidad no sólo de trabajar si no de poder conocer personalmente e incluso llegar a llamar amigos. También querría agradecer a mis alumnos Guoxiang, Laura, Víctor y Daniel porque estoy segura de que yo he aprendido tanto o más con ellos que ellos conmigo. Quiero nombrar también a Javier Civera, quien me introdujo a través de mi trabajo de fin de grado en el increíble mundo de la investigación y de la visión por computador.

Quiero agradecer sinceramente al Profesor Roland Siegwart, jefe del Autonomous Systems Lab (ASL) de la ETH de Zúrich, por acogerme en su laboratorio. Pasé cinco meses increíbles aprendiendo un montón, trabajando y compartiendo experiencias con César Cadena y todos los miembros de ASL en Zúrich. También agradezco a todos mis compañeros de trabajo en Oculus Zúrich, y especialmente a mi supervisor Mariano Jáimez, por hacerme sentir como uno más de ellos durante mis prácticas.

Por último pero no menos importante, quiero dar sinceramente las gracias a Rafa, a mi familia y a mis amigas por su apoyo incondicional a lo largo de estos años. Gracias.

Acknowledgements

I would like first of all to thank my supervisor, Prof. José Neira, for his help and advice, as well as for trusting me during all these years. Secondly, I am very grateful for the lab partners with whom I have had the opportunity not only to work but to meet personally and even call friends. I would also like to thank my students Guoxiang, Laura, Victor and Daniel because I am sure that I have learned as much or more with them as they have with me. I would also like to mention Javier Civera, who introduced me through my bacherlor's thesis to the incredible world of research.

I want to sincerely thank Prof. Roland Siegwart, the head of the Autonomous Systems Lab of ETH Zurich, for hosting me in his lab. I had five amazing months working and hanging out together with Dr. César Cadena and all the ASL members. I also thank all my coworkers at Oculus Zürich, and especially my supervisor Mariano Jáimez, for the warm welcome and making me feel as one of them during my internship.

Last but not least, I would like to sincerely thank Rafa, my family and my friends for their unconditional support over these years. Thank you.

SLAM Visual en Escenas Dinámicas

Resumen

El problema de localización y construcción visual simultánea de mapas (visual SLAM por sus siglas en inglés *Simultaneous Localization and Mapping*) consiste en localizar una cámara en un mapa que se construye de manera online. Esta tecnología permite la localización de robots en entornos desconocidos y la creación de un mapa de la zona con los sensores que lleva incorporados, es decir, sin contar con ninguna infraestructura externa. A diferencia de los enfoques de odometría en los cuales el movimiento incremental es integrado en el tiempo, un mapa permite que el sensor se localice continuamente en el mismo entorno sin acumular deriva.

Asumir que la escena observada es estática es común en los algoritmos de SLAM visual. Aunque la suposición estática es válida para algunas aplicaciones, limita su utilidad en escenas concurridas del mundo real para la conducción autónoma, los robots de servicio o realidad aumentada y virtual entre otros. La detección y el estudio de objetos dinámicos es un requisito para estimar con precisión la posición del sensor y construir mapas estables, útiles para aplicaciones robóticas que operan a largo plazo.

Las contribuciones principales de esta tesis son tres:

1. Somos capaces de detectar objetos dinámicos con la ayuda del uso de la segmentación semántica proveniente del aprendizaje profundo y el uso de enfoques de geometría multivisión. Esto nos permite lograr una precisión en la estimación de la trayectoria de la cámara en escenas altamente dinámicas comparable a la que se logra en entornos estáticos, así como construir mapas en 3D que contienen sólo la estructura del entorno estático y estable.

2. Logramos *alucinar* con imágenes realistas la estructura estática de la escena detrás de los objetos dinámicos. Esto nos permite ofrecer mapas completos con una representación plausible de la escena sin discontinuidades o vacíos ocasionados por las oclusiones de los objetos dinámicos. El reconocimiento visual de lugares también se ve impulsado por estos avances en el procesamiento de imágenes.

3. Desarrollamos un marco conjunto tanto para resolver el problema de SLAM como el seguimiento de múltiples objetos con el fin de obtener un mapa espacio-temporal con información de la trayectoria del sensor y de los alrededores. La comprensión de los objetos dinámicos circundantes es de crucial importancia para los nuevos requisitos de las aplicaciones emergentes de realidad aumentada/virtual o de la navegación autónoma.

Estas tres contribuciones hacen avanzar el estado del arte en SLAM visual. Como un producto secundario de nuestra investigación y para el beneficio de la comunidad científica, hemos liberado el código que implementa las soluciones propuestas.

Visual SLAM in Dynamic Environments

Abstract

Visual Simultaneous Localization and Mapping (SLAM) is the problem of localizing a camera in an unknown environment while building an online map. SLAM allows the localization of robots in unknown scenes by processing their on-board sensors and therefore not relying on any external infrastructure. Unlike odometry approaches where incremental motion is integrated over time, a map allows the sensor to be located continuously in the same environment without accumulating drift.

Assuming that the observed scene is static is common in visual SLAM algorithms. Although the static assumption holds for some robotic applications, it limits its usability in populated real-world scenarios for autonomous driving, service robots or AR/VR. Detecting and dealing with dynamic objects is a requisite to accurately estimate the sensor pose and to build stable maps, which are useful for long-term applications.

The overarching contributions of this thesis are threefold:

1. We are able to detect dynamic objects by leveraging the use of semantic segmentation from deep learning and the use of multi-view geometry approaches. This enables us to achieve an accuracy in the camera pose estimation in highly dynamic scenes comparable to that achieved in static environments, as well as to build 3D maps containing only the static and stable environment structure.

2. We succeed in inpainting with plausible imagery the static scene structure behind dynamic objects. This allows us to deliver complete maps with a plausible representation of the scene without discontinuities or gaps occasioned by the occlusions of the dynamic objects. The task of visual place recognition is also boosted by these advances in image inpainting.

3. We develop a joint framework for multi-object tracking and SLAM to deliver a spatial-temporal map with information from self and surroundings. Understanding surrounding dynamic objects is of crucial importance for the frontier requirements of emerging applications within AR/VR or autonomous things navigation.

These three contributions advance the state of the art in visual SLAM. As a *de facto* side-product of our research and for the benefit of the scientific community, we have made open-source the implementation of our proposed solutions.

Index

1 Introduction **1**
1.1 Visual Simultaneous Localization and Mapping . . . 1
1.1.1 Sensors . . . 2
1.1.2 Feature-Based Methods . . . 4
1.1.3 Direct Methods . . . 5
1.1.4 Bundle Adjustment . . . 5
1.1.5 Deep Learning Priors . . . 7
1.1.6 Front End and Back End . . . 7
1.2 Visual SLAM in Dynamic Environments . . . 8
1.2.1 Camera Tracking . . . 9
1.2.2 Mapping . . . 10
1.2.3 Place Recognition . . . 10
1.2.4 Dynamic Objects . . . 12
1.2.5 Challenges . . . 13
1.3 Contributions . . . 13
1.3.1 Detection . . . 14
1.3.2 Inpainting . . . 15
1.3.3 Multi-Object Tracking . . . 15
1.4 Organization of the Document . . . 16

2 Detection of Dynamic Objects **17**
2.1 Related Work . . . 17
2.1.1 Unsupervised Detection . . . 18
2.2 Detection of Moving Objects . . . 19
2.3 Detection of *a priori* Dynamic Objects . . . 23
2.3.1 Supervised Detection . . . 23
2.3.2 Semi-Supervised Detection . . . 24
2.4 Integration in a SLAM System . . . 26
2.4.1 Mask R-CNN . . . 27
2.4.2 Low-Cost Tracking . . . 28
2.4.3 Multi-View Geometry . . . 28
2.4.4 Tracking and Mapping . . . 29
2.5 Experimental Results . . . 29
2.5.1 TUM RGB-D Dataset . . . 29
2.5.2 KITTI Odometry Dataset . . . 32
2.5.3 Timing Analysis . . . 33
2.6 Failure Modes and Future Work . . . 34
2.7 Discussion . . . 34

2.8 Related Publications . . . 35

3 Inpainting of Dynamic Objects **37**
3.1 Related Work . . . 38
3.1.1 Video-Based Inpainting . . . 38
3.1.2 Image-Based Inpainting . . . 39
3.2 Geometric Inpainting . . . 40
3.3 Inpainting with Deep Learning . . . 42
3.3.1 Image-to-Image Translation . . . 42
3.3.2 Empty Cities . . . 44
3.4 Image-Based Experiments . . . 51
3.4.1 Data Generation . . . 51
3.4.2 Inpainting . . . 51
3.4.3 Transfer to Real Data . . . 55
3.5 Experiments . . . 59
3.5.1 Visual Odometry . . . 59
3.5.2 Visual Place Recognition . . . 64
3.5.3 Mapping . . . 66
3.5.4 Timing Analysis . . . 68
3.6 Failure Modes and Future Work . . . 68
3.7 Discussion . . . 70
3.8 Related Publications . . . 70

4 Multi-Object Tracking **73**
4.1 Related Work . . . 75
4.2 DynaSLAM II . . . 76
4.2.1 Notation . . . 77
4.2.2 Object Data Association . . . 77
4.2.3 Object-Centric Representation . . . 78
4.2.4 Bundle Adjustment with Objects . . . 79
4.2.5 Bounding Boxes . . . 81
4.3 Experiments . . . 84
4.3.1 Visual Odometry . . . 84
4.3.2 Multi-Object Tracking . . . 86
4.3.3 Timing Analysis . . . 88
4.4 Failure Modes and Future Work . . . 89
4.5 Discussion . . . 90
4.6 Related Publications . . . 90

5 Conclusions and Future Prospects **91**
5.1 Detection of Dynamic Objects . . . 91
5.2 Static Background Inpainting . . . 92
5.3 Multi-Object Tracking . . . 93

6 Summary of Results **95**
6.1 Research Stays . . . 95
6.2 Supervision of Students . . . 95
6.3 Dissemination . . . 96
6.3.1 Peer-Reviewed Publications . . . 96

6.3.2 Open-Source Software 97
6.3.3 Videos 97
6.3.4 Conference and Research Seminar Attendance 97
6.4 Peer Reviews 98

Chapter 1

Introduction

In the last decade, advancements in the field of mobile robotics have been notably pronounced. Increasing numbers of tasks that were traditionally performed by humans can now be accomplished by robots. This is particularly noteworthy in situations where such tasks are tedious, require a high degree of precision or pose a risk for human health and life. Robots are also becoming part of our daily routines, with robotic vacuum cleaners or smart assistants inhabiting our homes. Furthermore, mobile platforms will soon become ubiquitous in crowded spaces, coping better and better with dynamic obstacles and unstructured environments. This widespread use of mobile robotics poses challenges for the algorithms used to guarantee the safety and reliability of these devices.

There are numerous applications of mobile robotics that have evolved through advancements in simultaneous localization and mapping (SLAM) technology. Industry starts to deploy SLAM for indoor logistics, warehouse automation, surveillance, monitoring and inspection of various structures. On the other hand, agriculture is one of the rising markets, where monitoring the crops by MAVs will soon increase the efficiency and reduce the usage of pesticides and fertilizers. Significant media coverage is devoted to autonomous driving, where estimating the precise location of a vehicle within annotated maps helps to comply with the traffic regulations and avoid collisions. Finally, the steadily decreasing cost of the sensing and computation hardware brings SLAM technology into the households. Some examples may include the entertainment sector, with various AR and VR applications, but also mobile phone mapping applications, robotic lawn mowers and vacuum cleaners.

The following sections of this chapter introduce the concepts on which visual SLAM builds, as well as the most established and the most innovative lines of research on this subject. Special attention is given to visual SLAM in dynamic environments since the presence of dynamic elements induces common SLAM pipelines to exhibit large errors or fail. The main contributions of this thesis addressing the largely open problem of visual SLAM in highly dynamic environments are summarized in Section 1.3.

1.1 Visual Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) techniques estimate jointly a map of an unknown environment and the robot pose within such map, only from the data streams of its on-board sensors. The map allows the robot to continually localize within the same environment without accumulating drift. This is in contrast to odometry

approaches that integrate the incremental motion estimated within a local window and are unable to correct the drift when revisiting places.

Visual SLAM, where the main sensor is a camera, has received a high degree of attention and research efforts over the last years. The minimalistic solution of a monocular camera has practical advantages with respect to size, power and cost, but also several challenges such as the unobservability of the scale or state initialization. By using more complex setups, like stereo or RGB-D cameras, these issues are solved and the robustness of visual SLAM systems can be greatly improved.

Modern visual SLAM systems usually fall into two main branches, namely the feature-based and the so called direct methods. Both of them have seen tremendous improvements on accuracy, robustness and efficiency, and have gained exponential popularity over recent years. Feature-based methods rely on salient points matching and can only estimate a sparse reconstruction (Klein & Murray (2007), Mur-Artal & Tardós (2017)); and direct methods are able to estimate in principle a completely dense reconstruction by the direct minimization of the photometric error and TV regularization (Stühmer et al. (2010), Newcombe et al. (2011), Graber et al. (2011)). Some direct methods focus though on the high-gradient image areas estimating semi-dense maps (Engel et al. (2014, 2017), Forster et al. (2014)).

1.1.1 Sensors

In this section we describe the most common sensor modalities used in visual SLAM: monocular, stereo and RGB-D cameras.

Monocular camera

The main parts of a monocular camera are the camera lens and the camera's image sensor. A 3D point $\mathbf{X}_{\mathtt{C}} \in \mathbb{R}^3$ in the camera coordinate reference system $\mathtt{C}$ is projected into 2D pixel coordinates $\mathbf{x}$ with column u and row v with the projection function $\pi_m : \mathbb{R}^3 \to \mathbb{R}^2$ (Eqn. 1.1). $\pi_m^{-1} : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ is the inverse projection function that computes the 3D location $\mathbf{X}_{\mathtt{C}}$ of a 2D point $\mathbf{x}$ in the image, given its depth d (Eqn. 1.2).

$$\mathbf{x} = \pi_m(\mathbf{X}_{\mathtt{C}}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = [X, Y, Z]^T, \quad \mathbf{x} = [u, v]^T \tag{1.1}$$

$$\mathbf{X}_{\mathtt{C}} = \pi_m^{-1}(\mathbf{x}, d) = \begin{bmatrix} d\frac{u - c_x}{f_x} \\ d\frac{v - c_y}{f_y} \\ d \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = [X, Y, Z]^T, \quad \mathbf{x} = [u, v]^T \tag{1.2}$$

The camera intrinsic parameters f_x, f_y, c_x and c_y are fixed –to a certain extent– to a particular camera setup and allow a mapping between camera coordinates and pixel coordinates in the image frame. f is the camera focal length measured respectively in horizontal (f_x) and vertical (f_y) pixels, and c_x and c_y are the optical center coordinates. There might be small changes in the intrinsic parameters due to camera movement, as the camera is not perfectly rigid, or changes in the environments, *e.g.*, temperature, but these are small enough and are usually ignored in most use cases. The camera coordinates system $\mathtt{C}$ has its origin at the camera optical center. With respect to the image, the Z axis points towards the observed scene, the X axis is horizontal and points towards the right, and finally, the Y axis is vertical and points downwards. Monocular

cameras cannot observe the scale of the world and therefore only up-to-scale maps can be built with such a setup.

Stereo cameras

Stereo cameras are composed of two rigidly attached cameras. Ideally both cameras are hardware synchronized so that image capturing is triggered at the same time. Depth can be estimated from just one stereo frame by finding correspondences between left and right pixels. To this end, in addition to intrinsic calibration of both cameras, the rotation and translation between both cameras have to be calibrated. The distance between both cameras, known as the baseline b, along with focal length and image resolution will determine the depth range at which depth estimation is accurate. Depth can be accurately estimated if the parallax angle is at least $1°$, *i.e.*, if the depth is less than 55 times the stereo baseline (Paz et al. (2008)). In order to facilitate stereo matching, images are typically rectified, removing distortion and rotating them so that the epipolar lines are horizontal, *i.e.*, the correspondence of a pixel in the left image lies on the same row in the right image. This significantly reduces the stereo matching cost. The projection function for a rectified stereo camera $\pi_s : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is defined in Eqn. 4.2, and its inverse projection is defined in Eqn. 1.4. The coordinates of a pixel in a stereo camera $\mathbf{x}$ need of three parameters: the column and row on the left image (u_L and v_L), and of the column on the right image (u_R). In contrast to monocular approaches, the use of stereo cameras allows building scaled maps.

$$\mathbf{x} = \pi_s(\mathbf{X_c}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix}, \quad \mathbf{X_c} = [X, Y, Z]^T, \quad \mathbf{x} = [u_L, v_L, u_R]^T \tag{1.3}$$

$$\mathbf{X_c} = \pi_s^{-1}(\mathbf{x}) = \begin{bmatrix} b \frac{u_L - c_x}{u_L - u_R} \\ b \frac{f_x}{f_y} \frac{v_L - c_y}{u_L - u_R} \\ b f_x \frac{1}{u_L - u_R} \end{bmatrix}, \quad \mathbf{X_c} = [X, Y, Z]^T, \quad \mathbf{x} = [u_L, v_L, u_R]^T \tag{1.4}$$

RGB-D cameras

RGB-D cameras are a combination of a monocular camera and a depth sensor. Depth sensing cameras store the distance to the points they observe. They can provide a fine-grained representation of the observed objects, and tend to be precise for close distances but become inaccurate for mid-long range measurements. These cameras rely on two different working principles. The time-of-flight (ToF) cameras emit light pulses and measure the time until they are received back at the sensor. Since they normally emit and detect infrared light, they are affected by the sun radiation. However, they can still provide decent measurements in outdoor scenarios if there is no direct sunlight. Structured light cameras work by projecting a known pattern of light on the scene and inferring the depth field from the way that pattern deforms. Structured light cameras typically employ an infrared pattern of light and, hence, become completely blind under the presence of sunlight.

Knowing the intrinsics of both sensors, as well as the extrinsic parameters between them, one can obtain an approximate 1:1 correpondence between the RGB image and

the registered depth. The use of depth sensing cameras allows building scaled maps. The projection and the inverse projection functions for a RGB-D camera $\pi_d : \mathbb{R}^3 \to \mathbb{R}^3$ and $\pi_d^{-1} : \mathbb{R}^3 \to \mathbb{R}^3$ are respectively defined in Eqn. 1.5 and Eqn. 1.6. As for stereo cameras, a pixel in a RGB-D camera is defined by three parameters: its column u, its row v and its value in the depth map d.

$$\mathbf{x} = \pi_d(\mathbf{X}_{\mathtt{C}}) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ Z \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = [X, Y, Z]^T, \quad \mathbf{x} = [u, v, d]^T \tag{1.5}$$

$$\mathbf{X}_{\mathtt{C}} = \pi_d^{-1}(\mathbf{x}) = \begin{bmatrix} \frac{u - c_x}{f_x} d \\ \frac{v - c_y}{f_y} d \\ d \end{bmatrix}, \quad \mathbf{X}_{\mathtt{C}} = [X, Y, Z]^T, \quad \mathbf{x} = [u, v, d]^T \tag{1.6}$$

1.1.2 Feature-Based Methods

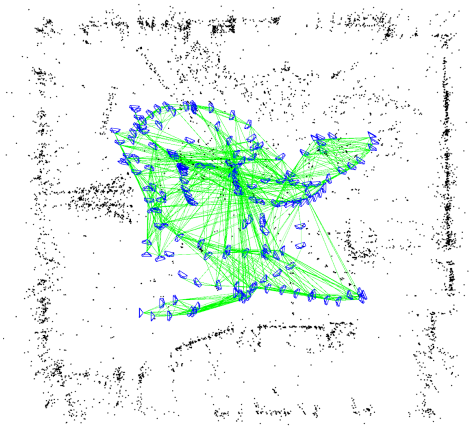
Feature-based methods extract a sparse set of key points from each image and match them across multiple frames. Camera poses and map points positions are estimated by minimizing the reprojection errors between feature pairs. The reprojection error function can be seen in Eqn. 1.7, where $\mathbf{R}^i_{\mathtt{CW}} \in \mathrm{SO}(3)$ and $\mathbf{t}^i_{\mathtt{CW}} \in \mathbb{R}^3$ represent the rotational and translational part respectively of the pose in world coordinates $\mathtt{W}$ of the camera i, and $\mathbf{X}^j_{\mathtt{W}} \in \mathbb{R}^3$ is the 3D position, referred also to the world, of the map point j, corresponding to the pixel $\mathbf{x}^j_i$ in the camera i.

$$\mathbf{e}^{i,j}_{repr} = \mathbf{x}^j_i - \pi_i(\mathbf{R}^i_{\mathtt{CW}} \mathbf{X}^j_{\mathtt{W}} + \mathbf{t}^i_{\mathtt{CW}}) \tag{1.7}$$

As modern feature descriptors are invariant to some extent to illumination and view-point changes, feature-based methods are robust to brightness inconsistencies and large view-point changes among consecutive frames. However, feature extraction and matching bring additional computational overhead, which highly limits the number of features that can be maintained in the system. As a result, the reconstructed 3D maps are rather sparse with little use for other robotic tasks other than localization.

The main limitation of these approaches is precisely that they can only exploit visual information where features, typically corners, can be extracted. Lack of texture or motion blur can make a feature-based method fail or perform very poorly. An example of a map built with such approaches can be seen in Fig. 1.1a, in contrast to one built with direct methods (Fig. 1.1b).

Among the most popular feature-based SLAM systems, one can find PTAM by Klein & Murray (2007), and ORB-SLAM by Mur-Artal et al. (2015). The former, while limited to small scale operation, provides simple but effective methods for keyframe selection and feature matching among others. They lack however of loop closing and adequate handling of occlusions, and the need of human intervention for map bootstrapping severely limits its application. Mur-Artal & Tardós (2017) designed ORB-SLAM from scratch building on top of the main ideas of PTAM, the place recognition work of Gálvez-López & Tardos (2012), and the scale-aware loop closing of Strasdat et al. (2010), achieving unprecedented accuracy and robustness in general scenarios.



(a) Sparse map built with ORB-SLAM by Mur-Artal et al. (2015).

(b) Dense map reconstructed with DTAM by Newcombe et al. (2011).

Figure 1.1: Comparison of a sparse map (a) and a dense map (b). Whereas maintaining the former map has a low cost and it has enough information for localization purposes, the latter can be exploited for many other tasks. However, keeping a large-scale dense map is computationally demanding and prohibitive in real-time applications.

1.1.3 Direct Methods

Direct methods in turn use all image pixels –dense as in Fig. 1.1b– (Newcombe et al. (2011)), all pixels with sufficiently large intensity gradient –semi-dense– (Engel et al. (2014)) or sparsely selected pixels –sparse– (Forster et al. (2014), Engel et al. (2017)), and minimize the photometric error instead of the reprojection error, as feature-based methods do. The photometric error is obtained by direct image alignment on the selected pixels, and its equation can be seen in Eqn. 1.8, where $\mathbf{I} : \mathbb{R}^2 \rightarrow \mathbb{R}^+$ is the function that returns the interpolated pixel intensity on the image for a given pixel position $\mathbf{x}$.

$$\mathbf{e}_{phot} = \mathbf{I}_{i+1}(\mathbf{x}^j_{i+1}) - \mathbf{I}_i(\pi_i(\mathbf{R}^i_{\mathtt{CW}}\mathbf{X}^j_{\mathtt{W}} + \mathbf{t}^i_{\mathtt{CW}})) \tag{1.8}$$

Camera poses and 3D map points positions are estimated by minimizing the photometric error using non-linear optimization algorithms. Since more image information can be used, direct methods are more robust in low-texture scenarios and can deliver denser 3D reconstructions than the feature-based methods. Correspondingly, due to the direct image alignment formulation, direct methods are very sensitive to non-modeled artifacts such as rolling shutter effect, camera auto exposure and gain control. More crucially, the brightness constancy assumption does not always hold in practice, which drastically reduces the performance of direct methods in environments with rapid lighting changes.

1.1.4 Bundle Adjustment

Given a set of images depicting a number of 3D points from different viewpoints, bundle adjustment can be defined as the problem of simultaneously refining the 3D coordinates describing the scene geometry, the parameters of the relative motion, and, optionally, the optical characteristics of the cameras employed to acquire the images, according to an optimality criterion involving the corresponding image projections of all

points. Its name refers to the bundles of light rays originating from each 3D feature and converging on each camera's optical center, which are adjusted optimally with respect to both the structure and viewing parameters. Bundle adjustment is by definition tolerant to missing image projections and minimizes a physically meaningful criterion. For a long time this approach was considered unaffordable for real time applications such as visual SLAM. However, nowadays we know that if an initial good estimation of the frame poses and point locations is provided, and if the optimization complexity does not grow unboundlessly, it is feasible to use bundle adjustment for visual SLAM.

The main limitation of dense and semi-dense direct approaches is the computational complexity that forbids to jointly optimize in real-time structure and cameras, which reduces the achievable accuracy of these methods. In general, sparse direct optimizations only work if the initial seed for the optimization is close to the optimal, due to the nature of the photometric error. The reprojection of a pixel has to be close to the optimal projection so that the intensity gradients on the image can guide the optimization to its true location. To mitigate this problem, direct methods use image pyramids, but still the basin of convergence is narrower than for feature-based methods. On the other hand, the convergence properties of a geometric error like the reprojection error make bundle adjustment suitable for its optimization. This is one of the main advantages of such approaches. The equation for a bundle adjustment optimization minimizing the reprojection error for m 3D points and n cameras can be found in Eqn. 1.9, where ρ is a robust cost function, such as the Huber norm, used to downweight the outliers influence, $\|\cdot\|$ is the Mahalanobis distance and Σ_i^j is the covariance matrix for the location of the keypoint $\mathbf{x}_i^j$ on the camera i. $\hat{\mathbf{R}}_{\mathtt{CW}}^i$ and $\hat{\mathbf{t}}_{\mathtt{CW}}^i$ stand for the optimized rotational and translational pose respectively of the camera i, and $\hat{\mathbf{X}}_{\mathtt{W}}^j$ represents the optimized position of the 3D point j. So that this equation has a solution, the scale of the map and a camera pose need to be fixed.

$$(\hat{\mathbf{R}}_{\mathtt{CW}}^i, \hat{\mathbf{t}}_{\mathtt{CW}}^i, \hat{\mathbf{X}}_{\mathtt{W}}^j) = \underset{\mathbf{R}_{\mathtt{WC}}^i, \mathbf{t}_{\mathtt{WC}}^i, \mathbf{X}_{\mathtt{W}}^j}{\arg\min} \sum_{i=1}^{n} \sum_{j=1}^{m} \rho(\|\mathbf{x}_i^j - \pi_i(\mathbf{R}_{\mathtt{CW}}^i \mathbf{X}_{\mathtt{W}}^j + \mathbf{t}_{\mathtt{CW}}^i)\|_{\Sigma_i^j}^2) \tag{1.9}$$

Bundle adjustment boils down to minimizing the reprojection error between image locations and predicted image points, which is expressed as the sum of squares of a large number of nonlinear functions (Triggs et al. (1999)). Thus, the minimization is achieved using nonlinear least-squares algorithms such as the standard Gauss-Newton method, and variants like Levenberg-Marquadt. Levenberg-Marquadt has proven to be one of the most successful ones due to its ease of implementation and its use of an effective damping strategy that lends it the ability to converge quickly from a wide range of initial guesses. By iteratively linearizing the function to be minimized in the neighborhood of the current estimate, the Levenberg-Marquadt algorithm involves the solution of linear systems (the so called *normal equations*). When solving the minimization problems arising in the framework of bundle adjustment, the normal equations have a sparse block structure owing to the lack of interaction among parameters for different 3D points and cameras (Triggs et al. (1999)). This can be exploited to gain tremendous computational benefits by employing a sparse variant of the Levenberg–Marquardt algorithm which explicitly takes advantage of the normal equations zeros pattern, avoiding storing and operating on zero-elements. These approaches can be seamlessly generalized to variables belonging to smooth manifolds (*e.g.*, rotations), which are of interest in robotics (Forster et al. (2016)).

1.1.5 Deep Learning Priors

Humans can understand the layout of a scene, estimate depth and detect obstacles from a single image. Many methods have been proposed in recent years to exploit the geometry cues and scene assumptions in order to build simplified 3D models. Especially, with the advent of Convolutional Neural Networks (CNNs) (Krizhevsky et al. (2012)), performance of visual understanding has been greatly increased.

On top of the two classic revisited methods –feature-based and direct–, one can find many SLAM systems that include other priors (Concha & Civera (2015), Yang & Scherer (2019), Hosseinzadeh et al. (2019)). These priors come, especially in the recent years, from deep learning and CNNs. Conventionally, dense representations of the environment are known to require a large number of parameters. However, in the recent years, the computer vision and the robotics communities have seen attempts to undermine this claim. For example, since the methods mentioned above usually perform well in environments with rich features but cannot work well in low-texture scenes as often found in corridors, Yang, Song, Kaess & Scherer (2016) leverage the Manhattan assumption using some planes in the SLAM optimization to get a better estimation. They use a CNN to generate plane landmarks in SLAM from only one image and achieve awesome dense reconstructions in low-texture scenes as the one seen in Fig. 1.2a. In a different way, Czarnowski et al. (2020) recently published a work called *DeepFactors* where learning has been leveraged to inform the factor graph on priors about the continuity of the world for the map reconstruction. They make use of a learned compact depth map representation that is further used within the standard SLAM factor graph, and achieve completely dense representations (Fig. 1.2b).

Apart from planes and depth inference, semantic segmentation also offers a higher level scene geometric information about objects that can be exploited within visual SLAM (Bescos et al. (2018), Xu et al. (2019), Hosseinzadeh et al. (2019)). Semantic information can allow machines to reason categorically, *i.e.*, to build associations across distinct object instances. Examples of maps with a richer semantics representation can be seen in Figs. 1.2c and 1.2d. This semantic information can allow to reason about the dynamics of individual objects rather than separate 3D points landmarks.

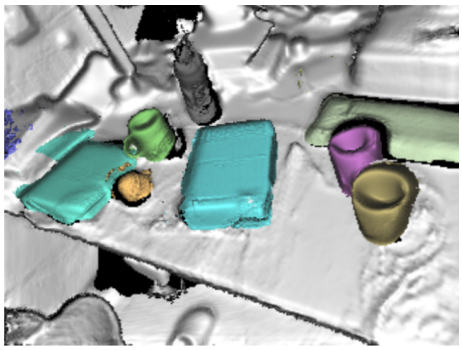
1.1.6 Front End and Back End

The architecture of a SLAM system –either feature-based or direct– commonly includes two main components: the front end and the back end (see Fig. 1.3). The front end abstracts sensor data into models that are amenable for estimation, while the back end performs inference on the abstracted data produced by the front end.

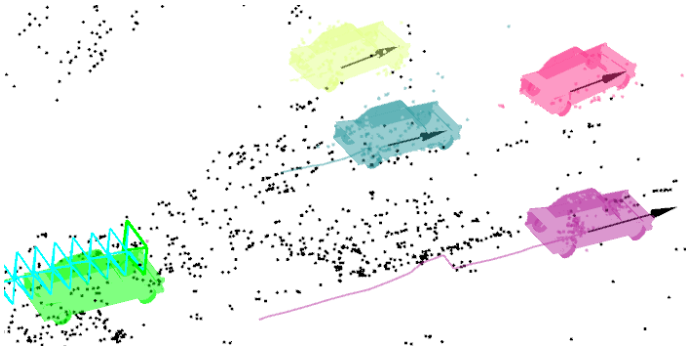
In visual SLAM the front end is in charge of extracting pixels information and associating pixel locations to specific landmarks or 3D points in the environment. This data association module in the front end includes a short-term data association component and a long-term one. Whereas short-term data association is responsible for associating corresponding features in consecutive camera measurements, the long-term data association connects new measurements to older landmarks. Finally, the front end also provides an initial guess for the variables in the non-linear optimization of the back end. Bundle adjustment occurs in the back end, where the abstracted data produced in the front end is optimized to obtain the refined camera poses and the refined map structure. The back end usually feeds back information to the front end to support loop closure detection and validation.

(a) Dense map built with the system Pop-Up SLAM by Yang, Song, Kaess & Scherer (2016).

(b) Dense map reconstructed with DeepFactors by Czarnowski et al. (2020).



(c) Dense map built with MID-Fusion by Xu et al. (2019).



(d) Sparse spatial-temporal dynamic map built with our system DynaSLAM II (Chapter 4).

Figure 1.2: The authors of (a) use a CNN to detect planes that are included in the SLAM formulation as a prior. The authors of (b) make use of a learned compact depth map to achieve dense representations from a monocular camera. The use of CNNs allow to integrate semantics into SLAM and to reason about higher level entities (c), (d).

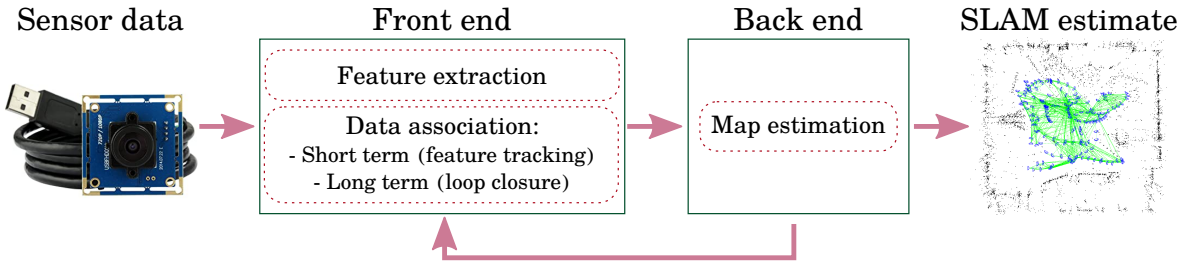


Figure 1.3: Front end and back end in a SLAM pipeline. The sensor data is associated in the front end and the back end is in charge of keeping a long-term stable map.

1.2 Visual SLAM in Dynamic Environments

Visual SLAM in dynamic environments is a largely open problem. This section aims to provide the reader with an understanding of the challenges in front of us in this area.

The research community has addressed SLAM from many different angles. However, the vast majority of the approaches and datasets assume a static environment. Although the static assumption holds for some robotic applications, it limits the applicability of visual SLAM in many relevant cases, such as intelligent autonomous systems

operating in populated real-world environments over long periods of time. None of the methods mentioned above –both feature-based and direct methods–, considered the state of the art –ORB-SLAM (Mur-Artal & Tardós (2017)), DSO (Engel et al. (2017)) and SVO (Forster et al. (2014))–, address the very common problem of dynamic objects in the scene, *e.g.*, people walking, bicycles or cars driving around, *etc.* They are not specifically designed to handle dynamics but they can –up to a certain limit– be robust to them through heuristics by focusing on having a good estimate of the trajectory. That is, they can only manage small fractions of dynamic content by classifying it as outlier to the static model. These decisions to handle dynamics can be done either in the system front end by rejecting outliers within the data association step, or with robust cost functions in the back end. On the one hand, was the majority of the scene static, these heuristics would help maintain a good estimate of the camera trajectory. On the other hand, was an important part of the scene dynamic, these heuristics could end up rejecting the static scene information and would compute the trajectory of the observed dynamic objects as the ego motion of the camera.

An example of the previous explained behavior can be seen in the following video `https://youtu.be/VZ67vBkwI3Q`. This video shows how the state-of-the-art SLAM system ORB-SLAM (RGB-D) fails at both tracking and mapping when dynamic objects become relevant within the observed scene. Since one of the two moving people within the scene is wearing a high-texture shirt, there is a high density of features detected on this dynamic object, and therefore the algorithm computes the motion of such features cluster as the camera ego motion. The map also gets corrupted with 3D points belonging to this moving person. In such scenes, robust cost functions and heuristics are not enough, and it is of crucial importance to detect which image regions belong to the static scene and which ones do not. Another failure example of the same system can be seen in the video `https://youtu.be/8uOZ0fZRTx0`.

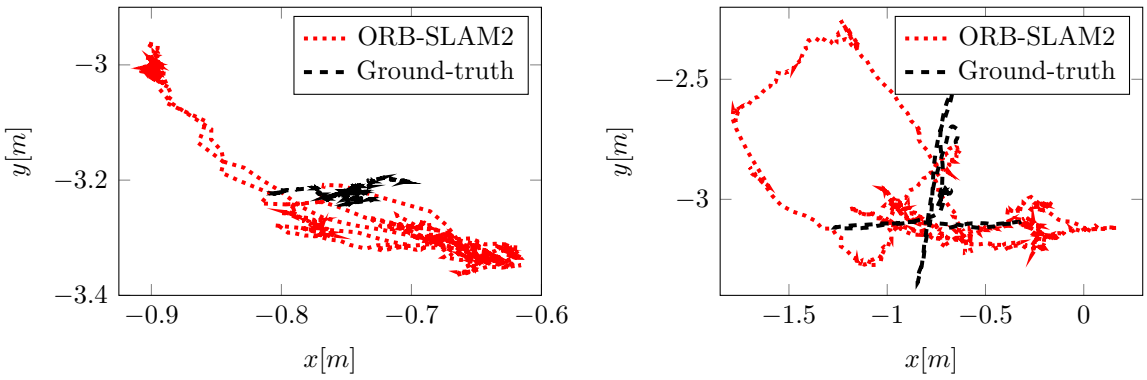
The synergy of tracking, mapping and place recognition is essential for the successful deployment of visual SLAM (Cadena et al. (2016)). Now, we would like to take a closer look at those three building blocks to better understand the challenges in front of us.

1.2.1 Camera Tracking

The first building block empowering applications of vision-based mapping and localization is a precise odometry estimation. Certain scenarios require camera tracking precision in the order of centimeters or millimeters, *e.g.*, when interacting with industrial environments or to guarantee a fully immersive experience to the user in AR/VR.

When dynamic objects become an important part of the scene, visual SLAM systems that do not specifically address dynamic content tend to compute the dynamic objects motion as camera ego motion. That is, the static scene becomes the spurious data of the model used by the SLAM system (Bescos et al. (2018)). Not handling dynamics can lead to a poor sensor tracking that might lead to catastrophic consequences: a bad camera tracking in AR/VR applications often causes feelings of nausea and disorientation in the user, or worse, in autonomous driving, a bad camera tracking can lead to traffic violations and collisions. An example of this poor tracking behaviour can be seen in Figs. 1.4a and 1.4b. In both figures, the black dashed line (- - -) represents the ground-truth camera trajectory, and the dotted red line (···) is the camera trajectory estimated by ORB-SLAM2 (RGB-D). These examples are the estimated trajectories for the TUM RGB-D Dataset (Sturm et al. (2012)) sequences $fr3/walking\_static$ and

$fr3/walking\_xyz$, depicted in the previously mentioned videos.



(a) Sequence $fr3/walking\_static$.

(b) Sequence $fr3/walking\_xyz$.

Figure 1.4: Examples of camera tracking failure in presence of dynamic objects. The trajectories have been estimated by ORB-SLAM2 (Mur-Artal & Tardós (2017)) in the TUM RGB-D dataset (Sturm et al. (2012)).

1.2.2 Mapping

The second building block is the mapping capability. A map allows to continually localize in the same environment without accumulating drift, in contrast to odometry approaches where incremental motion is integrated over time. The main purpose of a map for visual SLAM is to localize the camera in it (Cadena et al. (2016)). Therefore when the robot is traversing an already mapped region, it should not duplicate the map but localize using the already existing map. Whereas the process of mapping and localization is reliable in controlled spaces, it still remains an open challenge in large-scale scenarios that require life-long map maintenance. It is key not to bloat the size of visual maps unnecessarily, as this could adversely affect the cost and computing power required.

We want to construct maps that remain compact without redundant elements or dynamic objects that are not persistent in the environment. A map containing information about dynamic objects might become useless for future reuse. Only the information belonging to objects that remain stable in the long term (buildings, sidewalks, *etc.*) should be included in the visual map. An example of a dense 3D map which is corrupted by dynamic objects can be found in Fig. 1.5.

1.2.3 Place Recognition

The third building block is the place recognition capability. Recognizing a place and matching it to the prior map has one fundamental application in visual SLAM: as the local odometry estimates drift, at some point it does not provide anymore a reliable information about the agent's pose in the environment. One way to correct it is to recognize the place in a prior map or in a map being currently constructed. Recognizing an already visited place adds additional constraints and thus reduces the error.

Place recognition can suffer from dynamic objects in two different manners. On the one hand, two images of the same place with a very different setup of dynamic objects

(a) Sequence sample images depicting dynamic objects.

(b) Dense reconstruction corrupted with dynamic objects.

Figure 1.5: Example of a corrupted map in presence of dynamic objects (—). This is the sequence 04 from the KITTI odometry dataset (Geiger et al. (2013)). The map has been created with the multi-view software colmap (Schönberger, Zheng, Frahm & Pollefeys (2016)) and the ground-truth camera poses.

could be incorrectly tagged as a different place. On the other hand, two images of different places with the same dynamic objects setup could be wrongly matched as the same place. False positive loop closures due to incorrect place recognition can have a catastrophic effect and constitute an open challenge. The use of robust loop closing techniques (Sünderhauf & Protzel (2012), Latif et al. (2013), Lajoie et al. (2019), Yang et al. (2020)) can help to mitigate false loop detections. Consequently, only the features belonging to elements that remain stable in the long term (buildings, sidewalks, *etc.*) would benefit place recognition.

We depict in Fig. 1.6 an example of two different images from the same place with different viewpoints and weathers, and also with dynamic and structural changes. The state-of-the-art place recognition algorithms, both the learning-based ones (Arandjelović et al. (2016)) and mainly the traditional ones (Gálvez-López & Tardos (2012)), struggle to work under large amounts of clutter, *e.g.*, people, cars. Even though place recognition has made great advances in the last few decades, *e.g.*, on the principles of how animals recognize and remember places, and the relationship between places from a neuroscience perspective (Lowry et al. (2015)), we are still a long way from a robust long-term visual place-recognition system that can be applied well to a variety of scenarios of real-world places.

(a) (b)

Figure 1.6: (a) and (b) have been taken from a nearby position depicting the same scene with different view point and weather, and with dynamic and structural changes. These changes might make place recognition algorithms fail if not specifically handled.

1.2.4 Dynamic Objects

In this subsection we want to highlight that there are two different types of dynamic objects with respect to their nature and their influence on visual SLAM systems:

1. The former type consists of objects that inherently possess the capability to move either artificially or naturally. People, animals and means of transport are within such group. We name these objects *a priori* dynamic. While the visual SLAM sensor is observing them, they can either move (*e.g.*, person walking) or remain static (*e.g.*, person laying down). If such objects are moving they would certainly damage the camera trajectory estimation. They can be detected with a multi-view geometry approach. In contrast, if they remain static they would be helpful for the camera trajectory estimate but would corrupt the map with long-term spurious information. These can only be detected via learning-based approaches. For example, the parked cars on Fig. 1.7a would not be detected if we used an algorithm capable of detecting scene motion. However, it would be straight forward to detect them if a learning-based algorithm was to be used. Moreover, the two people on Fig. 1.7b could be detected by using either a multi-view geometry or a learning-based approach because they are *a priori* dynamic and they are in motion.

2. The latter type consists though of objects that are moving while the visual SLAM sensor is observing them. Their motion can be exploited and detected mainly by multi-view geometry approaches. Examples of these objects could be a ball being thrown in the air, driving cars, or furniture being moved by people (Fig. 1.7b). These objects would corrupt both the camera trajectory and the map estimates, and cannot be detected on a single frame basis.

(a) *A priori* dynamic objects.

(b) Moving objects.

Figure 1.7: Whereas geometry-based approaches are used to detect scene movement, semantic information can be utilized to detect *a priori* dynamic classes.

1.2.5 Challenges

Overall, the visual mapping and localization technology is getting more and more mature and becoming widespread in industrial solutions, service robotics and consumer devices. It is a key to guarantee spatial awareness with a limited cost, weight and computational power. We want to make it more reliable and robust in real-life environments, including crowded indoor and outdoor spaces. That is, we want our algorithms to work everywhere, without limiting them to specifically pre-configured static environments. Dealing with dynamic objects in visual SLAM reveals several challenges including: (i) How to detect such dynamic objects in the images to prevent the tracking algorithm from using matches that belong to moving objects, to prevent the mapping algorithm from including both *a priori* dynamic and moving objects as part of the 3D map, and to let the loop closing algorithm use only the scene information that is static in the long term. (ii) How to complete the 3D map section that is temporally occluded by dynamic objects. (iii) How to estimate the trajectory of the dynamic objects within the scene to help with decision making and scene understanding.

Many applications would greatly benefit from progress along these lines. Among others, augmented reality, autonomous vehicles, and medical imaging. All of them could for instance safely reuse maps from previous runs. Detecting and dealing with dynamic objects is a requisite to estimate stable maps, useful for long-term applications. If the dynamic content is not detected, it becomes part of the 3D map, complicating its usability for tracking or relocation purposes. Furthermore, in augmented and virtual reality applications, dynamic objects need to be explicitly tracked to enable interactions of virtual objects with moving instances in the real world. In autonomous driving scenarios, a car should not only localize itself but also reliably sense other moving cars to act consequently and avoid catastrophic collisions.

1.3 Contributions

The overarching goal of this doctoral thesis is to advance the current state of the art in visual SLAM with the particular following objectives:

1. **Detection:** While the current localization and mapping algorithms perform rea-

sonably well in controlled and static environments, they struggle in the presence of noise or dynamic objects. The detection of the dynamic objects within the images is the first step towards a robust SLAM. We aim to detect dynamic and temporary objects in the environment. We want to construct maps that only contain persistent elements of the robot's surroundings, but reject everything that is not deemed stable. Consequently, we will make the camera tracking and the place recognition process more robust to spurious and dynamic objects.

2. **Inpainting:** Detecting dynamic objects within the image streams usually leads to a robust camera tracking and the creation of stable maps. However, these maps might present discontinuities and empty gaps due to dynamic objects permanently occluding a scene region. Also, pose estimation could be degraded and drifting could increase if features do not follow a uniform distribution in the image space. We aim to perform image *inpainting* of the plausible static scene behind the dynamic objects. This allows us to deliver complete maps with a plausible scene representation. Place recognition and camera tracking can also become more robust with these advances in image inpainting.

3. **Multi-Object Tracking:** Whereas building stable maps, or having a robust camera tracking and robust place recognition might be enough for some applications, others might require information about the poses of other dynamic agents within the scene. Understanding surrounding dynamic objects is of crucial importance for the frontier requirements of emerging applications within AR/VR or autonomous things navigation. We aim to deliver a spatio-temporal map with information from self and surroundings for outdoor and indoor applications.

Next we present this thesis' contributions leading towards the aforementioned goals.

1.3.1 Detection

The goal of this part of the thesis is to enable the detection of dynamic objects either as a standalone or within a visual SLAM framework. In the latter case, we aim to minimize the error of the camera trajectory estimation and reduce the presence of spurious and non-stable data within the created maps, which is a must for long-term applications in real-world environments. In the effort to build a versatile framework, we aim to develop an algorithm capable of detecting dynamic objects that suits the most common visual sensors: monocular, stereo and RGB-D configurations.

We therefore propose a novel approach that leverages the use of semantic segmentation from deep learning and the use of multi-view geometry methods to detect both *a priori* dynamic and moving objects (Bescos et al. (2018)). Our experiments show that it is possible to achieve a SLAM accuracy in highly dynamic scenes comparable to the one accomplished in static environments, as well as to build a 3D map containing only the static and stable environment structure. Even though it is of common practice in the robotics community to assume that certain semantic classes are by definition dynamic, we demonstrate that it is possible to acquire this knowledge in a self-supervised way (Zhou, Bescos, Dymczyk, Pfeiffer, Neira & Siegwart (2018)). These methods and their evaluation are presented in detail in Chapter 2.

In the benefit of the robotics and mainly the SLAM community, we have developed a system, a *de facto* side-product of the research efforts, that provides the user with the benefits of a visual SLAM system robust in dynamic environments.

1.3.2 Inpainting

The second part of the thesis addresses the problem of inpainting with plausible imagery the image regions where dynamic objects have been previously detected. The general objective is to improve vision-based localization and mapping tasks in dynamic environments, where the presence –or absence– of different dynamic objects in different moments makes these tasks less robust. To the best of our knowledge, this is the first attempt to estimate the static world directly from the sensor data and then feed this static world to a classic SLAM pipeline. This approach is based on the fact that the static-world assumption is ideal for most SLAM systems, since it has a minimum number of outliers for tracking, mapping and place recognition.

We present a data-driven approach to obtain the static image of a scene, eliminating dynamic objects that might have been present at the time of traversing the scene with a camera (Bescos et al. (2019)). To meet this goal, we introduce an end-to-end deep learning framework to turn images of an urban environment that include dynamic content, such as vehicles or pedestrians, into realistic static frames suitable for localization and mapping. This is approached with a generative adversarial model that, taking as input the original dynamic image and its dynamic/static binary mask, is capable of generating the final static image. This framework makes use of two new losses, one based on image steganalysis techniques, useful to improve the inpainting quality, and another one based on ORB features, designed to enhance feature matching between real and hallucinated image regions. To validate this approach, we perform an extensive evaluation on different tasks that are affected by dynamic entities, *i.e.,* visual odometry, place recognition and multi-view stereo, with the hallucinated images. This method and its evaluation is presented in detail in Chapter 3.

1.3.3 Multi-Object Tracking

The last part of the thesis not only focuses on enabling a robust camera tracking and on building stable maps, but also focuses on tracking the poses of all dynamic agents that are present in the scenario. We aim to develop a conjoint framework for both SLAM and multi-object tracking, by minimizing the error of the camera trajectory estimation and that of the surrounding agents. This paves the way for manifold applications including autonomous driving navigation as well as augmented and virtual reality among others.

We therefore present a feature-based SLAM framework for stereo and RGB-D sensor configurations whose front end and back end are tailored for multi-object tracking. Given the layer of complexity that object tracking adds to the problem of SLAM, we address it by paying special attention to the number of parameters involved in order to maintain a real-time performance. We demonstrate that these two tasks are highly beneficial to each other, contradicting the usual approaches in the current literature. To validate this approach, we perform an extensive evaluation on both ego motion and object tracking in multiple environments –indoors and outdoors–, demonstrating that it is possible to obtain a state-of-the-art performance without assuming scene priors. The explanation of this method and its evaluation is presented in Chapter 4 below.

1.4 Organization of the Document

The following three chapters relate the research conducted in the direction of the main contributions of this thesis regarding detection of dynamic objects (Chapter 2), inpainting of the static background occluded by dynamic objects (Chapter 3) and multi-object tracking (Chapter 4). Each of these chapters intends to be self-explanatory and contains its specific related work as well as its respective conclusions and discussions.

In Chapter 5 we seek to outline the general conclusions of the thesis, as well as our vision of its future directions. A summary of the results can be found in Chapter 6.

Chapter 2

Detection of Dynamic Objects

The vast majority of SLAM approaches and datasets assume a static environment. They can only manage small fractions of dynamic content by classifying them as outliers to such static model. Although the static assumption holds for some robotic applications, it limits the applicability of visual SLAM in many relevant cases, such as intelligent autonomous systems operating in populated real-world environments. It is seldom the case that robots operate in strictly static environments.

Image-based detection of dynamic objects is the first step towards dynamic Visual SLAM. We hereby propose two different methods to detect the two existing types of dynamic objects: the former is based on multi-view geometry and is useful to detect moving objects –see Section 2.2– and the latter is based on deep learning and can detect *a priori* dynamic objects –see Section 2.3–. We have integrated these two methods into the state-of-the-art feature-based SLAM system ORB-SLAM to render it robust against moving agents in the scene. So, as a by-product of this research we have developed the system DynaSLAM (Bescos et al. (2018)). It is an open-source project available at `https://github.com/BertaBescos/DynaSLAM/`. A study of the related work as well as the contributions of this thesis regarding the detection of dynamic content are described in the following sections.

2.1 Related Work

The straightforward approach to dealing with dynamic objects in SLAM systems is to classify them as spurious data that is neither mapped nor used for camera tracking. The most typical outlier rejection algorithms are RANSAC, *e.g.*, in ORB-SLAM by Mur-Artal et al. (2015), and robust cost functions, *e.g.*, in PTAM by Klein & Murray (2007). These systems reject the dynamic content in an effort to minimize the error of the camera trajectory estimation. Whereas these methods work in quasi-static scenarios, they fail when the scene becomes highly dynamic. In such cases, the need arises to design algorithms specifically aimed at dynamic environments:

- Wangsiripitak & Murray (2009) make use of a 3D object tracker in parallel with a monocular SLAM implementation. The SLAM process provides the object tracker with information to register objects to the map's frame, and the object tracker allows the marking of features on objects. Similarly, Riazuelo et al. (2017) deal with human activity by detecting and tracking people.

- Alcantarilla et al. (2012) detect moving objects by means of a dense scene flow representation with stereo cameras.
- Tan et al. (2013) detect changes that take place in the scene by projecting the previous frames key points into the current frame for appearance and structure validation. If the key points have suffered a significant change, they would be considered as dynamic.
- Wang & Huang (2014) segment the dynamic objects in the scene using RGB optical flow. Such segments are further removed from the direct cost function.
- Kim & Kim (2016) propose to obtain the static parts of the scene by computing the difference between consecutive depth images projected over the same plane.
- Sun et al. (2017) calculate the difference in intensity between consecutive RGB images. Pixel classification is later done by combining this difference image with the segmentation of the quantized depth image.
- More recently, the work of Li & Lee (2017) uses depth edges points, which have an associated weight indicating its probability of belonging to a dynamic object. These weights are calculated from the euclidean distance between the different 3D edge map projections.
- Scona et al. (2018) propose StaticFusion, where they simultaneously estimate the camera motion as well as a probabilistic static/dynamic segmentation of the current RGB-D image pair. This segmentation is then used for weighted dense RGB-D fusion to estimate a 3D model of only the static parts of the environment.
- Rosinol et al. (2020) proposed this past year to use an IMU to filter out moving objects within their impressive work 3D Dynamic Scene Graphs.

All previous methods –both feature-based and direct ones– that map the static scene parts only from the information contained in the sequence (Klein & Murray (2007), Mur-Artal et al. (2015), Alcantarilla et al. (2012), Tan et al. (2013), Wang & Huang (2014), Kim & Kim (2016), Sun et al. (2017), Li & Lee (2017), Scona et al. (2018), Rosinol et al. (2020)), fail to estimate lifelong models when an *a priori* dynamic object remains static, *e.g.*, parked cars in outdoors scenes or people sitting in indoors scenarios. On the other hand, Wangsiripitak & Murray (2009) and Riazuelo et al. (2017) would detect those *a priori* dynamic objects, but would fail to detect changes produced by static objects, *e.g.*, a chair a person is pushing, or a ball that someone has thrown. That is, the former approach uniquely succeeds in detecting moving objects, and the second one exclusively succeeds in detecting several movable or *a priori* dynamic objects.

We hereby propose a SLAM system for monocular, stereo and RGB-D setups that combines multi-view geometry and deep learning in order to address both previous situations involving dynamic objects, resulting in a highly robust and accurate system.

2.1.1 Unsupervised Detection

A standard approach to detect the known dynamic content in the scene is the use of a CNN for pixel-wise semantic segmentation, like Mask R-CNN (He et al. (2017)) or SegNet (Badrinarayanan et al. (2017)). Such systems succeed in detecting objects

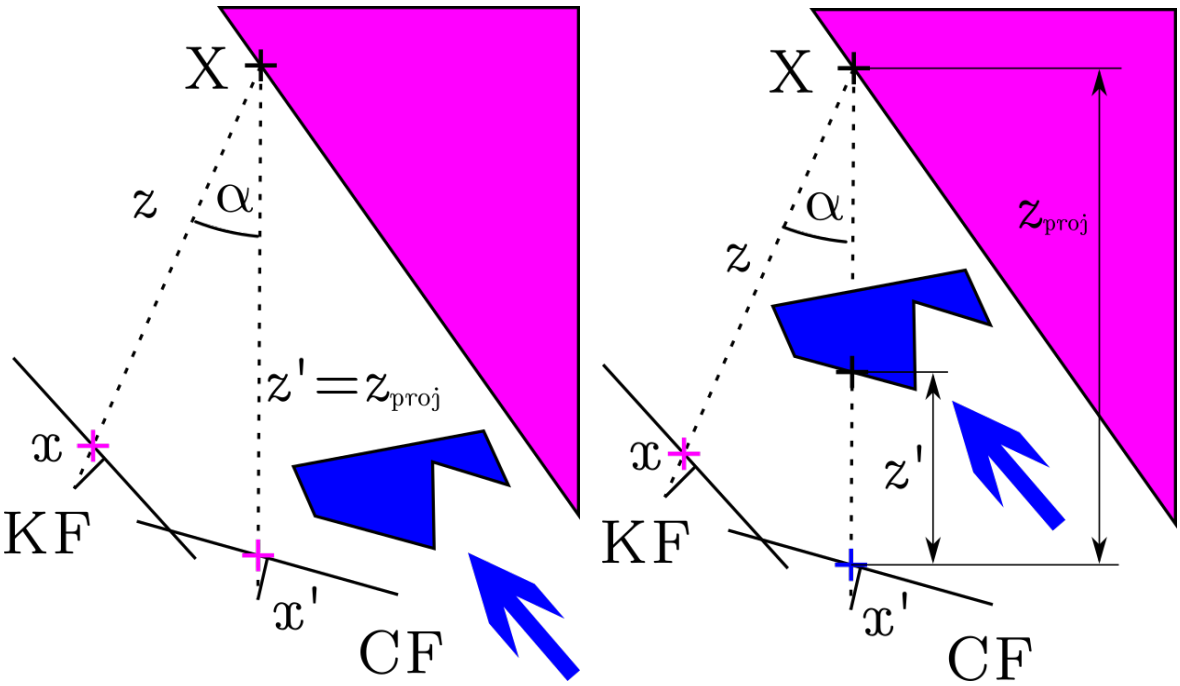
that are known to move, *i.e.*, people, vehicles, *etc.*, but fail to detect new dynamic classes. By contrast, Guizilini & Ramos (2015) apply a two-level classification mechanism (RANSAC, Gaussian Process) to detect and further learn the dynamic regions in images. Recently, the work of Barnes et al. (2018) has shown how to train a pixel-level segmentation classifier to identify dynamic and static regions in the images. They leverage large-scale offline mapping approaches with LiDAR to obtain ephemerality masks for images. Similarly, Ambrus et al. (2016) make no prior assumptions on what is dynamic and static. They model the static part of the environment by focusing on improving the accuracy and quality of the segmented dynamic objects over long periods of time. They address the issue of adapting the static structure over time and incorporating new elements, for which they train and use a classifier whose output gives an indication of the dynamic nature of the segmented elements. Other works also use machine learning (Hartmann et al. (2014)) and deep learning (Dymczyk et al. (2016)) to deal with dynamic objects. These approaches are capable of detecting new dynamic objects, on a single-view basis, yet require a specific and expensive sensor suite, a significant amount of calculations and multiple mapping sessions.

In contrast to those approaches, we propose a system that relies on single RGB frames to identify dynamic objects and train a classifier to reject spurious data. Our model learns to segment dynamic parts of the image in a semi-supervised way, without being restricted to certain semantic object classes. We hypothesize that our approach will be able to learn general dynamic objects, even if no specific semantic labels are present for training.

2.2 Detection of Moving Objects

We propose to detect scene changes with a RGB-D camera by projecting the key points from previous frames into the current frame for appearance and structure validation. That is, for each input frame, we select the previous key frames that have the highest overlap. This is done by taking into account both the distance and the rotation between the new frame and each of the key frames, similarly to Tan et al. (2013). Assuming that the poses of the previous key frames and of the current frame are known, one can then compute the projection of each key point $\mathbf{x} \in \mathbb{R}^3$ of the previous key frames into the current frame, obtaining the key point $\mathbf{x}' \in \mathbb{R}^3$. This is done with the projection and inverse projection functions for a RGB-D camera in Eqns. 1.5 and 1.6. Our movement detection proposal builds on the following idea: was the observed scene static, $\mathbf{x}$ and $\mathbf{x}'$ would depict the same 3D point $\mathbf{X} \in \mathbb{R}^3$ and would have a similar 2D appearance. However, if the scene shows dynamic content, $\mathbf{x}$ and $\mathbf{x}'$ would represent a different 3D map point and would also probably look different.

This idea is shown in Fig. 2.1. In this illustration, the magenta triangle represents the static scene and the blue polygon embodies an object in motion. On the one hand, in Fig. 2.1a the key points $\mathbf{x}$ and $\mathbf{x}'$ have the same appearance (magenta) and the projected depth $z_{proj} \in \mathbb{R}^+$ of the 3D point $\mathbf{X}$ is similar to the depth measurement $z' \in \mathbb{R}^+$ of the pixel $\mathbf{x}$ in the current frame (CF). The new key point $\mathbf{x}'$ represents therefore the static scene structure. On the other hand, in Fig. 2.1b the key points $\mathbf{x}$ and $\mathbf{x}'$ have a different appearance (magenta and blue) and the projected depth z_{proj} is not similar to the corresponding depth measurement z'. In this case, the key point $\mathbf{x}'$ depicts a 3D map point belonging to an object in motion.



(a) The key point $\mathbf{x}'$ belongs to a static object. (b) The key point $\mathbf{x}'$ belongs to a dynamic object.

Figure 2.1: The key point $\mathbf{x}$ from the key frame (KF) is projected into the current frame (CF) using its depth and camera pose, resulting in the key point $\mathbf{x}'$. The depth and the appearance of $\mathbf{x}$ and $\mathbf{x}'$ can be compared to detect dynamic instances.

Following this approach, we need to decide how to best describe the two points $\mathbf{x}$ and $\mathbf{x}'$ so that dynamic objects are correctly detected. Image descriptors such as ORB (Rublee et al. (2011)), BRISK (Leutenegger et al. (2011)) and FREAK (Alahi et al. (2012)) describe an image patch with a binary string, providing a high invariance to illumination and rotation, as well as a real-time operability. We then compute the descriptors of the key points $\mathbf{x}$ and of their projections on the current frame $\mathbf{x}'$, $\mathbf{B}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x}')$. The distance d between such descriptors can be calculated with the following equation:

$$d = \min_{\mathbf{r}} H(\mathbf{B}(\mathbf{x}), \mathbf{B}(\mathbf{x}' + \mathbf{r})), \tag{2.1}$$

where H stands for the Hamming distance and $\mathbf{r} \in \mathbb{R}^2$ is a two-dimensional translational vector to account for the reprojection error of $\mathbf{x}'$. If this error is below a threshold $\tau_{\mathbf{B}}$, the key points $\mathbf{x}$ and $\mathbf{x}'$ belong to the same static object. Otherwise, if the error is above this threshold, either the key point $\mathbf{x}$, $\mathbf{x}'$ or both belong to an object in motion.

Also according to this concept, we have considered to use the depth measurement as the key point "descriptor". That is, as a result of the projection of the 3D point $\mathbf{X}$ in the current frame, we know the approximate depth z_{proj} that the point $\mathbf{x}'$ should have were the scene static. If the current depth measurement $z' = z(\mathbf{x}')$ moves away from the depth projection more than a threshold τ_z, then either one or the two key points are the representation of a moving object. In this case, Eqn. 2.1 can be rewritten as:

$$d = \min_{\mathbf{r}}(z_{proj} - z(\mathbf{x}' + \mathbf{r})), \tag{2.2}$$

In theory, both approaches –the one based on binary descriptors and the one based

on depth measurements– could be used jointly because they detect different types of scene movement. The former detects appearance changes whereas the latter detects depth changes. For example, on the one hand, if in the illustration of Fig. 2.1 both the static and dynamic object were magenta, only the depth-based approach would succeed in detecting the motion of the small object. On the other hand, if an object moves in a plane parallel to that of the camera, the depth measurements would be consistently similar, but the appearance may allow to detect this object's motion.

We have tested the behaviour of all the above-mentioned descriptors to choose the best performing one. For this evaluation, we have manually labeled the pixel-wise segmentation of the dynamic instances in twenty images from different sequences from the TUM RGB-D dataset (Sturm et al. (2012)). An example of the images from this dataset can be found in Fig. 2.6a. This has allowed us to compute the precision-recall curves for these twenty images by varying the values of the thresholds $\tau_{\mathbf{B}}$ and τ_z respectively. Examples of these curves are depicted in Fig. 2.2. According to these curves, utilizing the depth of the key points to detect dynamic instances is the most suitable configuration.

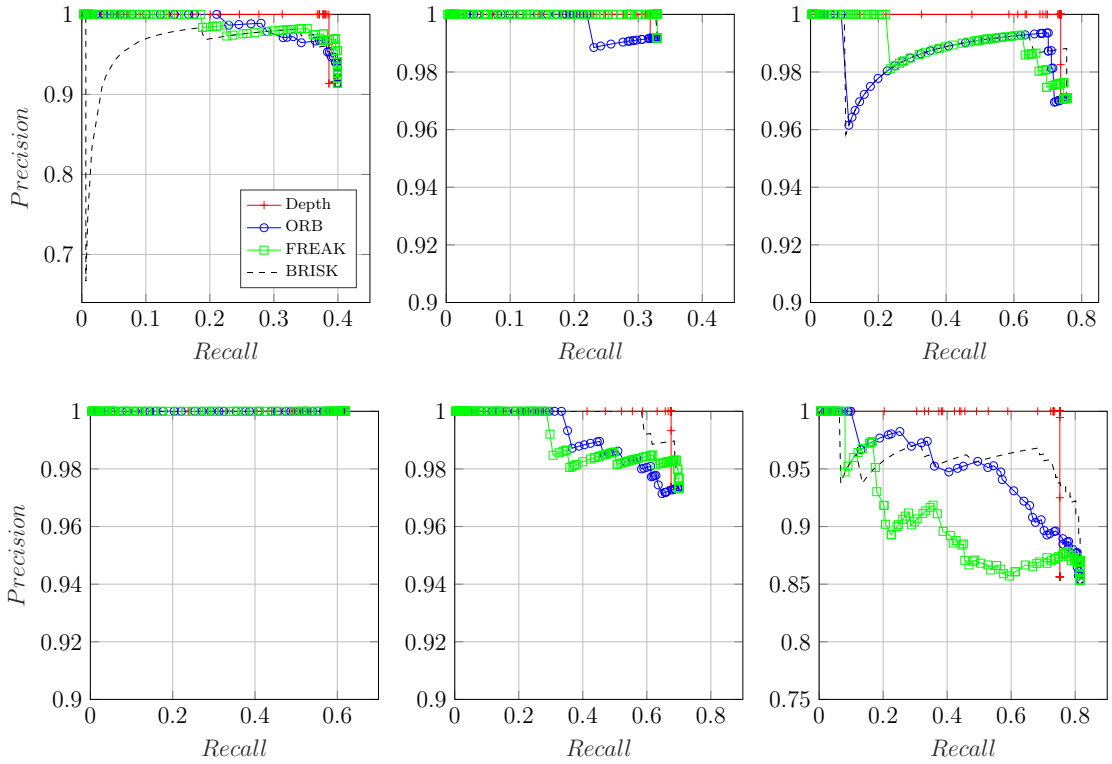
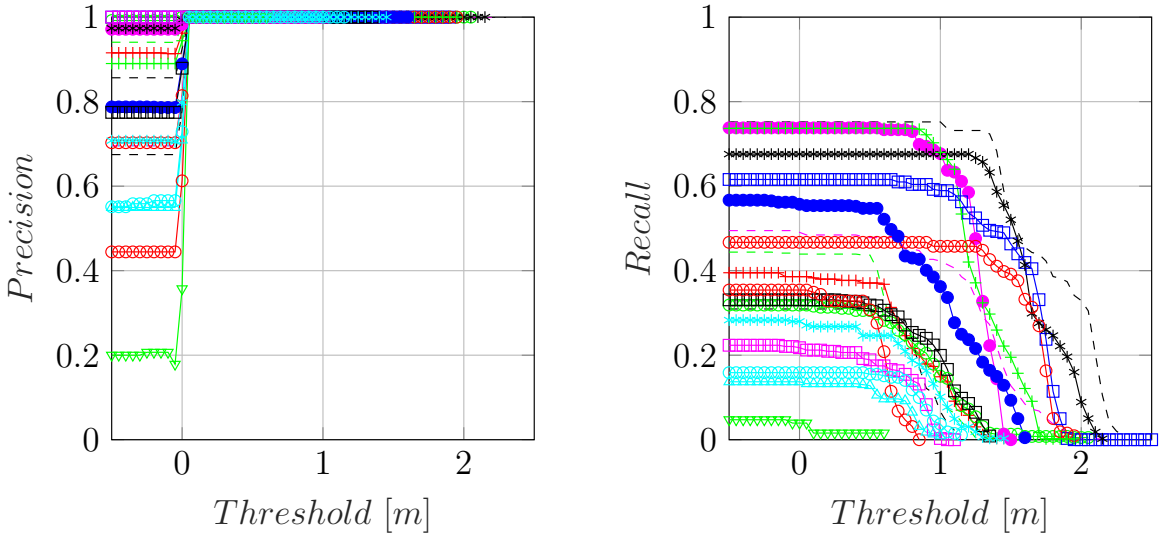


Figure 2.2: We report the precision recall curves for six images with dynamic content that have been manually labeled (TUM dataset). The performance of four descriptors has been analyzed, concluding that the use of depth is the most accurate one.

In order to set the threshold τ_z, we have also made use of these twenty labeled images with dynamic objects. The precision-recall curves of the proposed method have been evaluated for different values of this threshold (Fig. 2.3). In light of these results, we can conclude that the equality $\tau_z = 0.4m$ is a reasonable choice, providing a good compromise between precision and recall.

An issue that arises when RGB and depth images are exploited together is that of registering them properly. Since RGB and depth images are captured by different



(a) Precision - threshold curves for 20 examples.

(b) Recall - threshold curves for 20 examples.

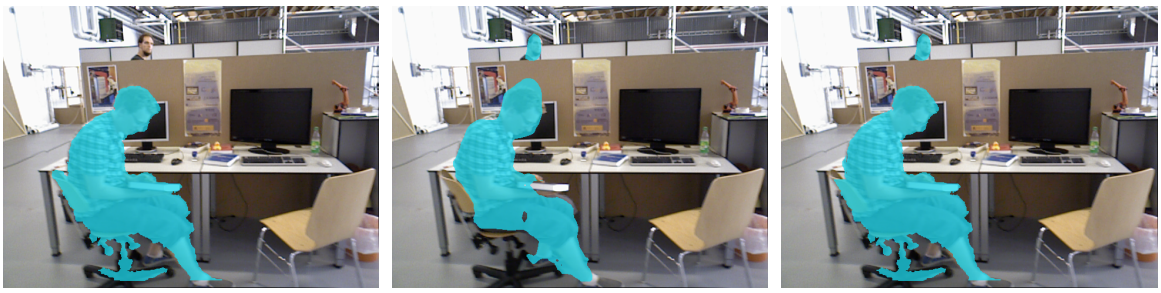
Figure 2.3: For the choice of the detection threshold τ_z, we have manually labeled all the dynamic instances in twenty images from the TUM dataset and computed the curves (a) and (b). In the view of these results, τ_z has been chosen to be 0.4 m.

lenses, there is no direct correspondence between their pixels. Often there will be points in the scene that are visible from the RGB lens but not from the infrared lens, and vice versa. In addition, this issue is enhanced if both cameras are not fully synced. The effect of this desynchronization could be considered as negligible in static scenarios, but it becomes relevant when dealing with a highly dynamic scene. A pixel in the RGB image may belong to a dynamic object, and in its corresponding depth image it may belong to an object in the background of the static scene, or vice versa.

To avoid dynamic misclassifications, key points that lay near depth discontinuities are discarded. Also, to render this method more robust against occlusions, we have computed the parallax angle α of the projection of all the key points from the previous key frames. If this angle is big enough, it becomes hard to tell if its projected depth has changed because of a dynamic object's occlusion or because of the point of view scene occlusion. In such cases, it is safer to estimate such point as static than as dynamic.

The proposed method can render feature-based SLAM systems robust against highly dynamic scenes at a very low cost. However, dense methods could not benefit from this approach since not all the pixels have a motion classification. Only a sparse set of pixels are classified as either moving or stationary. To classify all the pixels belonging to dynamic instances, we propose to grow the region in the depth image around the classified dynamic pixels (Gerlach et al. (2014)). Even though the traditional region-growing algorithm is time consuming since it belongs to sequential process, it can be decomposed for the purpose of utilizing parallel computation (Yau et al. (2013)), which allows us to quickly segment dynamic instances on a GPU. An example of a RGB frame and its corresponding dynamic segmentation computed with this multi-view geometry method can be seen in Fig. 2.4a.

Whereas this method succeeds in detecting motion, it lacks of semantic understanding of the scene. As a result, only objects moving at the time of data collection can be detected, and dynamic objects which remain temporarily static would not be correctly



(a) Multi-view geometry. (b) Semantic segmentation. (c) Combination.

Figure 2.4: Detection and segmentation of dynamic objects using multi-view geometry (a), deep learning (b), and a combination of both geometric and learning methods (c). Notice that in (a) the person behind the desk cannot be detected, in (b) the book carried by the person and the chair he is sitting on cannot be segmented, and that finally the combination of the two approaches is the best performing (c).

identified, becoming part of the 3D map, eventually leading to failure in long-term SLAM scenarios. Also, a drawback of this method and in general of multi-view geometry approaches is that the initialization cannot be done on a single frame basis and it is not trivial. Even though they can distinguish different motions within the environment, they can only hypothesize that the motion of majority of the scene belongs to the camera ego motion and that the other motions belong to dynamic objects.

2.3 Detection of *a priori* Dynamic Objects

By *a priori* dynamic objects, we refer to semantic classes that inherently possess the capability to move either artificially or naturally. People, animals and means of transport are within such group. It is straightforward to detect these objects by the means of a CNN trained end-to-end in a supervised fashion for the specific task of classification. However, while in many applications a pure semantic analysis might be enough to segment out dynamic objects, a more general approach becomes valuable, especially, when moving towards unknown environments. This allows to detect general dynamic objects in a semi-supervised manner even if no prior segmentation classes are provided.

2.3.1 Supervised Detection

To detect *a priori* dynamic objects we propose to use a CNN trained end to end for image classification and pixel-wise semantic segmentation. The idea is to segment out those classes that are potentially dynamic or movable (person, bicycle, cat, dog, *etc.*).

A variety of recently proposed neural network architectures are tackling the problem of instance-level object segmentation (He et al. (2017), Pinheiro et al. (2016), Li et al. (2017)). They outperform traditional methods and are capable of handling a large set of object classes. Of these methods, Mask R-CNN (He et al. (2017)) is especially compelling, as it provides superior segmentation quality at a relatively high frame-rate of 5 Hz. We therefore propose to use Mask R-CNN and its TensorFlow implementation by Matterport[1] for dynamic objects detection. Mask R-CNN has been trained on the

[1] `https://github.com/matterport/Mask_RCNN`

MS COCO dataset (Lin et al. (2014)) in a supervised manner and estimates both pixel-wise semantic segmentation and the instance labels. Mask-RCNN achieves this by extending the architecture of Faster-RCNN (Ren et al. (2015)). Faster-RCNN is a two-stage approach that proposes regions of interest first and then predicts an object class and bounding box per region and in parallel. He et al. (2017) added a third branch to the second stage, which generates masks independently of class IDs and bounding boxes. Both stages rely on a feature map, which is extracted by a backbone network based on ResNet (He et al. (2016)), and apply convolutional layers for inference.

In our case, the pixel-wise semantic segmentation is sufficient to detect dynamic objects. Instance labels are useful though to track the different moving objects along time. Our input for Mask R-CNN is the RGB dynamic image. The list of potentially dynamic objects that Mask R-CNN can detect is the following: person, bicycle, car, motorcycle, airplane, bus, train, truck, boat, bird, cat, dog, horse, sheep, cow, elephant, bear, zebra and giraffe. We consider that, for most environments, the dynamic objects likely to appear are included within this list. If other classes were needed, the network could be fine-tuned with new training data. The output of the network for a $m \times n \times 3$ RGB input image is a matrix of size $m \times n \times l$, where l is the number of detected objects in the image. For each output channel $i \in l$ a binary semantic segmentation mask is obtained. The combination of the dynamic instance channels allows us to obtain the segmentation of all dynamic objects appearing in every image of the scene.

An example of the results of Mask R-CNN in the dynamic sequences from the TUM dataset (Sturm et al. (2012)) can be seen in Fig. 2.4b. The main limitation of the use of these methods to deal with dynamic content is that objects that are supposed to be static can be moved, and they would not be able to identify them. This is what occurs with the book and the chair of Fig. 2.4b. Also, the contours of its pixel-wise segmentation is far less accurate than the multi-view one (Fig. 2.4a).

2.3.2 Semi-Supervised Detection

When moving towards unknown environments, a general approach to segment out dynamic objects becomes especially valuable. Even though the length of the list of potentially dynamic objects that Mask R-CNN can detect, the real world is composed of many more objects that move in practice. We propose to use a combination of easily accessible simulation data and more expensive but specific real-world data to fine-tune Mask R-CNN so that it generalizes well to unseen environments. In this way, general dynamic objects can be detected, even if no prior segmentation classes are provided.

We leverage simulation data and offline short real-world video snippets recorded with a static RGB camera to automatically generate a per-pixel static/dynamic mask for each input image, which we use to fine-tune Mask R-CNN. Most of the training samples are sourced from the CARLA driving simulator (Dosovitskiy et al. (2017)). Then, we augment them with real-world samples that need to be manually collected. The synthetic data can be easily generated and also annotated using the semantic segmentation provided by the rendering engine. In order to avoid manual labeling of the real-world data, we propose an approach to obtain image masks from short video clips recorded with a static RGB camera, without any human intervention. This approach consists of three steps:

1. **Frame subtraction:** Several short video snippets are recorded in dynamic urban environments using a static camera. Each snippet is taken at another location.

The training frame set is obtained by sampling 5 images from each video clip. For each query image in this training set, all images from the snippet (except the query image) are subtracted from the query image to get absolute difference frames. Given that the camera is static, pixel differences correspond to changes in the environment which are caused by moving objects. The absolute difference frames are then thresholded to obtain binary instances masks.

2. **Voting scheme:** All binary instances masks for one training frame are summed up into one single image. Large intensity pixels belong to dynamic regions in the query frame, as those will differ from all the other images. More specifically, those pixels whose intensity is above a threshold value are marked as dynamic.

3. **Super-pixel segmentation:** The result of the previous pixel-wise operations are sparse and noisy masks. To address this issue, we partition the image into superpixels (Ren & Malik (2003)). If the percentage of dynamic pixels within a single super-pixel exceeds 5 %, the entire super-pixel is labeled as dynamic and holes inside dynamic masks are filled. At this stage, parts of a single dynamic object can be disjoint. To reduce this effect, we apply morphological operations to obtain the final segmentation masks.

In order to train the model, RGB images and their corresponding dynamic instance masks need to be provided. We use a combination of synthetic and real-world data to train our model. The synthetic data is generated by CARLA (Dosovitskiy et al. (2017)), which also allows for the extraction of semantic labels and therefore typical dynamic classes like pedestrians, cars or motorbikes. Only using this synthetic and semantically classified data would restrict our approach to a pre-defined set of semantic classes. Therefore, we also recorded our own real-world dataset in Zurich. This also allows to find general location-specific dynamic objects which might not be part of the synthetic data, *e.g.*, flags, animals or even water. The dynamic instance masks are extracted from the real-world data using the previous presented approach. For both datasets, we covered different weather conditions as well as different times of the day.

During the training procedure we always use a pre-trained version of the backbone part to avoid learning feature extraction from scratch. The top layers are initialized randomly due to the change in the output classes. The training phase is divided into two different stages. During the first training stage, only the model head layers are trained while the pre-trained backbone layers remain fixed. The training in the first stage is conducted purely with synthetic data, and is run for 50 epochs. During the second training stage only real-world data is used (also for 50 epochs). The learning rates are set as 10^{-3} and 10^{-4} in stages one and two, respectively.

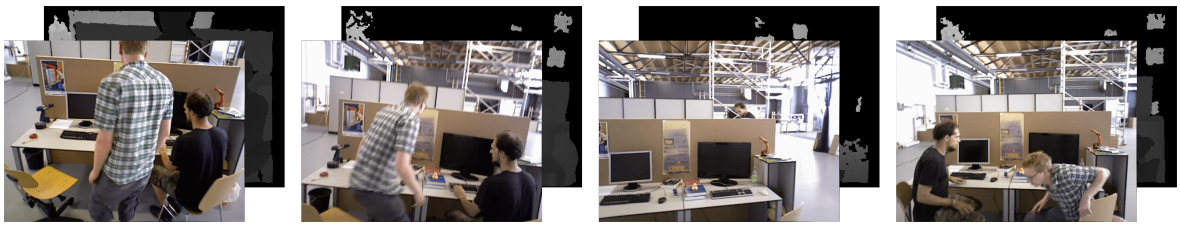
Mask R-CNN learns to segment dynamic parts of the image in a semi-supervised way, without being restricted to certain semantic object classes. We hypothesize that our approach will be able to learn general dynamic category-agnostic objects, even if no specific semantic labels are present during the training process. Fig. 2.5 shows various image frames of the detected dynamic object masks of our presented framework and their semantic segmentation. The qualitative results demonstrate that the classifier with real-world data manages to capture the location-specific objects, to the detriment of the accuracy of the pixel-wise segmentation of the different instances.

Figure 2.5: Qualitative analysis of the proposed dynamic instance detection compared to a standard semantic segmentation. Note how our algorithm trained on real-world data manages to capture the location-specific dynamic objects, such as trees or flags.

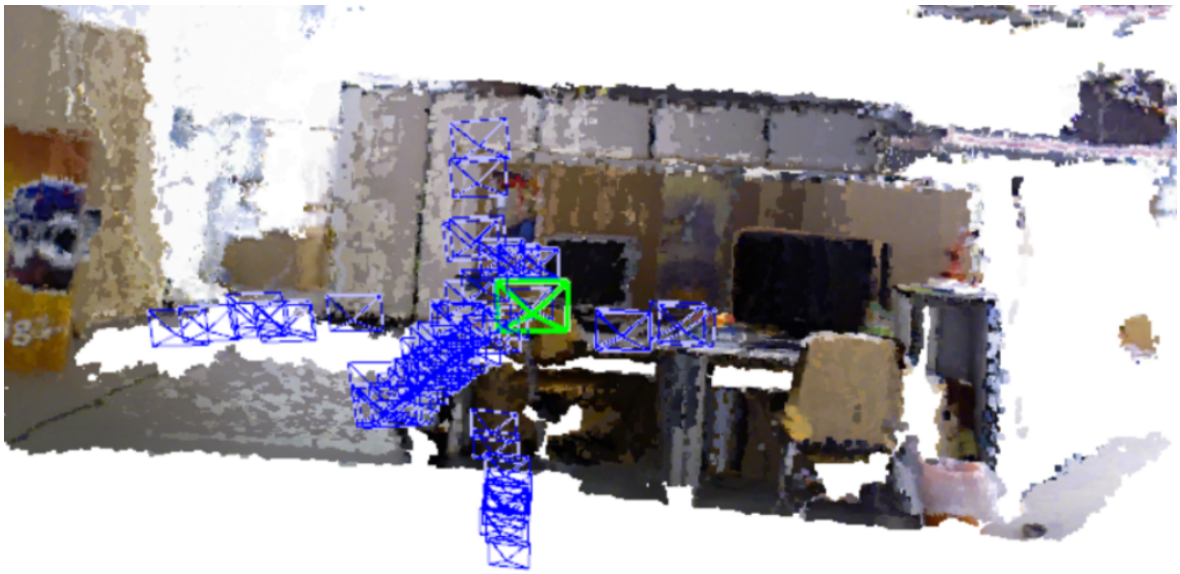
2.4 Integration in a SLAM System

At this point, we take the opportunity to introduce our system DynaSLAM (Bescos et al. (2018)). DynaSLAM is capable of detecting the moving objects either by multi-view geometry, deep learning or both. This allows us to detect both movement and *a priori* dynamic classes. It achieves state-of-the-art tracking accuracy in highly dynamic scenarios, reaching the baseline performance that usual SLAM systems achieve in static scenes. DynaSLAM can handle different input sensors such as monocular, stereo and RGB-D cameras. An example of our results for a RGB-D input sensor can be seen in Fig. 2.6: the input image stream presents significant dynamic content (Fig. 2.6a) that is handled with multi-view geometry and deep learning approaches to achieve an accurate camera trajectory and a map free of dynamic objects (Fig. 2.6b). This example corresponds to the sequence $fr3\_walking\_xyz$ from the TUM RGB-D dataset.

In this work we propose an on-line algorithm to deal with dynamic objects in RGB-D, stereo and monocular SLAM. DynaSLAM builds on top of the state-of-the-art ORB-SLAM2 system by Mur-Artal & Tardós (2017) with the purpose of rendering its tracking more accurate and building reusable maps of the scene. In the monocular



(a) Input RGB-D frames with dynamic content.



(b) Point-cloud map built with the poses computed by DynaSLAM and the segmented static frames.

Figure 2.6: Even though the input image sequence (a) contains large dynamic content, we can still compute an accurate camera tracking and build a $3D$ map which is stable in the long term and is which is not corrupted by dynamic objects (b). This $3D$ map, which is free of dynamic or spurious objects, can be of utility for future reuse.

and stereo cases our proposal is to use Mask R-CNN to pixel-wise segment the *a priori* dynamic objects in the frames, so that the SLAM algorithm does not extract features on them. In the RGB-D case we propose to combine multi-view geometry models and deep-learning-based algorithms for detecting dynamic objects.

Fig. 2.7 shows an overview of DynaSLAM. First of all, the RGB image stream channels pass through Mask R-CNN that estimates the pixel-wise segmentation of the image *a priori* dynamic content, *e.g.*, people or vehicles. In the RGB-D case, we use multi-view geometry to improve the dynamic content segmentation in two ways. First, we refine the segmentation of the dynamic objects previously detected by Mask R-CNN. Second, we label as dynamic new moving object instances that are either missed by the CNN or are not *a priori* dynamic. These segmented frames are used to estimate the camera trajectory and the map of the scene. Notice that if the moving objects in the scene are not within the CNN classes, the multi-view geometry stage would still detect the dynamic content, but the tracking accuracy might decrease.

2.4.1 Mask R-CNN

The input RGB image stream passes through Mask R-CNN to estimate the pixel-wise semantic segmentation of the *a priori* dynamic content, *e.g.*, people, animals or

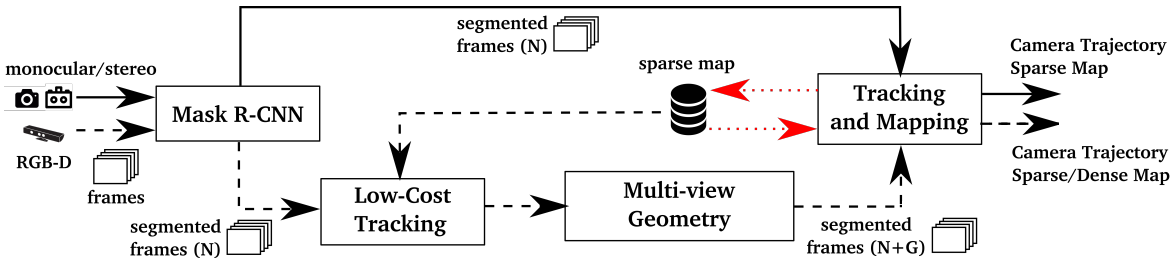


Figure 2.7: Block diagram of our pipeline for dynamic objects detection. In the stereo and monocular pipeline (—) the images pass through a CNN (Mask R-CNN) for computing the pixel-wise semantic segmentation of the *a priori* dynamic objects before being used for the mapping and tracking threads. In the RGB-D case (- - -) a second approach based on multi-view geometry is added for a more accurate motion segmentation, for which we need a low-cost tracking algorithm. The red dotted line (·····) represents the data flow of the stored sparse map and is used for all sensor types.

vehicles. We use for our experiments the Mask R-CNN implementation trained in a supervised manner on the MS COCO dataset (Lin et al. (2014)). One could also use a method like the one presented in Subsection 2.3.2 to discover new dynamic classes.

2.4.2 Low-Cost Tracking

After the potentially dynamic content has been segmented, the pose of the camera is tracked using the static part of the image. Because the segment contours usually become high-gradient areas, salient point features tend to appear. We do not consider the features in such contour areas. The tracking implemented at this stage of the algorithm is a simpler and therefore computationally lighter version of the one in ORB-SLAM2 (Mur-Artal & Tardós (2017)). It projects the map features in the image frame, searches for the correspondences in the static areas of the image, and minimizes the reprojection error to optimize the camera pose.

2.4.3 Multi-View Geometry

By using Mask R-CNN, most of the dynamic objects can be segmented and not used for tracking and mapping. However, there are objects that cannot be detected by this approach because they are not *a priori* dynamic, but movable. Examples of the latest are a book carried by someone, a chair that someone is moving, or even furniture changes in long-term mapping. The approach utilized for dealing with these cases has been previously detailed in Section 2.2.

The results of the CNN (Fig. 2.4b) can be combined with those of the multi-view geometry method (Fig. 2.4a) for a complete detection of dynamic objects (Fig. 2.4c). We can find several strengths and limitations in both methods, hence the motivation for their combined deployment. For the geometric approaches, the main problem is that initialization is not trivial because of its multi-view nature. Learning methods and their impressive performance using a single view do not have such initialization problems. Their main limitation though is that objects that are supposed to be static can be moved, and the method is not able to identify them. This last case can be solved though by applying multi-view consistency tests.

These two ways of facing the moving objects detection problem are illustrated in Fig. 2.4. In Fig. 2.4a we see that the person in the back, which is potentially a dynamic object, is not detected. The reasons for this issue are twofold. First, the difficulties that RGB-D cameras face when measuring the depth of distant objects since their accuracy typically decreases quadratically with the distance (Khoshelham & Elberink (2012)). And second, the fact that reliable features lie on defined, and therefore nearby, parts of the image. Albeit, this person is detected by the deep learning method (Fig. 2.4b). Apart from this, on one hand we see in the Fig. 2.4a that not only is detected the person in the front of the image, but also the book he is holding and the chair he is sitting on. On the other hand, in the Fig. 2.4b the two people are the only objects detected as dynamic, and also their segmentation is less accurate. If only the deep learning method is used, a floating book would be left in the images and would incorrectly become part of the 3D map. Because of the advantages and disadvantages of both methods, we consider that they are complementary and therefore their combined use is an effective way of achieving accurate tracking and mapping. In order to achieve this goal, if an object has been detected with both approaches, the segmentation mask should be that of the geometrical method. If an object has only been detected by the learning-based method, the segmentation mask should contain this information too. The final segmented image of the example in the previous paragraph can be seen in the Fig. 2.4c. The segmented dynamic parts are removed from the current frame and therefore from the map.

2.4.4 Tracking and Mapping

The input to this stage of the system contains the RGB and depth images, as well as their dynamic segmentation mask. We extract ORB features in the segments of the image which are classified as static. Because the segment contours are high gradient areas, the key points that fall at this intersection have to be eliminated.

2.5 Experimental Results

We have evaluated our system in the public datasets TUM RGB-D (Sturm et al. (2012)) and KITTI (Geiger et al. (2013)) and compared to other state-of-the-art SLAM systems in dynamic environments, using when possible results published in the original papers. Furthermore we have compared our system against the original ORB-SLAM2 to quantify the improvement of our approach in dynamic scenes. In this case, the results for some sequences were not published and we have ourselves completed their evaluation. Mur-Artal et al. (2015) propose to run each sequence five times and show median results, to account for the non-deterministic system nature. We have run each sequence ten times as dynamic objects are prone to exalt this non-deterministic effect.

2.5.1 TUM RGB-D Dataset

The TUM RGB-D dataset (Sturm et al. (2012)) is composed of 39 sequences recorded with a Microsoft Kinect sensor in different indoor scenes at full frame rate (30Hz). Both the RGB and the depth images are available, together with the ground-truth trajectory, the latest recorded by a high-accuracy motion capture system. In the sequences named sitting (s in our tables) there are two people sitting in front of a desk while speaking

and gesticulating, *i.e.*, there is a low degree of motion. In the sequences named walking (w), two people walk both in the background and the foreground and sit down in front of the desk. This dataset is highly dynamic and therefore challenging for standard SLAM systems. For both types of sequences sitting (s) and walking (w) there are four types of camera motions: (1) halfsphere ($half$): the camera moves following the trajectory of a 1-meter diameter half sphere, (2) xyz: the camera moves along the x-y-z axes, (3) rpy: the camera rotates over roll, pitch and yaw axes, and (4) static: the camera is kept static manually. We use the absolute trajectory root mean square error (RMSE) as the metric for our experiments, as proposed by Sturm et al. (2012).

The results of different variations of our system for six sequences within this dataset are shown in Table 2.1. Firstly, DynaSLAM (N) is the system in which only Mask R-CNN segments out the *a priori* dynamic objects. Secondly, in DynaSLAM (G) the dynamic objects have been only detected with the multi-view geometry method based on depth changes. Thirdly, DynaSLAM (N+G) stands for the system in which the dynamic objects have been detected by combining both the geometrical and deep learning approaches. According to this table, the system (N+G) that uses learning and geometry is the most accurate one in most sequences. The improvement over (N) comes from the segmentation of movable objects and refinement of the dynamic segments. The system (G) has higher error because it needs a relatively large object motion and its segmentation is only accurate after a small delay, during which the dynamic content introduces some error in the estimation. The DynaSLAM results shown from now on are from the best variant, that is, (N+G).

Sequence	DynaSLAM (N)	DynaSLAM (G)	DynaSLAM (N+G)
$w\_halfsphere$	**0.025**	0.035	**0.025**
w_xyz	**0.015**	0.312	**0.015**
w_rpy	0.040	0.251	**0.035**
w_static	0.009	0.009	**0.006**
$s\_halfsphere$	**0.017**	0.018	**0.017**
s_xyz	0.014	**0.009**	0.015

Table 2.1: Absolute trajectory RMSE [m] for several variants of DynaSLAM (RGB-D).

Table 2.2 shows our results on the same sequences, compared against RGB-D ORB-SLAM2. Our method outperforms ORB-SLAM2 in highly dynamic scenarios ($walking$), reaching an error similar to that of the original RGB-D ORB-SLAM2 system in static scenarios. In the case of low-dynamic scenes ($sitting$) the tracking results are slightly worse because the tracked key points find themselves further than those belonging to dynamic objects. Albeit, DynaSLAM's map does not contain the dynamic objects that appear along the sequence. Fig. 2.8 shows an example of the estimated trajectories of DynaSLAM and ORB-SLAM2, compared to the ground truth.

Table 2.3 shows a comparison between our system and several state-of-the-art RGB-D SLAM systems designed for dynamic environments. In account for the effectiveness of our and the state-of-the-art approaches for motion detection (independently of the utilized SLAM system), we also show the respective improvement values against the original SLAM system used in every case. DynaSLAM significantly outperforms all of them in all sequences (both high and low dynamic ones). The error is, in general,

Sequence	ORB-SLAM2 (RGB-D) Mur-Artal & Tardós (2017)	DynaSLAM (N+G) (RGB-D)		
	median	median	min	max
w_halfsphere	0.351	**0.025**	0.024	0.031
w_xyz	0.459	**0.015**	0.014	0.016
w_rpy	0.662	**0.035**	0.032	0.038
w_static	0.090	**0.006**	0.006	0.008
s_halfsphere	0.020	**0.017**	0.016	0.020
s_xyz	**0.009**	0.015	0.013	0.015

Table 2.2: Comparison of the RMSE of ATE [m] of DynaSLAM against ORB-SLAM2 for RGB-D cameras. To account for the non-deterministic nature of the system, we show the median, minimum and maximum error of ten runs.

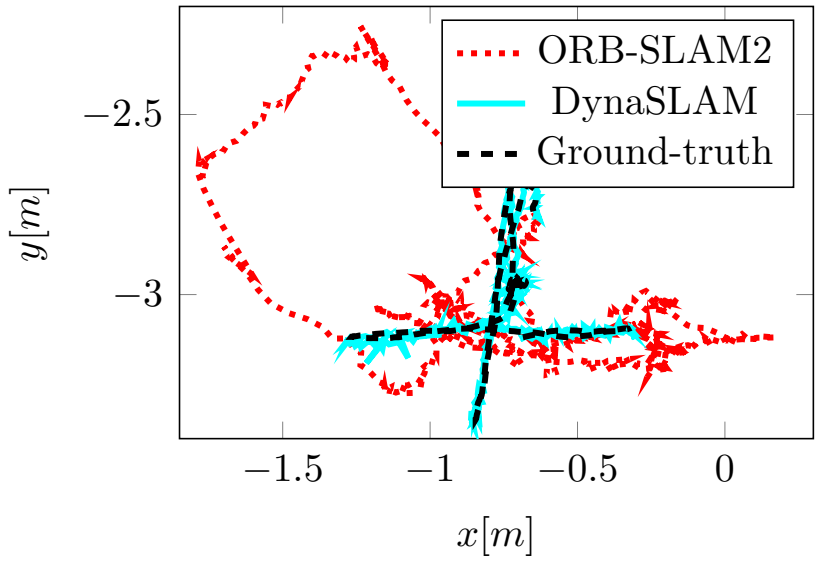


Figure 2.8: Ground truth and trajectories estimated by DynaSLAM and ORB-SLAM2 respectively in the TUM dataset sequence *fr3/walking_xyz*.

around 1-2 cm, similar to that of the state of the art in static scenes. Our motion detection approach also outperforms the other methods.

ORB-SLAM, the monocular version of ORB-SLAM2, is generally more accurate than the RGB-D one in dynamic scenes, due to the nature of their initialization algorithms. RGB-D ORB-SLAM2 is initialized and starts the tracking from the very first frame, and hence dynamic objects can introduce errors. Monocular ORB-SLAM though delays the initialization until there is parallax and consensus using the staticity assumption. Hence, it does not track the camera for the full sequence, sometimes missing a substantial part of it, or even not initializing.

Table 2.4 shows the tracking results and percentage of the tracked trajectory for ORB-SLAM and DynaSLAM (monocular) in the TUM dataset. The initialization in DynaSLAM is always quicker than that of ORB-SLAM. In fact, in highly dynamic sequences, ORB-SLAM initialization only occurs when the moving objects disappear –or almost disappear– from the scene. In conclusion, although the accuracy of DynaSLAM

Sequence	DES Li & Lee (2017)	MS DSLAM Wang & Huang (2014)			MR DVO-SLAM Sun et al. (2017)			DynaSLAM		
	w/	w/o	w/	↑ [%]	w/o	w/	↑ [%]	w/o	w/	↑ [%]
w_half	0.049	0.116	0.055	52.59	0.529	0.125	76.32	0.351	**0.025**	**92.88**
w_xyz	0.060	0.202	0.040	80.20	0.597	0.093	84.38	0.459	**0.015**	**96.73**
w_rpy	0.179	0.515	0.076	85.24	0.730	0.133	81.75	0.662	**0.035**	**94.71**
w_stat	0.026	0.470	0.024	**94.89**	0.212	0.066	69.06	0.090	**0.006**	93.33
s_half	0.043	-	-	-	0.062	0.047	**23.70**	0.020	**0.017**	15.00
s_xyz	0.040	-	-	-	0.051	0.048	**4.55**	0.009	**0.015**	X

Table 2.3: Absolute trajectory RMSE [m] of DynaSLAM against state-of-the-art RGB-D SLAM systems specifically designed to work in dynamic scenes. To evaluate the effectiveness of the specific module addressing dynamic content, we report the improvement with respect to the original SLAM systems without any motion detection stage (w/o). Our results are estimated using Mask R-CNN and multi-view geometry.

is slightly lower, it succeeds in bootstrapping the system with dynamic content and producing a map without such content, to be re-used for long-term applications. The reason why DynaSLAM is slightly less accurate is that the estimated trajectory is longer, and there is therefore room for accumulating more tracking errors.

Sequence	ORB-SLAM Mur-Artal et al. (2015)		DynaSLAM (Monocular)	
	ATE [m]	% Traj	ATE [m]	% Traj
fr3/walking_halfsphere	**0.017**	87.16	0.021	**97.84**
fr3/walking_xyz	**0.012**	57.63	0.014	**87.37**
fr2/desk_with_person	**0.006**	95.30	0.008	**97.07**
fr3/sitting_xyz	**0.007**	91.44	0.013	**100.00**

Table 2.4: Absolute trajectory RMSE [m] and percentage of successfully tracked trajectory for both ORB-SLAM and DynaSLAM (monocular).

2.5.2 KITTI Odometry Dataset

The KITTI Dataset (Geiger et al. (2013)) contains stereo sequences recorded from a car perspective in urban and highway environments. Table 2.5 shows our results in the eleven training sequences, compared against stereo ORB-SLAM2. We use two different metrics, the absolute trajectory RMSE proposed by Sturm et al. (2012), and the average relative translation (RPE) and rotation errors (RRE), proposed by Geiger et al. (2013). Furthermore, Table 2.6 shows the results in the same sequences for the monocular variants of ORB-SLAM and DynaSLAM.

Note that the comparison of the results are similar in both the monocular and stereo cases, but the former is more sensitive to dynamic objects and therefore to the additions in DynaSLAM. In some sequences the accuracy of the tracking is improved when not using features belonging to *a priori* dynamic objects, *i.e.*, cars, bicycles, *etc.* An example of this would be the sequences KITTI 01 and KITTI 04, in which

Sequence	ORB-SLAM2 (Stereo) Mur-Artal & Tardós (2017)			DynaSLAM (Stereo)		
	RPE [%]	RRE [°/100m]	ATE [m]	RPE [%]	RRE [°/100m]	ATE [m]
KITTI 00	**0.70**	**0.25**	**1.3**	0.74	0.26	1.4
KITTI 01	**1.39**	**0.21**	10.4	1.57	0.22	**9.4**
KITTI 02	**0.76**	**0.23**	**5.7**	0.80	0.24	6.7
KITTI 03	0.71	0.18	0.6	**0.69**	0.18	0.6
KITTI 04	0.48	0.13	0.2	**0.45**	**0.09**	0.2
KITTI 05	0.40	0.16	0.8	0.40	0.16	0.8
KITTI 06	0.51	**0.15**	0.8	**0.50**	0.17	0.8
KITTI 07	**0.50**	**0.28**	0.5	0.52	0.29	0.5
KITTI 08	1.05	0.32	3.6	1.05	0.32	**3.5**
KITTI 09	**0.87**	**0.27**	3.2	0.93	0.29	**1.6**
KITTI 10	**0.60**	**0.27**	**1.0**	0.67	0.32	1.2

Table 2.5: Comparison of the RMSE of the ATE [m], the average of the RPE [%] and the RRE [°/100m] of DynaSLAM against ORB-SLAM2 system for stereo cameras.

Sequence	ORB-SLAM Mur-Artal & Tardós (2017)	DynaSLAM (Monocular)
KITTI 00	**5.33**	7.55
KITTI 02	**21.28**	26.29
KITTI 03	**1.51**	1.81
KITTI 04	1.62	**0.97**
KITTI 05	4.85	**4.60**
KITTI 06	**12.34**	14.74
KITTI 07	**2.26**	2.36
KITTI 08	46.68	**40.28**
KITTI 09	6.62	**3.32**
KITTI 10	8.80	**6.78**

Table 2.6: Absolute trajectory RMSE [m] for ORB-SLAM and DynaSLAM when utilizing a monocular sensor.

all vehicles that appear are moving. In the sequences in which most of the recorded cars and vehicles are parked (hence static), the absolute trajectory RMSE is usually bigger since the key points used for tracking are more distant and usually belong to low-texture areas (KITTI 00, KITTI 02, KITTI 06). However, the loop closure and relocalization algorithms work more robustly since the resulting map only contains structural objects, *i.e.*, the map can be re-used and work in long-term applications.

2.5.3 Timing Analysis

To complete the evaluation of our proposal, Table 2.7 shows the average computational time for its different stages. Note that DynaSLAM is not optimized for real-time

operation. However, its capability for creating life-long reusable maps of the static scene content are also relevant for running on offline mode.

Sequence	Low-Cost Tracking	Multi-View Geometry
w_halfsphere	1.69	333.68
w_rpy	1.59	235.98

Table 2.7: DynaSLAM average computational time [ms].

On the one hand, Mur-Artal et al. (2015) and Mur-Artal & Tardós (2017) show real-time results for both ORB-SLAM and ORB-SLAM2 respectively. On the other hand, He et al. (2017) report that Mask R-CNN runs at 195 ms per image on a Nvidia Tesla M40 GPU. The recent work of YOLACT (Bolya et al. (2019)) achieves an accuracy similar to that of Mask R-CNN at the incredible speed of 33.5 fps on a single Titan Xp. These two methods could be easily swapped for a faster performance.

The addition of the multi-view geometry stage is an additional slowdown, due mainly to the region growth algorithm. This step could also be speeded up by decomposing the typical region-growing algorithm utilizing parallel computation (Yau et al. (2013)), which allows to quickly segment dynamic instances on a GPU. Furthermore, the region-growing step is only necessary when building a dense map is necessary.

2.6 Failure Modes and Future Work

The contributions that we have presented in this chapter show unprecedented accuracy and robustness of visual SLAM in highly dynamic scenarios. However, there is still plenty of room for further improvement. As future work, it would be interesting to develop a motion detection approach that is based uniquely on RGB information, so that it could be helpful for monocular setups too. Optimizing DynaSLAM for real-time performance would also bring great benefits to the robotics community. For example, implementing the suggestions made in Subsection 2.5.3 would yield great progress in this direction. Also, future extensions of this work might include explicitly distinguishing between those *a priori* dynamic and moving objects. For example, if a car is detected by Mask R-CNN (*a priori* dynamic) but is not currently moving, its corresponding keypoints should be used for the local tracking, but should neither be included in the map nor used for visual place recognition.

2.7 Discussion

We have presented a visual SLAM system that, building on top of ORB-SLAM2, adds a motion segmentation approach that renders it robust in dynamic environments for monocular, stereo and RGB-D cameras. Our system robustly and accurately tracks the camera and creates a static and therefore reusable map of the scene. We believe that DynaSLAM represents a significant advance in visual SLAM in highly dynamic environments and paves the way for future research in the topic. We refer the reader to a video that shows the potential of DynaSLAM[2].

[2] `https://youtu.be/EabI_goFmQs`

The comparison against the state of the art shows that DynaSLAM achieves in most cases the highest accuracy. In the TUM Dynamic Objects dataset, DynaSLAM is currently the best RGB-D SLAM solution. In the monocular case, our accuracy is similar to that of ORB-SLAM, obtaining however a static map of the scene with an earlier initialization. In the KITTI dataset DynaSLAM is slightly less accurate than monocular and stereo ORB-SLAM, except for those cases in which dynamic objects represent an important part of the scene. However, our estimated map only contains structural objects and can therefore be re-used in long-term applications.

2.8 Related Publications

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes". *IEEE Robotics and Automation Letters* and oral presentation at *IEEE International Conference on Intelligent Robots and Systems*, October 2018.

- Guoxiang Zhou, Berta Bescos, Marcin Dymczyk, Mark Pfeiffer, José Neira , Roland Siegwart. "Dynamic Objects Segmentation for Visual Localization in Urban Environments". Poster Presentation within the Workshop From freezing to jostling robots: Current challenges and new paradigms for safe robot navigation in dense crowds at *IEEE International Conference on Intelligent Robots and Systems*, October 2018.

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "Detecting, Tracking and Eliminating Dynamic Objects in 3D Mapping using Deep Learning and Inpainting". Oral and Poster Presentation within the Workshop Representing a Complex World: Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding at *IEEE International Conference on Robotics and Automation*, May 2018.

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "Robust and Accurate 3D Mapping by combining Geometry and Machine Learning to deal with Dynamic Objects". Poster Presentation within the Workshop Learning for Localization and Mapping at *IEEE International Conference on Intelligent Robots and Systems*, September 2017.

Chapter 3

Inpainting of Dynamic Objects

Most vision-based localization systems are conceived to work in static environments (Mur-Artal & Tardós (2017), Engel et al. (2017), Forster et al. (2014)). They can deal with small fractions of dynamic content through RANSAC and robust cost functions mostly, but tend to compute dynamic objects motion as camera ego-motion when these become relevant within the scene. Thus, their performance is compromised. Building stable maps is also of key importance for long-term autonomy. Mapping dynamic objects prevents vision-based robotic systems from recognizing already visited places and reusing pre-computed maps.

To deal with dynamic objects, some approaches include in their model the behavior of the observed dynamic content (Lamarca et al. (2019)). Such strategy is needed when the majority of the observed scene is not rigid. However, when scenes are mainly rigid, as in Fig. 2.6a, the standard strategy consists of detecting the dynamic objects within the images and not to use them for localization and mapping, as in the work presented in the previous chapter (Bescos et al. (2018)) among others (Alcantarilla et al. (2012), Wang & Huang (2014), Tan et al. (2013)). To address mainly rigid scenes, we propose in this chapter to instead modify these images so that dynamic content is deleted and the scene is converted realistically into static. We consider that the combination of experience and context allows to hallucinate, *i.e.*, inpaint, a geometrically and semantically consistent appearance of the rigid and static structure behind dynamic objects. This hallucinated structure can be used by localization and mapping systems to provide them with robustness against dynamic objects. Turning images containing dynamic objects into realistic static frames reveals several challenges:

- Detecting such dynamic content in the image. By this, we mean to detect not only those objects that are known to move such as vehicles, people and animals, but also the shadows and reflections that they might generate, since they also change the image appearance. This can be done with any of the approaches presented in the previous chapter.
- Inpainting the resulting space left by the detected dynamic content with plausible imagery. The resulting image would succeed in being realistic if the inpainted areas are both semantically and geometrically consistent with the static content of the image.

The first challenge can be addressed with geometrical approaches if an image sequence is available. This procedure usually consists in studying the optical flow consistency along the images (Wang & Huang (2014), Alcantarilla et al. (2012), Bescos

et al. (2018)). In the case in which only one frame is available, deep learning is the approach that excels at this task by the use of CNNs (He et al. (2017), Romera et al. (2018), Badrinarayanan et al. (2017). These frameworks are trained with the previous knowledge of what classes are dynamic and which ones are not. More recent works show that it is possible to acquire this knowledge in a self-supervised way (Barnes et al. (2018), Zhou, Bescos, Dymczyk, Pfeiffer, Neira & Siegwart (2018)).

Regarding the second challenge, traditional image inpainting approaches use image statistics of the remaining image to fill in the holes (Telea (2004), Bertalmio et al. (2001)). The former work estimates the pixel value with the normalized weighted sum of all the known pixels in the neighbourhood. While this approach generally produces smooth results, it is limited by the available image statistics and has no concept of visual semantics. Neural networks learn semantic priors and meaningful hidden representations in an end-to-end fashion, which have been used for recent image inpainting efforts (Liu et al. (2018), Yu et al. (2018), Iizuka et al. (2017), Pathak et al. (2016)). These networks employ convolutional filters on images, replacing the removed content with inpainted areas that have geometrical and semantic consistency with the whole image. The recent work of Rosinol et al. (2020) uses a tecnhique called *dynamic masking* to map static parts of dynamic scenes. Besides that, regarding video-based inpainting works, the work by Granados et al. (2012) removes marked dynamic objects from videos by aligning other candidate frames in which parts of the missing region are visible, assuming that the scene can be approximated using piecewise planar geometry. The recent work of Uittenbogaard et al. (2019) utilizes a GAN to learn to use information from different viewpoints and select imagery information from those views to generate a plausible inpainting which is similar to the ground-truth static background.

We address the inpainting of the static background with video-based techniques by means of multi-view geometry (Section 3.2) and also with deep-learning image-based inpainting (Section 3.3). To the best of our knowledge, we are the first ones to attempt to solve visual SLAM in dynamic environments by hallucinating the plausible static background directly from the sensor data.

3.1 Related Work

Inpainting is the process of reconstructing lost or deteriorated parts of images and videos. For instance, in the museum world, in the case of a valuable painting, this task would be carried out by a skilled art conservator or art restorer. In the digital world, inpainting (also known as image interpolation or video interpolation) refers to the application of sophisticated algorithms to replace lost or corrupted parts of the image data. We hereby summarize the different digital inpainting techniques that are the current state of the art, and highlight how they differ from ours.

3.1.1 Video-Based Inpainting

The work by Granados et al. (2012) removes marked dynamic objects from videos by aligning other candidate frames in which parts of the missing region are visible, assuming that the scene can be approximated using piecewise planar geometry. The recent work by Uittenbogaard et al. (2019) utilizes a Generative Adversarial Network (GAN) to learn to use information from different viewpoints and select imagery information

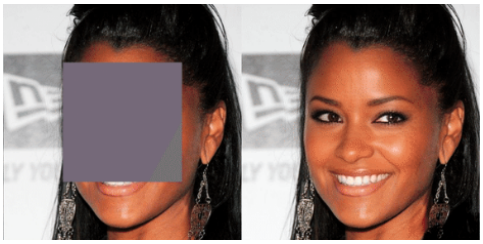
from those views to generate a plausible inpainting which is similar to the ground-truth static background. The former work assumes that there is very little motion between consecutive frames, and the latter work uses GPS data to help with the inpainting task. These assumptions make these methods unsuitable for SLAM applications.

Previous works on SLAM in dynamic scenes have attempted to reconstruct the background occluded by dynamic objects in the images with information from previous frames (Scona et al. (2018)). In this chapter, more precisely in Section 3.2, we present one of our contributions to video-based inpainting within a SLAM framework for RGB-D sensor setups (Bescos et al. (2018)). Such works –both StaticFusion (Scona et al. (2018)) and ours– need per-pixel depth information and only make use of the static content of the pre-built map to create the inpainted frames, but do not add any semantic consistency. Eventually, if only one frame is available, the static occluded background can only be reconstructed by utilizing image-based inpainting techniques.

3.1.2 Image-Based Inpainting

Among the non-learning approaches to image-based inpainting, propagating appearance information from neighboring pixels to the target region is the usual procedure (Telea (2004), Bertalmio et al. (2001)). Accordingly, these methods succeed in dealing with small and narrow holes, as can be seen in Fig. 3.1a, where color and texture vary smoothly, but fail when handling big holes, resulting in over-smoothing. Differently, patch-based methods iteratively search for relevant patches from the rest of the image (Efros & Freeman (2001)). These approaches are computationally expensive and hence not fast enough for real-time applications. Yet, they do not make semantically aware patch selections.

(a) The areas of the damaged image can be correctly recovered using the statistics of the image itself, as these are small (Telea (2004)). It is not necessary to have a semantic understanding of the image to reconstruct it.



(b) This reconstruction can only be done with deep learning techniques, since the reconstructed image has semantic as well as geometric coherence (Liu et al. (2018)).

Figure 3.1: Examples of inpainting methods without (a) and with deep learning (b). On the one hand, the former approaches can handle small narrow holes, but result in over-smoothing when dealing with big holes. For example, inpainting the face in (b) with one of these methods would generate a face without eyes. On the other hand, the combination of experience and context allows deep-learning works to inpaint a geometrically and semantically consistent appearance.

Deep learning based inpainting methods usually initialize the image holes with a constant value, and further pass it through a CNN. Context Encoders (Pathak et al. (2016)) were among the first ones to successfully use a standard pixel-wise reconstruction loss, as well as an adversarial loss for image inpainting tasks. Due to the resulting

artifacts, Yang et al. (2017) take the results from Context Encoders as input and then propagate the texture information from non-hole regions to fill the hole regions as post-processing. Song et al. (2017) use a refinement network in which a blurry initial hole-filling result is used as the input, then iteratively replaced with patches from the closest non-hole regions in the feature space. Iizuka et al. (2017) extend Content Encoders by defining global and local discriminators, then apply a post-processing. Following this work, Yu et al. (2018) replaced the post-processing with a refinement network powered by the contextual attention layers. The recent work of Liu et al. (2018) obtains excellent results by using partial convolutions, as can be seen with the example of Fig. 3.1b. The combination of experience and context allows these works to hallucinate, *i.e.*, inpaint, a geometrically and semantically consistent appearance of the reconstructed scene.

In contrast, the work by Ulyanov et al. (2018) proves that there is no need for external dataset training. The generative network itself can rely on its structure to capture a great deal of low-level image statistics and complete the corrupted image. However, this approach usually applies many iterations to get good and detailed results.

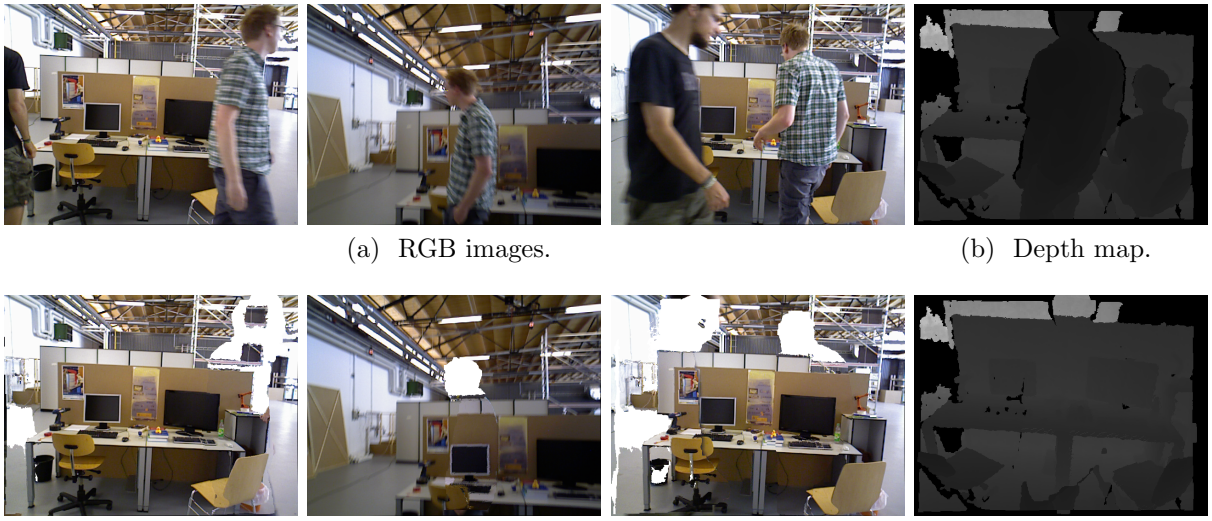
One of our works, which will be presented in Section 3.3, bins the image sequences and treats the frames independently. Therefore, it makes use of image-based "inpainting". We first use deep learning to segment out the *a priori* moving objects –vehicles, animals and pedestrians–, and the binary dynamic mask is fed to the inpainting process. Since partial or missing segmentation can happen frequently for any semantic segmentation network when being used in practice, it is of high importance to use a framework robust to such inaccuracies. We therefore aim to use an image-to-image translation framework rather than a pure inpainting one. Image-to-image translation is the task of taking images from one domain and transforming them so they have the style or characteristics of images from another domain. The adoption of an image-to-image translation framework allows us to slightly modify the image non-hole regions for better accommodation of the reconstructed areas. Differently to inpainting methods, the "holes" cannot be initialized with any placeholder values because we do not want the framework to only modify those values and hence, our image-to-image translation network input consists of the dynamic original image with the mask concatenated.

3.2 Geometric Inpainting

In this section we build on top of the contributions presented in Chapter 2 and go one step further by inpainting the background behind dynamic objects with the static 3D map information. For every removed dynamic object, we aim at inpainting the occluded background with static information from previous views, so that we can synthesize a realistic image without moving content. We believe that such synthesized frames, containing the static structure of the environment, are useful for applications such as virtual and augmented reality, and for relocation and camera tracking after the map is created.

Once we know the position of the previous and current frames, we project into the dynamic segments of the current frame the RGB and depth channels from a set of all the previous key frames (the last 20 in our experiments). Some gaps have no correspondences and are left blank: some areas cannot be inpainted because their correspondent part of the scene has not appeared so far in the key frames, or, if it has appeared, it

has no valid depth information. These gaps cannot be reconstructed with geometrical methods and would need a more elaborate inpainting technique. Fig. 3.2 shows the resulting synthesized images for three input frames from different sequences of the TUM benchmark. Notice how the dynamic content has been successfully segmented and removed. Also, most of the segmented parts have been properly inpainted with information from the static background. We can see though a small color difference between the real and the reconstructed parts due to illumination changes.



(a) RGB images. (b) Depth map.

(c) Inpainted RGB images. (d) Inpainted depth map.

Figure 3.2: Qualitative results of our inpainting approach. In (a) we show three RGB input frames, and in (c) we show the output of our system, in which all dynamic objects have been detected and the background has been reconstructed. (b) and (d) show respectively the depth input and output maps, which have also been processed.

Another application of these synthesized frames would be the following: if the frames dynamic areas are inpainted with the static content, the system can work as a SLAM system under the staticity assumption using the inpainted images. The pipeline of a such a system can be seen in Fig. 3.3. Opposed to the previously introduced pipeline (Fig. 2.7), this system introduces the background inpainting stage right before the localization of the camera. The resulting camera ATE can be consulted in Table 3.1 in comparison with the SLAM results with dynamic objects detection. Adding the background inpainting stage (BI) before the localization of the camera usually leads to less accuracy in the tracking. The reason is that the background reconstruction is strongly correlated with the camera poses. Hence, for sequences with purely rotational motion (*rpy*, *halfsphere*), the estimated camera poses have a greater error and lead to a non-accurate background reconstruction. The main accomplishment of the background reconstruction is seen in the synthesis of the static images (Fig. 3.2) for applications such as virtual reality or cinematography. These frames could also be of particular value for the task of visual place recognition.

Geometry and video-based inpainting methods as this one allow to reconstruct images that can be semantically and geometrically coherent. However, either an accurate camera frame pose is needed or assumptions that limit its applicability are made about the camera movement. The nature of this solution limits its applicability for visual odometry problems, however, it can provide a great benefit for place recognition prob-

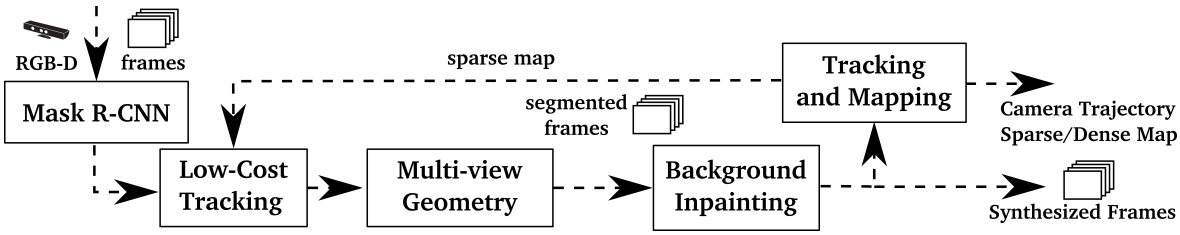


Figure 3.3: Block diagram of our pipeline for dynamic objects detection and inpainting. In the case in which a RGB-D camera is used (- - -), we inpaint the background occluded by dynamic objects once they are detected. These synthesized frames are directly used for both camera tracking and mapping.

Sequence	DynaSLAM (N)	DynaSLAM (G)	DynaSLAM (N+G)	DynaSLAM (N+G+BI)
w_halfsphere	**0.025**	0.035	**0.025**	0.029
w_xyz	**0.015**	0.312	**0.015**	**0.015**
w_rpy	0.040	0.251	**0.035**	0.136
w_static	0.009	0.009	**0.006**	0.007
s_halfsphere	**0.017**	0.018	**0.017**	0.025
s_xyz	0.014	**0.009**	0.015	0.013

Table 3.1: Absolute trajectory RMSE [m] for several variants of DynaSLAM (RGB-D).

lems. Image-based inpainting methods can though create realistic images uniquely based on the image statistics, regardless of the camera trajectory within the scene structure. Among image-based inpainting methods, the ones that make use of deep learning excel at reconstructing the static scene when semantic information –on top of geometric– is required. The nature of this solution allows to translate dynamic images into static as a pre-processing stage of any visual odometry or SLAM framework. This is what the following sections in this chapter are about.

3.3 Inpainting with Deep Learning

In this section we aim at utilizing more elaborate inpainting techniques based on deep learning. This allows us to perform inpainting on images uniquely with RGB information –opposed to with RGB-D data as in the previous section–. Also, deep learning leverages experience and the image context to inpaint the static background with plausible imagery with no explicit knowledge of the camera pose.

3.3.1 Image-to-Image Translation

Our work makes use of the successful image-to-image translation framework by Isola et al. (2017). For the sake of completeness, we summarize the basis of their approach.

A GAN is a generative model that learns a mapping from a random noise vector z to an output image y, G: $z \rightarrow y$ (Goodfellow et al. (2014)). In contrast, a conditional GAN (cGAN) learns a mapping from observed image x and optional random noise vector z, to y, $G : \{x, z\} \rightarrow y$ (Gauthier (2014)), or $G : x \rightarrow y$ (Isola et al. (2017)).

The generator G is trained to produce outputs indistinguishable from the "real" images by an adversarially trained discriminator, D, which is trained to do as well as possible at detecting the generator's "fakes". The objective of a cGAN can be expressed as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_x[\log (1 - D(x, G(x)))], \tag{3.1}$$

where G tries to minimize this objective against an adversarial D that tries to maximize it. Previous approaches have found it beneficial to mix the GAN's objective with a more traditional appearance loss, such as the $L1$ or $L2$ distance (Pathak et al. (2016)). The discriminator's job remains unchanged, but the generator is tasked not only with fooling the discriminator, but also with being near the ground-truth in a $L1$ sense, as expressed in

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda_1 \cdot \mathcal{L}_{L1}(G), \tag{3.2}$$

where $\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[||y - G(x)||_1]$. The recent work of Isola et al. (2017) shows that cGANs are suitable for image-to-image translation tasks, where the output image is conditioned on its corresponding input image, *i.e.*, it translates an image from one space into another (semantic labels to RGB appearance, RGB appearance to drawings, day to night, *etc.*). The realism of their results is also enhanced by their generator architecture. They employ a U-Net (Ronneberger et al. (2015)), which allows low-level information to shortcut across the network. We have employed such architecture in our work and obtained fair results for 256×256 resolution images. However, visual localization systems see their accuracy degraded when working with such low-resolution images. For this objective, we therefore decide to employ, as the architecture of our generator G, a UResNet (Guerrero et al. (2018)), see Fig. 3.4. This architecture uses residual blocks (Hinton & Salakhutdinov (2006)) and has shown impressive results for super-resolution images (Kingma & Welling (2013)).

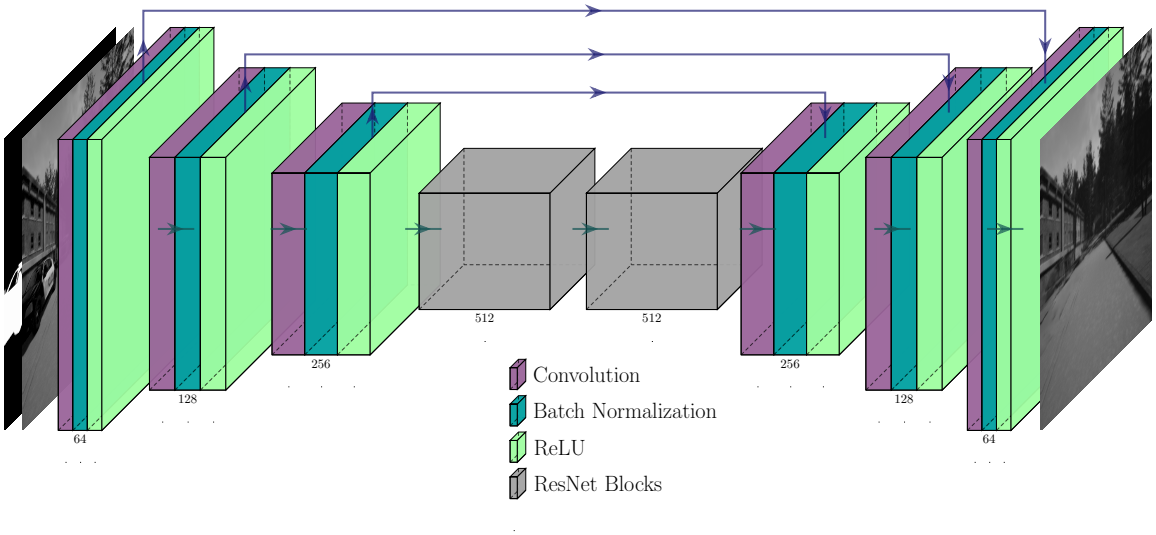


Figure 3.4: Our generator G adopts a UResNet-like architecture. It employs 3 down-convolutional layers with a stride of 2, 6 ResNet blocks and 3 up-convolutional layers with a fractional stride of 1/2, with skip connections between corresponding down- and up-convolutional layers. Only 2 ResNet blocks are shown for simplicity.

It is well known that $L2$ and $L1$ losses produce blurry results on image generation problems, *i.e.*, they can capture the low frequencies but fail to encourage high

frequency crispness. This motivates restricting the GAN discriminator to only model high frequency structures. Following this idea, Isola et al. (2017) adopt a discriminator architecture that classifies each $N \times N$ patch in an image –rather than classifying the image as a whole– as real or fake. Due to their excellent results, we adopt this same architecture for our discriminator.

3.3.2 Empty Cities

We name our proposed system Empty Cities. Empty Cities turns images of an urban environment that show dynamic content, such as vehicles or pedestrians, into realistic static frames which are suitable for localization and mapping. We first obtain the pixel-wise semantic segmentation of the RGB dynamic image (SS) –see Fig. 3.5–. Then, the segmentation of only the dynamic objects is obtained with the convolutional network $DynSS$. Once we have this mask, we convert the RGB dynamic image to gray scale and we compute the static image, also in gray scale, with the use of the generator G, which has been trained in an adversarial way –for simplicity, the discriminator is not shown in this diagram–. To fully exploit the capabilities of this framework for localization and mapping, inpainting is enriched with a loss based on ORB features detection, orientation and descriptors between the ground-truth and computed static images. Another feature of our framework for localization and mapping is the fact that we perform the inpainting in gray-scale rather than in RGB. The motivation for this is that many visual localization applications only need the images grayscale information. The different stages are described in the following blocks.

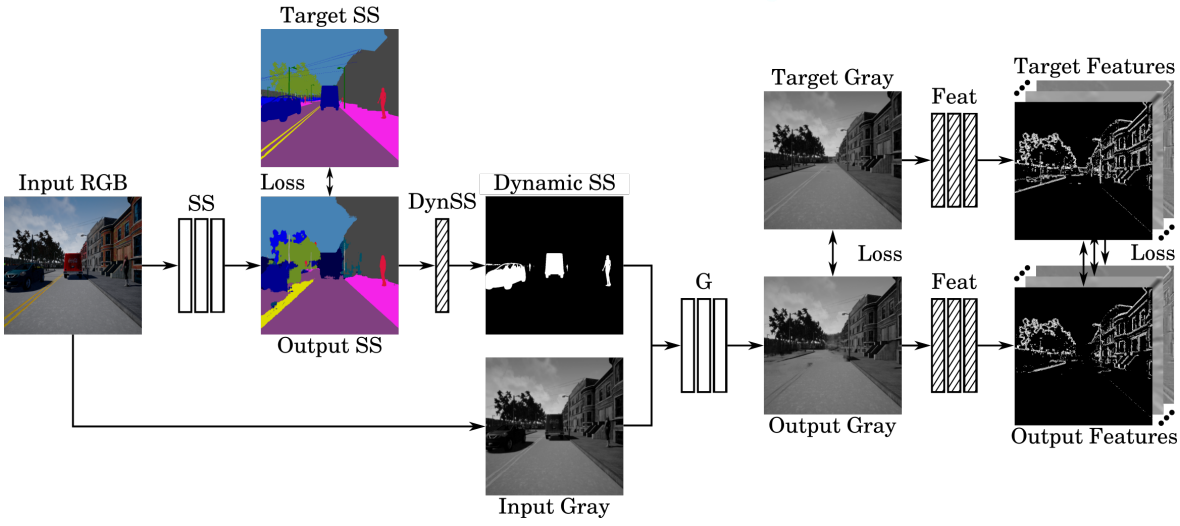


Figure 3.5: Block diagram of our proposal. We first compute the segmentation of the RGB dynamic image, as well as its loss against its ground truth. Both the dynamic/static binary mask and dynamic image are used to obtain the static image. A loss based on ORB features together with an appearance and an adversarial loss are obtained and back-propagated until the RGB dynamic image. The striped blocks are differentiable layers that are fixed and hence not modified during training time. The adversarial discriminator is not shown here for simplicity.

From Image-to-Image Translation to Inpainting

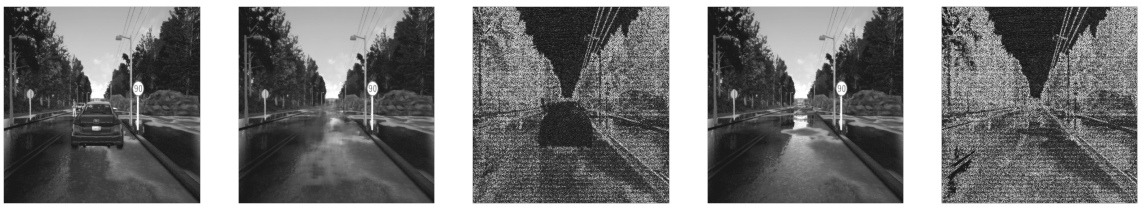
For our objective, dynamic objects masks are specially considered to re-formulate the training objectives of the general purpose image-to-image translation work by Isola et al. (2017). We adopt a variant of the *cGAN* that learns a mapping from observed image x and dynamic/static binary mask m, to y, $G : \{x, m\} \rightarrow y$. Also, the discriminator D learns to classify $\hat{y} = G(x, m)$ patches as "fake" from $\hat{y}$, m and x, and the patches of y as "real" from y, m and x, $D : \{x, y/\hat{y}, m\} \rightarrow real/fake$.

In most of the training dataset images, the relationship between the static and dynamic regions sizes is unbalanced, *i.e.*, static regions occupy usually a much bigger area. This leads us to believe that the influence of dynamic regions on the final loss is significantly reduced. As a solution to this problem, we propose to re-formulate the *cGAN* and $L1$ losses so that there is more emphasis on the main areas that have to be inpainted, according to Eqns. 3.3 and 3.4. The weights w are computed as $w = \frac{N}{N_{dyn}}$ if $m = 1$ (dynamic object), and as $w = \frac{N}{N - N_{dyn}}$ if $m = 0$ (background). N stands for the number of elements in the binary mask m, and N_{dyn} means the number of pixels where $m = 1$.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y}[w \cdot \|y - G(x, m)\|_1], \tag{3.3}$$

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[w \cdot \log D(x, y, m)] + \mathbb{E}_x[w \cdot \log (1 - D(x, G(x, m), m))], \tag{3.4}$$

An important feature that we have also incorporated to the framework is the computation of our output and target images "noise". This is motivated by the use of the noise domain for steganalysis to detect if an image has been tampered or not. Fig. 3.6 shows an example of why working in the noise domain is helpful for detecting "fake" images. While the static generated image (Fig. 3.6b) looks visually similar to its target image (Fig. 3.6d), their computed noises (Figs. 3.6c and 3.6e) are very different. It would be very easy for us, humans, to tell what parts of the original image (Fig. 3.6a) have been changed by analyzing their noise mapping. In the same way, the discriminator could more easily learn to distinguish "real" from "fake" images if it can take as input their noise. This idea is explained more in depth in the following block, and the whole training procedure is diagrammed in Fig. 3.7. To the best of our knowledge, steganalysis noise features have not been used before in the context of GANs.



(a) (b) (c) (d) (e)

Figure 3.6: (b) shows the image generated by our framework when taking (a) as input. (d) is the static objective image. (b) and (d) are visually similar, but their computed noise ((c) and (e) respectively) clearly show what image and what parts of it have been modified the most. The noise magnitude has been amplified ($\times$ 10) for visualization.

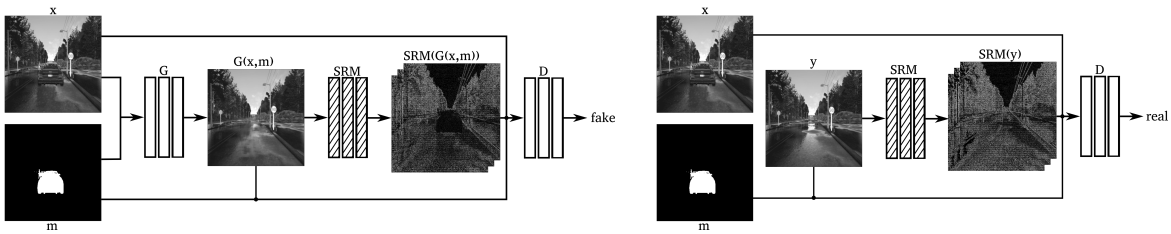


Figure 3.7: The discriminator D has to learn to differ between the real images y and the images generated by the generator $G(x, m)$. D makes a better decision (*real / fake*) by seeing the inputs of the generator x and m, and by seeing the SRM noise features of $G(x, m)$ and y. The striped blocks are convolutional layers whose weights do not require upgrading during training time.

This GAN training setup leads to having good inpainting results. However, despite the efforts of the discriminator to catch the high frequency of the "real" images, the outputs of our framework are still slightly blurry. One of the objectives of this work is to use our images for localization tasks, therefore, if the inpainted regions are somewhat blurry, features would not be extracted in these areas. Image features are important for localization since many visual SLAM systems rely on them as their core –ORB-SLAM (Mur-Artal et al. (2015))–. Having blurriness in inpainted areas could be seen as a good feature of our framework for navigation because it would allow feature-based localization systems to work with our images without any modification in their architecture, and "fake" features would not be introduced. This would be equivalent to modifying the utilized localization system to work with the raw images and the dynamic/static binary masks. We have proved with our localization experiments (Subsection 3.5.1) that not utilizing dynamic objects' features leads to worse tracking results than working with fully static images. For that reason, we want to exploit our framework for obtaining both high-quality inpainting results, and to succeeding in generating reliable features for visual localization tasks. Fortunately, these two assignments are highly related. Solving one of them leads to having the other one tackled. Therefore, we have implemented a new loss based on ORB features (Rublee et al. (2011)). That is, we want the output of our generator G to have the same ORB features than its target image, while keeping it realistic and close to its target in a $L1$ sense. By the same ORB features we mean the same detected keypoints with their same orientation and descriptors, following ORB's implementation to the extent possible. This procedure is further described in Subsection 3.3.2.

Steganalysis-Based Loss

With the advances of image editing techniques, tampered or manipulated image generation processes have become widely available. As a result, distinguishing authentic images from tampered images has become increasingly challenging.

What our framework is actually trying to achieve is to eliminate certain regions from an authentic image followed by inpainting, *i.e.*, *removal*, one of the most common image manipulation techniques. It is the discriminator's job to classify the generated image patches as tampered –fake– or real.

Images have a low-frequency component dependent on their content, and a high-frequency component dependent on their source. These high-frequency components are

known as noise features or noise residuals, and can be extracted using linear and non-linear high-pass filters. Recent works on image forensics utilize noise features as clues to classify a specific patch or pixel in an image as tampered or not, and localize the tampered regions (Fridrich & Kodovsky (2012), Zhou, Han, Morariu & Davis (2018)). The intuition behind this idea is that when an object is removed from one image (source) and the gap is inpainted (target), the noise features between the source and target are unlikely to match.

To provide the discriminator with better clues to distinguish real from fake inputs, we first extract the noise features from our images and concatenate them to the gray-scale images, as depicted in Fig. 3.7. The cGAN's objective is re-formulated as

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[w \cdot \log D(x, y, m, n)] + \mathbb{E}_x[w \cdot \log (1 - D(x, \hat{y}, m, \hat{n}))], \qquad (3.5)$$

where $n = SRM(y)$ and $\hat{n} = SRM(\hat{y})$. There are many ways to produce noise features from an image. Inspired by recent progress on Steganalysis Rich Models (SRM) for image manipulation detection (Fridrich & Kodovsky (2012)), we use SRM filter kernels to extract the local noise features from the static images as the input to our discriminator. The SRM use statistics of neighboring noise residual samples as features to capture the dependency changes caused by embedding. Zhou, Han, Morariu & Davis (2018) use SRM residuals, together with the RGB image to detect and localize corrupted regions in images. They only use three SRM kernels, instead of thirty –as in the original work of Fridrich & Kodovsky (2012)–, and claim that they achieve comparable performance. Similarly, we use these same three filters (Fig. 3.8), setting the kernel size of the SRM filter layer to be $5 \times 5 \times 3$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 2 & -4 & 2 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{bmatrix}$$

Figure 3.8: The three utilized SRM kernels to extract noise features. The left kernel is useful in regions with a strong gradient. The middle and the rightmost kernels provide the layer with a high shift-invariance.

ORB-Features-Based Loss

ORB features allow real-time detection and description, and provide good invariance to changes in viewpoint and illumination. Furthermore, they are useful for visual SLAM and place recognition, as demonstrated in the popular ORB-SLAM (Mur-Artal et al. (2015)) and its binary bag of visual words (Gálvez-López & Tardos (2012)). The following paragraphs summarize how the ORB features detector, descriptors and orientation are computed, and how we have adapted them into a new loss.

Detector: The ORB Detector is based on the FAST algorithm (Rosten & Drummond (2006)). It takes one parameter, the intensity threshold t between the center pixel p, I_p, and those in a circular ring around the center. If there exists a set of contiguous pixels in the circle which are all brighter than $I_p + t$, or all darker than

$I_p - t$, the pixel p would be a key point candidate. Then the Harris corner measure is computed for each of these candidates, and the target N key points with the highest Harris measure are finally selected. FAST does not produce multi-scale features, therefore, ORB uses a scale pyramid of the image and extracts FAST features at each level in the pyramid.

To bring this to a differentiable solution, we have defined a convolution capable of detecting corners in an image in the same way that FAST does it. We have approximated the FAST corner detection and have used instead a convolution with the kernels shown in Fig. 3.9. These images show some of the kernels used for corners detection for a circular ring of 3 pixels around the center. By convolving the image with these kernels for different kernel sizes, we obtain its corner response for the different image pyramid levels. We keep the maximum score per pixel and per level, and we then raise each element to its 2nd power –to equally leverage positive and negative responses– and subtract a value, equivalent to the FAST threshold t. This is followed by a sigmoid operation. Its output is the probability of a pixel of being a FAST feature, and could also be seen as the Harris corner measure. The features for the output and target images are computed following this procedure. We define this network as *det*, and the corresponding loss $\mathcal{L}_{det}(G)$ can be expressed as

$$\mathcal{L}_{det}(G) = -\mathbb{E}_{x,y}[w_{det} \cdot (det(y) \cdot \log(det(\hat{y})) + (1 - det(y)) \cdot \log(1 - det(\hat{y})))], \quad (3.6)$$

where $\hat{y} = G(x, m)$ and w_{det} is calculated following Eqn. 3.7. This weights definition allows us to leverage the uneven distribution of non-features and features pixels, and to affect only those image regions with a wrong feature response. N stands for the number of pixels in the features map, and N_f represents the number of pixels in the response map $det(y)$ where $det(y) > 0.5$, *i.e.*, the number of FAST features in the current objective frame.

$$w_{det} = \begin{cases} \frac{N}{N_f}, & det(y) > 0.5 \quad \text{and} \quad det(\hat{y}) \leq 0.5 \\ \frac{N}{N - N_f}, & det(y) \leq 0.5 \quad \text{and} \quad det(\hat{y}) > 0.5 \\ 0, & \text{otherwise} \end{cases} \quad (3.7)$$

Figure 3.9: A subset of the 16 kernels used to obtain corner responses in the images. The 12 black pixels have a value of $-1/12$, the gray pixels are set to 0, and the white pixel is set to 1. A very positive or a very negative response will be obtained when convolving these kernels with a corner area in an image.

According to our results, the optimum number of image pyramid levels for this objective is 1. More levels lead to a greater training time and the results are barely influenced. This is coherent with the idea that we want to maximize the sharpness of small features rather that of the big corners. These convolutions have been applied with a stride of 5, offering a good trade-off between computational training time and good-quality results.

Other approaches have tried before to include a similar loss inside a CycleGAN framework: the work by Porav et al. (2018) uses the SURF detector (Bay et al. (2008)), which is already differentiable, but does not compute a binary loss. They compute a more traditional L1 loss between the blob responses of both output and ground-truth images. Computing a binary loss as in Eqn. 3.7 allows us to have more emphasis on the high-gradient areas.

Orientation: Once FAST features have been detected, the original ORB work extracts their orientation to provide them with rotation invariance. This is done by computing its orientation $\theta = \text{atan2}(m_{01}, m_{10})$ and its intensity centroid $C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}})$, where $m_{pq} = \sum_{x,y} x^p y^q I(x,y)$ are the moments of an image patch. More precisely, the three utilized patch moments are $m_{10} = \sum_{x,y} x \cdot I(x,y)$, $m_{01} = \sum_{x,y} y \cdot I(x,y)$ and $m_{00} = \sum_{x,y} I(x,y)$. We have created three 14-pixel-radius circular kernels with the values x, y and 1 respectively for m_{10}, m_{01} and m_{00} (centered in 0), so that when convolving the image with them, we obtain their respective patch moments m_{01}, m_{10} and m_{00}. We define this network as ori, and the objective of its corresponding loss is that the "fake" static image detected features, $det(G(x,m))$, have the same orientation parameters m_{01}, m_{10} and m_{00} than the ground-truth static image detected features, $det(y)$. This loss can be expressed as

$$\mathcal{L}_{ori}(G) = -\mathbb{E}_{x,y}[w_{ori} \cdot \|ori(y) - ori(\hat{y})\|_1]. \tag{3.8}$$

Even though these convolutions are applied to the whole image with a stride of 5 –as in the detection loss–, the weighting term w_{ori} in Eqn. 3.8 has a value of 1 if a FAST feature has been detected in either the ground-truth static image or the output image, *i.e.*, if $det(y) > 0.5 \quad \text{or} \quad det(\hat{y}) > 0.5$. Otherwise, the weighting term w_{ori} is 0.

Descriptor: The ORB descriptor is a bit string description of an image patch constructed from a set of binary intensity tests. Consider a smoothed image patch, $\mathbf{p}$, a binary test τ is defined by

$$\tau(\mathbf{p}; x, y) = \begin{cases} 1, & \mathbf{p}(x) < \mathbf{p}(y) \\ 0, & \mathbf{p}(x) \geq \mathbf{p}(y) \end{cases} \tag{3.9}$$

where $\mathbf{p}(x)$ is the intensity of $\mathbf{p}$ at a point x. The feature is defined as a vector of n binary tests

$$f_n(\mathbf{p}) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; x_i, y_i) \tag{3.10}$$

As in the work of Rublee et al. (2011), we use a Gaussian distribution around the center of the patch and a vector length $n = 256$. This can be achieved in a differentiable and convolutional manner by creating n kernels with all values set to 0 except for those in the positions x and y:

$$\mathbf{k}(z) = \begin{cases} 1, & z = x \\ -1, & z = y \\ 0, & \text{otherwise} \end{cases} \tag{3.11}$$

where $\mathbf{k}(z)$ is the value of the kernel $\mathbf{k}$ at a point z. Convolving an image with these n kernels yields each pixel's ORB descriptor –a negative output corresponds to the bit value 0 and a positive one to 1–. This convolution is followed by a sigmoid activation

function. We define this network as *desc*, and the corresponding loss $\mathcal{L}_{desc}(G)$ can be expressed as

$$\mathcal{L}_{desc}(G) = -\mathbb{E}_{x,y}[w_{desc} \cdot (desc(y) \cdot \log(desc(\hat{y})) + (1 - desc(y)) \cdot \log(1 - desc(\hat{y})))], \quad (3.12)$$

where the weights w_{desc} are defined in Eqn. 3.13. This descriptors loss is back-propagated to the whole image, whether a feature has been detected or not, as it helps keeping the image statistics.

$$w_{desc} = \begin{cases} 1, & desc(y) > 0.5 \ \ \& \ \ desc(\hat{y}) \leq 0.5 \\ 1, & desc(y) \leq 0.5 \ \ \& \ \ desc(\hat{y}) > 0.5 \\ 0, & \text{otherwise} \end{cases} \quad (3.13)$$

All these losses are combined into one loss $\mathcal{L}_{ORB}(G)$, that is computed as in Eqn. 3.14. The values for the weights of the different losses λ_{det}, λ_{ori} and λ_{desc} have been chosen empirically, and they are set to 10, 0.1 and 1 respectively.

$$\mathcal{L}_{ORB}(G) = \lambda_{det}\mathcal{L}_{det}(G) + \lambda_{ori}\mathcal{L}_{ori}(G) + \lambda_{desc}\mathcal{L}_{desc}(G) \quad (3.14)$$

The features detection, orientation and descriptor maps can be computed in a parallel way to decrease the training time, since their computation is not necessarily sequential. Finally, the generator's job can be expressed as in Eqn. 3.15.

$$G^* = \arg \min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda_1 \cdot \mathcal{L}_{L1}(G) + \mathcal{L}_{ORB}(G) \quad (3.15)$$

As an implementation detail, we have first trained the whole system without the ORB loss for 125 epochs, and then have fine-tuned including it for another 25 epochs.

Semantic Segmentation

Semantic segmentation is a challenging task that addresses most of the perception needs of intelligent vehicles in a unified way. Deep neural networks excel at this task, as they can be trained end-to-end to accurately classify multiple object categories in an image at pixel level. However, very few architectures have a good trade-off between high quality and computational resources. The recent work of Romera et al. (2018) runs in real time while providing accurate semantic segmentation. The core of their architecture (ERFNet) uses residual connections and factorized convolutions to remain efficient while retaining remarkable accuracy.

Romera et al. (2018) have made public some of their trained models. We use for our approach the ERFNet model with encoder and decoder both trained from scratch on Cityscapes train set (Cordts et al. (2016)). We have fine tuned their model to adjust it to our inpainting approach by back-propagating the loss of the semantic segmentation $\mathcal{L}_{CE}(SS)$, calculated with the cross entropy criterion using the class weights they suggest, w_{SS}, and the adversarial loss of our final inpainting model $\mathcal{L}_{cGAN}(G, D)$. The semantic segmentation network's job (SS) can be hence expressed as:

$$SS^* = \arg \min_{SS} \max_D \mathcal{L}_{cGAN}(G, D) + \lambda_2 \cdot \mathcal{L}_{CE}(SS), \quad (3.16)$$

where $\mathcal{L}_{CE}(SS) = w_{SS}[class] \cdot (log \sum_j \exp(y_{SS}[j]) - y_{SS}[class])$. Its objective is to produce an accurate semantic segmentation y_{SS}, but also to fool the discriminator D. The latter objective might occasionally lead the network to not only recognize dynamic objects but also their shadows.

Dynamic Object Semantic Segmentation

Once the semantic segmentation of the RGB image is done, we can select those classes known to be dynamic (vehicles and pedestrians). This has been done by applying a *SoftMax* layer, followed by a convolutional layer with a kernel of $n \times 1 \times 1$, where n is the number of classes, and with the weights of those dynamic and static channels set to w_{dyn} and w_{stat} respectively. With $w_{dyn} = \frac{n-n_{dyn}}{n}$ and $w_{stat} = -\frac{n_{dyn}}{n}$, where n_{dyn} stands for the number of dynamic existing classes. A positive output corresponds to a dynamic object, whereas a negative one corresponds to a static one.

The consequent output passes through a $Tanh$ layer to obtain the wanted dynamic/static mask. Note that the defined weights w_{dyn} and w_{stat} are not changed during training time.

3.4 Image-Based Experiments

3.4.1 Data Generation

We have analyzed the performance of our method using CARLA (Dosovitskiy et al. (2017)). CARLA is an open-source simulator for autonomous driving research, that provides open digital assets (urban layouts, buildings, vehicles, pedestrians, *etc.*). The simulation platform supports flexible specification of sensor suites and environmental conditions. We have generated over 12000 image pairs consisting of a target image captured with neither vehicles nor pedestrians, and a corresponding input image captured at the same pose with the same illumination conditions, but with cars, tracks and people moving around. These images have been recorded using a front and a rear RGB camera mounted on a car. Their ground-truth semantic segmentation has also been captured. CARLA offers two different towns that we have used for training and testing, respectively. Our dataset, together with more information about our framework, is available on `https://bertabescos.github.io/EmptyCities_SLAM/`.

At present, we are limited to training this framework on synthetic datasets since, to our knowledge, no real-world dataset exists that provides RGB images captured under same illumination conditions at identical poses, with and without dynamic objects. In order to render our framework –trained on synthetic data– transferable to real-world data, we have fine-tuned our models with data from the Cityscapes and KITTI semantic segmentation training datasets (Cordts et al. (2016), Geiger et al. (2013)). These datasets are semantically similar to the ones synthesized with CARLA. Nonetheless, their image statistics are different. We further explain this fine-tuning process in subsection 3.4.3.

3.4.2 Inpainting

In this subsection we report the improvements achieved by our framework for inpainting. Table 3.2 shows the ablation study of our work for the different reported inputs and losses. The existence of many possible solutions renders difficult to define a metric to evaluate image inpainting (Yu et al. (2018)). Nevertheless, we follow previous works and report the L_1, $PSNR$ and $SSIM$ errors (Wang et al. (2004)), as well as a feature-based metric $Feat$. This last metric computes the FAST features detection as

explained in Subsection 3.3.2 for the output and ground-truth images, and compares them, similarly to Eqn. 3.6.

Experiment		$G(x)$ $D(x,y)$	$G(x,m)$ $D(x,y)$	$G(x,m)$ $D(x,y,m)$	$G(x,m)\|^w$ $D(x,y,m)\|^w$	$G(x,m)\|^w$ $D(x,y,m,n)\|^w$	$G(x,m)\|^w_{ORB}$ $D(x,y,m,n)\|^w$
$L_1(\%)$	*Full image*	1.98	2.13	3.03	2.00	**0.96**	1.46
	In	10.04	7.70	7.37	5.73	**4.99**	5.16
	Out	1.70	1.95	2.92	1.88	**0.80**	1.32
Feat	*Full image*	1.29	1.95	1.44	1.45	0.80	**0.70**
	In	8.75	7.48	7.92	5.75	5.84	**5.14**
	Out	1.04	1.77	1.26	1.33	0.62	**0.53**
PSNR	*Full image*	29.89	29.85	28.31	30.46	**34.03**	33.03
	In	17.39	20.30	20.58	22.56	23.25	**23.29**
	Out	32.77	31.23	29.05	31.35	**36.47**	34.74
SSIM	*Full image*	0.985	0.985	0.981	0.987	**0.995**	0.993
	In	0.151	0.333	0.488	0.613	0.646	**0.655**
	Out	0.993	0.989	0.985	0.990	**0.997**	0.995

Table 3.2: Quantitative evaluations of our contributions in the inpainting task on the test synthetic images. The best results for almost all the inpainting metrics (L_1, $PSNR$ and $SSIM$) are obtained with the generator $G(x,m)|^w$ and the discriminator $D(x,y,m,n)|^w$. More correct features ($Feat$ metric) are detected though when adding the features based loss $G(x,m)|^w_{ORB}$. *Full image* designates the per-pixel error considering the whole image. For In and Out we refer respectively to the error per pixel considering the masked and unmasked pixels.

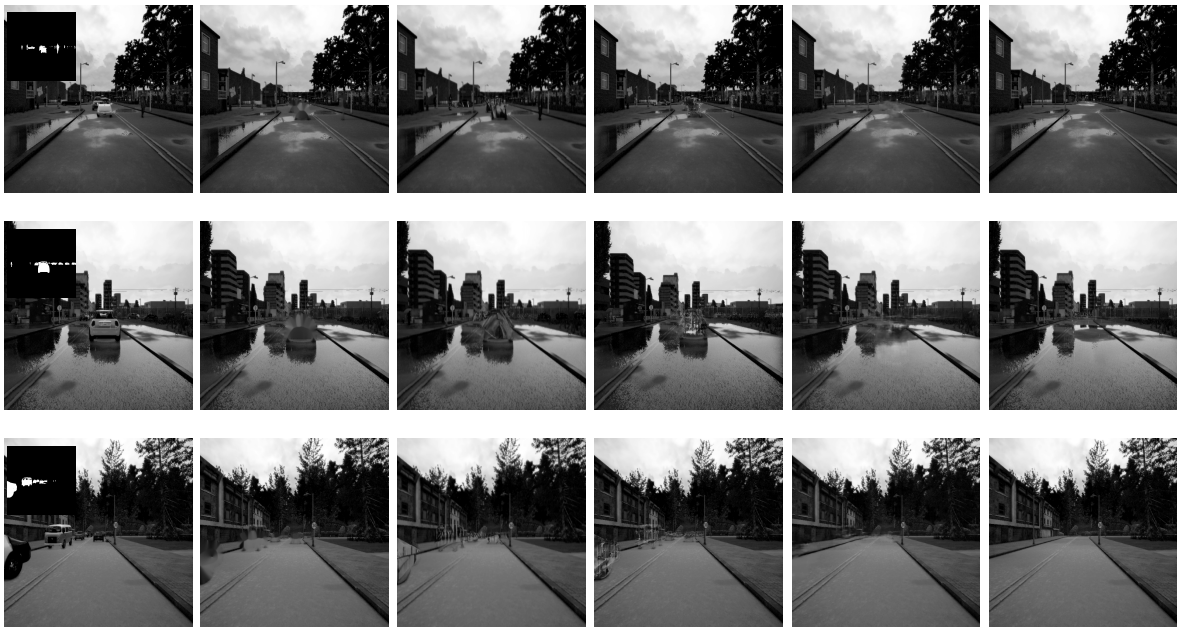
Adding the dynamic/static mask as input for both the generator and discriminator helps obtaining better inpainting results within the images hole regions –In–, at the expense of having worse quality results in the non-hole regions –Out–. Leveraging the unbalanced quantity of static and dynamic data within the dataset with w (Eqns. 3.3 and 3.4) helps obtaining better results too. Providing the GAN's discriminator with the images noise makes it learn better to distinguish between real and fake images, and therefore the generator learns to produce more realistic images. One could say that the SRM filters in Fig. 3.8 might already be learnt implicitly by the discriminator. With this loss we are helping the discriminator so that it requires less training data or less iterations. The ORB-based loss leads to slightly worse inpainting results – according to L_1, PSNR and SSIM metrics–, but renders this approach more useful for both localization and mapping tasks since more correct features are created.

Baselines for Inpainting

We compare qualitatively and quantitatively our image-based "inpainting" method with four other state-of-the-art approaches: two traditional non-learning approaches: Bertalmio et al. (2001) and Telea (2004), and two deep learning based methods: Iizuka et al. (2017) and Yu et al. (2018).

For a fair comparison, we have trained the approach by Yu et al. (2018) with our same training data. Iizuka et al. (2017) do not have their training code available and we have directly used their release model trained on the Places2 dataset (Zhou et al. (2017)). This dataset contains images of urban streets from a car perspective similar to ours. A more direct comparison is not possible. We provide them with the same mask than to our method to generate the holes in the images. We evaluate qualitatively on the 3000 images from our synthetic test dataset, on the 500 validation images from the

Cityscapes dataset (Cordts et al. (2016)) and on the images from the Oxford Robotcar dataset (Maddern et al. (2017)). We can see in Figs. 3.10, 3.11 and 3.12 the qualitative comparisons on these three datasets. Note that results generated with the two learning methods –Iizuka et al. (2017) and Yu et al. (2018)– have been generated with the color images and then converted to gray scale for visual comparison. Visually, we see that our method obtains a more realistic output –these results are computed without the ORB loss for an inpainting oriented comparison–. Also, it is the only one capable of removing the shadows generated by the dynamic objects even though they are not included in the dynamic/static mask (Fig. 3.10 row 2 and Fig. 3.12 row 1). The utilized masks are included in the images in Figs. 3.10a and 3.12a respectively.



(a) Input. (b) Telea (2004). (c) Yu et al. (2018). (d) Iizuka et al. (2017). (e) Ours. (f) Target.

Figure 3.10: Qualitative comparison of our method (e) against other inpainting techniques (b), (c), (d) on our synthetic dataset. Our results are semantically and geometrically coherent, and do not show the dynamic objects' shadows, even if they are not included in the input mask.

Table 3.3 shows the quantitative comparison of our method against the four presented state-of-the-art inpainting approaches on our CARLA dataset. It is not possible to quantitatively measure the performance of the different methods on the Cityscapes and Oxford Robotcar datasets, since ground-truth does not exist. By following these results, we can claim that our method outperforms both qualitatively and quantitatively the other approaches.

As seen in Fig. 3.12 row 1, the fact that our method does not perform pure inpainting but image-to-image translation with the help of a dynamic/static mask allows us to modify not only the dynamic objects themselves, but also their shadows or reflections. We want to highlight the importance of the inpainting robustness to inaccurate segmentation masks since, in practice, partial or missing segmentation happens frequently. Empty Cities cannot handle missing detections but can cope with partial segmentations covering at least the 85 % of the object image. We hereby report some

(a) Input. (b) Telea (2004). (c) Yu et al. (2018). (d) Ours.

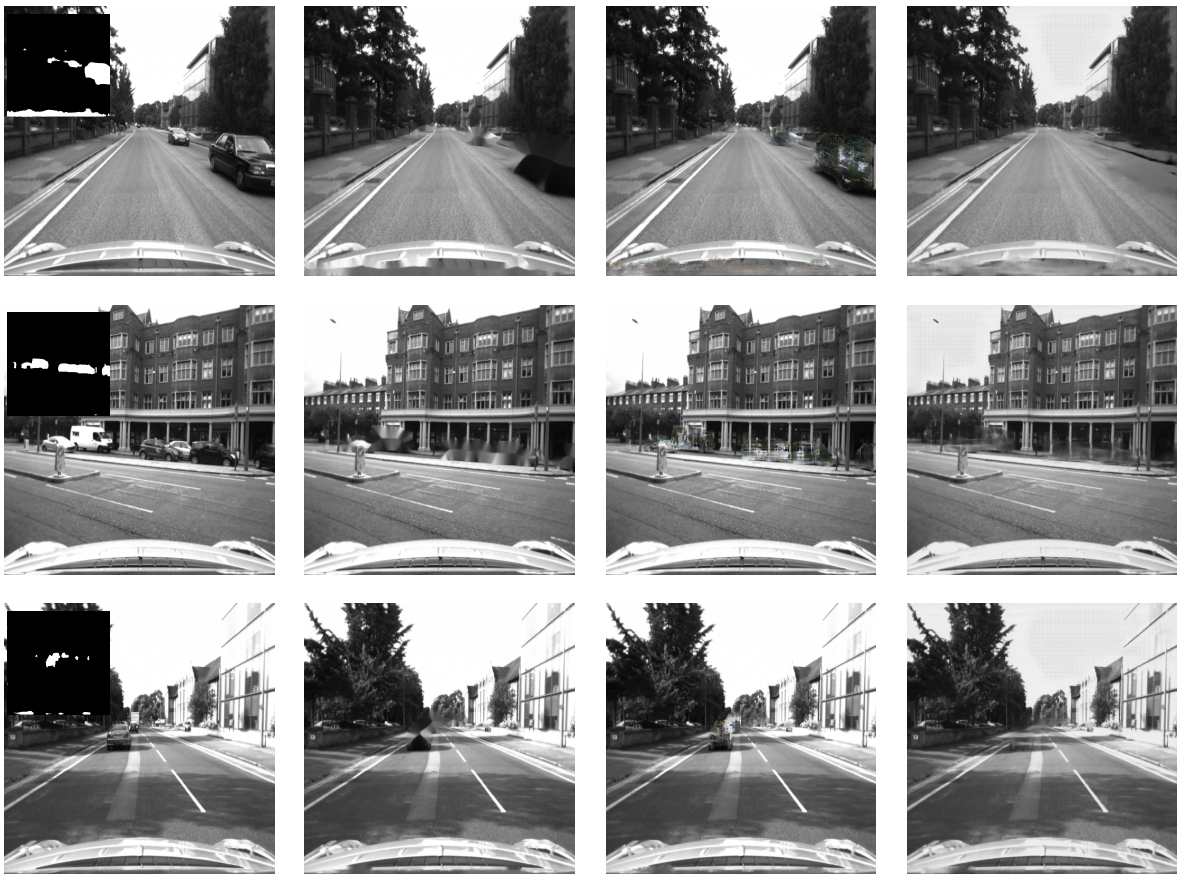
Figure 3.11: Comparison of our method (d) against other image inpainting approaches (b), (c) on the Cityscapes validation dataset (Cordts et al. (2016)). (c) and (d) show the results when real images have been incorporated into our training set together with the synthetic images with a ratio of 1/10.

	Telea (2004)	Bertalmio et al. (2001)	Yu et al. (2018)	Iizuka et al. (2017)	Ours
$L_1(\%)$	6.55	6.69	8.91	9.50	**4.99**
$Feat$	6.24	6.33	10.03	10.63	**5.84**
$PSNR$	21.03	20.90	18.16	18.23	**23.25**
$SSIM$	0.479	0.467	0.309	0.316	**0.646**

Table 3.3: Quantitative results of our method against other inpainting approaches in our CARLA dataset. For a fair comparison, we only report the different errors within the images' hole regions since the other methods are conceived to only significantly modify such parts.

metrics evaluating how our framework behaves with the dynamic objects' shadows.

Thresholding the difference between the dynamic and static image, and subtracting the dynamic-objects mask yields its dynamic objects shadows and reflections mask. We have first generated the shadows ground truth of our CARLA dataset, and then computed the shadows masks for our inpainted images in the same way. The in-



(a) Input. (b) Telea (2004). (c) Yu et al. (2018). (d) Ours.

Figure 3.12: Comparison of our method (d) against other image inpainting approaches (b), (c) on the Oxford Robotcar dataset (Maddern et al. (2017)). (c) and (d) show the results when Cityscapes images have been incorporated into our training set together with the synthetic images with a 1/10 ratio. The binary dynamic/static mask computed for every input image has been added on the top left corner of every raw.

tersection over union of the estimated shadows against the ground truth is 42.8 %. Following recent works in shadow detection (Hosseinzadeh et al. (2018)), we also report our method's shadow, non-shadow and total accuracy –59.7 %, 99.8 % and 99.5 % respectively–. With this framework, we can remove almost 50 % of the shadows of the dynamic objects of our CARLA test dataset. Admitting this could be improved, our method's non-shadow accuracy is almost 100 %, which means that it does not modify other objects' shadows.

3.4.3 Transfer to Real Data

Models trained on synthetic data can be useful for real world vision tasks (Gaidon et al. (2016), Peris et al. (2012), Skinner et al. (2016), Tobin et al. (2017)). Accordingly, we provide a study of synthetic-to-real transfer learning using data from the Cityscapes dataset (Cordts et al. (2016)), which offers a variety of urban real-world environments similar to the synthetic ones.

When testing our method on real data, we see qualitatively that the synthesized images show some artifacts. This happens because such data has different statistics

than the real one, and therefore cannot be easily used. The combination of real and synthetic data is possible during training despite the lack of ground-truth static real images. In the case of the real images, the network only learns the texture and the style of the static real world by encoding its information and decoding back the original image non-hole regions. The synthetic data is substantially more plentiful and has information about the inpainting process. The rendering, however, is far from realistic. Thus, the chosen representation attempts to bridge the reality gap encountered when using simulated data, and to remove the need for domain adaptation.

We provide implementation details: we have finetuned our model with real data for 25 epochs with a real/synthetic images ratio of 1/10. On the one hand, for every ten images there are nine synthetic images that provide our model with information about the inpainting task. On the other hand, one image out of those ten is a real image from the Cityscapes train dataset. There is groundtruth of its semantic information but there is no groundtruth of its static representation. In such cases we do backpropagation of the loss derivative only on those image areas that we consider as static. This way, the model can learn both the inpainting task –from the synthetic images–, and the static real-world texture.

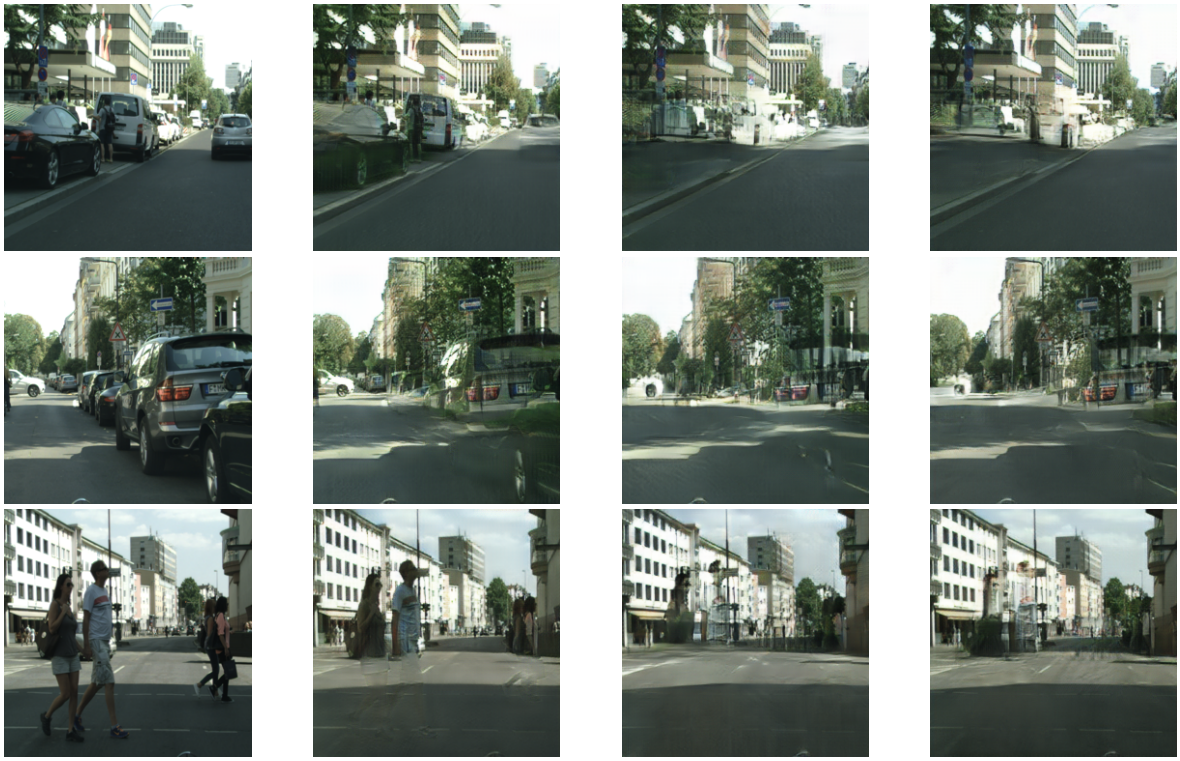
Dynamic to Static Image Translation via CycleGANs

We have also explored the task of translating dynamic images into static by means of CycleGANs (Zhu et al. (2017)). A CycleGAN uses cycle-consistent adversarial neural networks and is specifically designed to work with unpaired image datasets. This allows us to work with real-world images instead of synthetic ones, removing the need for synthetic-to-real domain adaptation. In pursuit of learning a mapping from dynamic to static images, this framework also learns an inverse mapping, that is, a mapping from static to dynamic images. The better one of the two mappings works, the better the other one does. For more information about the functionality of this framework, we refer the reader to the work by Zhu et al. (2017).

To create a suitable dataset for our specific task, we have made use of the Cityscapes dataset (Cordts et al. (2016)) and have chosen the images with the highest and the lowest dynamic content. This has been made thanks to the ground-truth semantic segmentation that the dataset authors provide. We have then created a dataset with a training set of 1185 dynamic images and 1049 static images, and a test set of 199 and 148 dynamic and static images respectively.

Dynamic object masks and the ORB-feature-based loss have been specially considered to turn the base CycleGAN framework from a general image-to-image translation framework into a dynamic-to-static one. This is done by closely following the approaches in Subsection 3.3.2. One can see in Fig. 3.13 the qualitative results obtained by the CycleGAN framework with the different incremental improvements for both learnt mappings: dynamic to static and static to dynamic. The addition of the dynamic/static masks during both training and testing brings remarkable improvements for the two mappings. The gains that the ORB-based loss brings are more subtle though but render the output image more realistic.

CycleGANs are known to be masters of steganography (Chu et al. (2017)) since they learn to hide information about the source image into the generated image in a nearly imperceptible, high-frequency signal. This trick ensures that the generator can recover the original sample and thus satisfy the cyclic consistency requirement, while



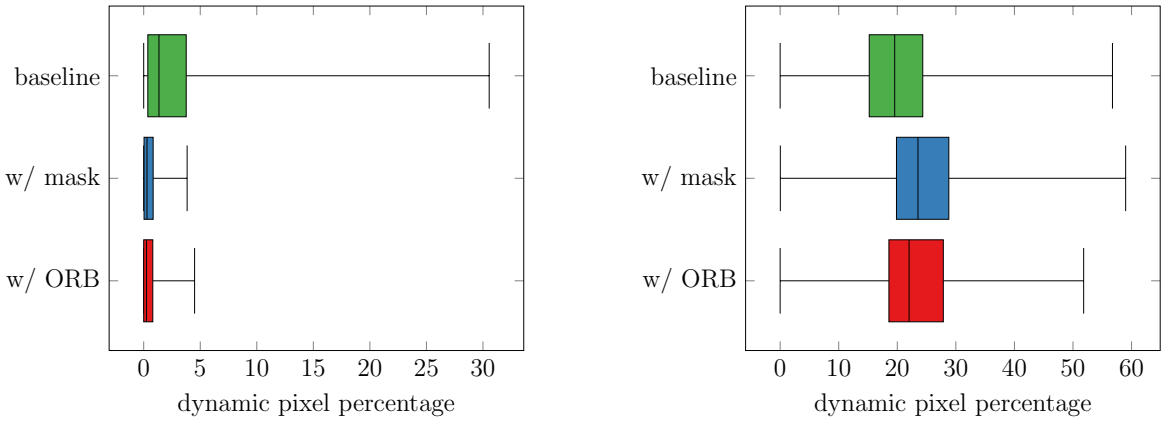
(a) Dynamic input. (b) Output. (c) Output w/ mask. (d) Output w/ ORB.

(e) Static input. (f) Output. (g) Output w/ mask. (h) Output w/ ORB.

Figure 3.13: Qualitative results of the modified CycleGAN framework trained with our dynamic/static dataset. One can see the results of the dynamic-to-static and the static-to-dynamic image translation in the first and second three rows respectively.

the generated image remains realistic. This can be also appreciated in our resulting images, for example, in the first and second rows of Fig. 3.13d the rear lights of the white van and the black car respectively are blended with the static background. Also, the windows of the building in the last raw become the car windows. Our conclusion is that even though the static generated images are highly realistic, the CycleGAN framework does not perform inpainting in the sense of propagating the static structure into the dynamic region, but keeps the dynamic structure and renders the background with the texture of the static image areas.

As a quantitative measure of the quality of these generated images we have used the semantic segmentation network ERFNet (Romera et al. (2017)) to classify each pixel in the generated image. We want to analyze if the new dynamic and static areas in the images are identified as such. The percentage of dynamic pixels of the resulting images of the dynamic-to-static image translation can be seen in Fig. 3.14a, whereas the dynamic pixel percentage of the images coming from the static-to-dynamic image translation are seen in Fig. 3.14b. With our incremental improvements on top of the CycleGAN framework the number of dynamic pixels tends to zero in the dynamic to static mapping, and gets higher in the static to dynamic mapping. We can then conclude that both our CycleGAN model and the semantic network implicitly learn very similar hidden features.



(a) Dynamic-to-static image translation.

(b) Static-to-dynamic image translation.

Figure 3.14: Quantitative results of our modified CycleGAN framework trained with our dynamic / static dataset. (a) shows the dynamic pixel percentage obtained by the different frameworks in our dynamic-to-static test dataset (the lower the better). On the other hand, (b) shows the dynamic pixel percentage in our static-to-dyanamic test dataset (the higher the better).

The static generated images with this framework turn out to be highly realistic. However, the results differ from the idea of inpainting, *i.e.*, the static image structure is not propagated into the image dynamic region. Also, our experiments are preliminary since they are performed with 128×128 resolution images. Visual odometry, place recognition and mapping applications usually require images with a higher resolution as input. The achieved results have potential but for the rest of the chapter we report the experiments for the Empty Cities framework with domain adaptation as explained at the beginning of this subsection.

3.5 Experiments

3.5.1 Visual Odometry

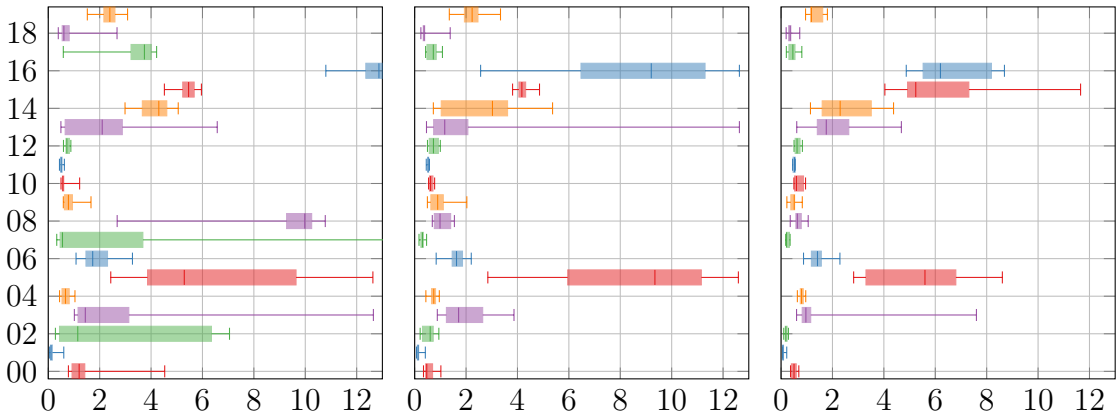
We have evaluated Empty Cities on 20 CARLA synthetic sequences and on 9 sequences from real world new environments. For these visual odometry experiments we have chosen the state-of-the-art feature-based system ORB-SLAM by Mur-Artal & Tardós (2017)[1] and the direct method DSO by Engel et al. (2017). The former system is ideal to test the influence of our ORB features loss, and the latter is useful to prove that other –very different– systems can also benefit from this approach.

Baselines for VO in our Synthetic Dataset

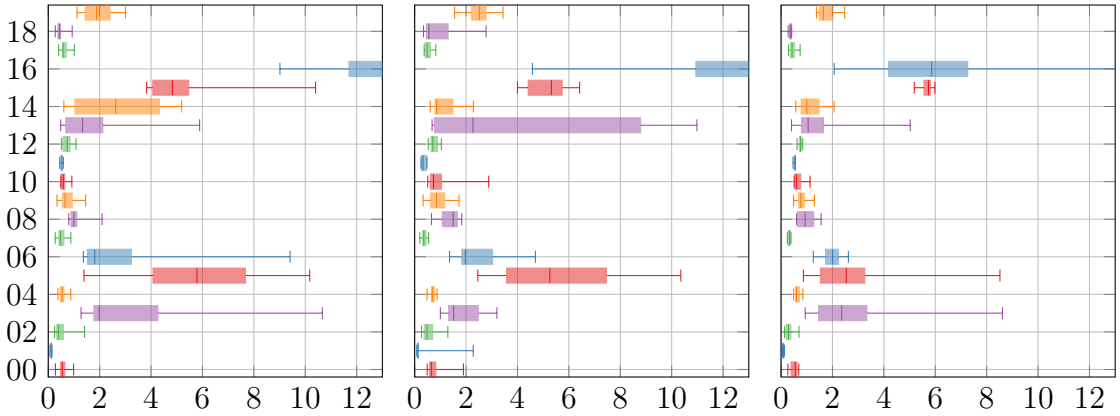
Fig. 3.15 displays the ORB-SLAM absolute trajectory RMSE [m] computed for 20 CARLA sequences of approximately 100 m long without loop closures. Fig. 3.15a shows the results when many vehicles and pedestrians are moving independently –more precisely, the number of vehicles and pedestrians have been set to the maximum allowed by CARLA–, and Fig. 3.15f shows the same odometry results for the ground-truth static sequences. We can see that dynamic objects have in many sequences a big influence on ORB-SLAM's performance –sequences 02, 07, 08, *etc.*–. Fig. 3.15b shows the trajectory error obtained with our previous system DynaSLAM (Bescos et al. (2018)). This system is based on ORB-SLAM and uses the semantic segmentation network Mask R-CNN (He et al. (2017)) to detect the moving objects and not extract ORB features within them. Even though better odometry results are obtained –compared against using the raw dynamic images–, this experiment shows that using static images leads to a more accurate camera tracking. One reason for this is that dynamic objects might occlude the nearby regions in the scene, which are the most reliable for camera pose estimation. Another reason might be that using dynamic objects masks does not yield anymore a homogeneous features distribution within the image. ORB-SLAM looks for a uniform distribution of image features. Pose optimization could be degraded and drifting could increase if the features do not follow such distribution. Finally, Fig. 3.15c shows the ORB-SLAM error when using our inpainted images. Our odometry results show that better results are usually obtained when using our inpainted images. The inpainting is realistic enough to provide the visual odometry system with consistent features that are useful for localization.

We want to highlight the importance of the influence of using the ORB loss during training (Fig. 3.16). Figs. 3.16a and 3.16b present the ATE obtained by ORB-SLAM with our inpainted images without and with the ORB loss respectively. The estimated errors are smaller and more constant when using this loss. This performance gain can be due to features in the regions that originally contained dynamic objects, and to more stable features in the static-content regions. Fig. 3.16c presents the DynaSLAM ATE with the inpainted images generated with our model trained with this loss term. We expect these errors to be very similar to those shown in Fig. 3.15b, and slightly bigger than those in Fig. 3.16b. This experiment shows that our model barely damages the static content of the scene, keeping the static features as they used to be. It also demonstrates that the hallucinated features are useful to estimate the camera SE3 pose. To support this claim, on the right-most side of Fig. 3.16 one can find the percentage

[1] ORB-SLAM acts as visual odometry in trajectories without loop closures.



(a) Dynamic images. (b) Dynamic images w/ mask. (c) Our images w/ ORB.



(d) Images of Telea (2004). (e) Images of Yu et al. (2018). (f) Static images.

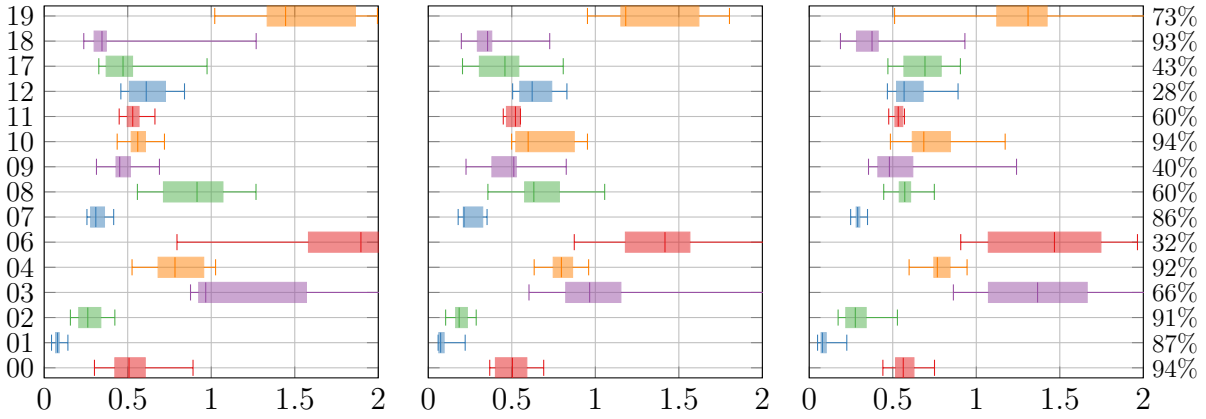
Figure 3.15: The vertical axis shows the different sequences from our CARLA dataset in which we have tested our model, and the horizontal axes show the ATE [m] obtained by ORB-SLAM and DynaSLAM. We have computed the boxplots' minimum, maximum and quartiles with the results from 10 repetitions of every test.

of key points extracted in the inpainted areas w.r.t. to the ground-truth key points. We show in Fig. 3.17 two examples of the visual influence of such loss on enhancing high frequencies inpainted areas.

Fig. 3.18 shows the DSO error for the same 20 CARLA sequences for the different input images –with dynamic content (Fig. 3.18a), without dynamic content (Fig. 3.18d), and the images obtained by our framework without and with the ORB-based loss (Figs. 3.18c and 3.18c)–. Even though direct systems are more robust to dynamic objects, utilizing our approach also yields a higher tracking accuracy. Despite the fact that our feature-based loss follows the ORB implementation, better results are also obtained with other visual odometry system that do not rely on ORB features.

Baselines for Inpainting

We have compared in Section 3.4 the quality of our results with respect to the inpainting metrics against four other methods. We want to compare now how our approach compares to them w.r.t. visual odometry metrics. Among these four other methods we have chosen two of them for this evaluation: that of Telea (2004), and that of Yu



(a) Our images w/o ORB. (b) Our images w/ ORB. (c) Our images w/ ORB w/ mask.

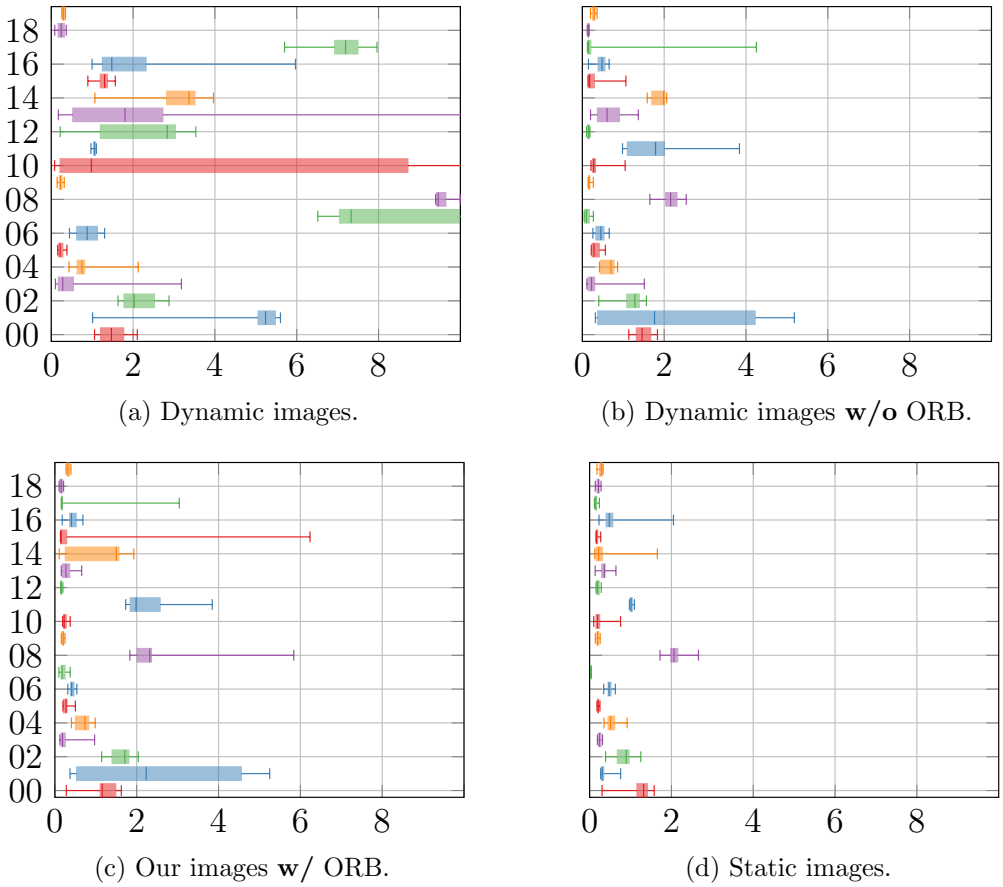
Figure 3.16: The vertical axis shows the different sequences from our CARLA dataset, and the horizontal axes show the ORB-SLAM ATE [m]. (a) and (b) show the ORB-SLAM absolute trajectory RMSE [m] for our images obtained with the model trained without and with the ORB loss term respectively. (c) shows the ATE computed by DynaSLAM for our images obtained with the model trained with the ORB loss term. On the right-most side of the figure one can find the percentage of key points extracted in the inpainted areas, w.r.t. to the ground-truth key points.

(a) Input. (b) Output **w/o** ORB. (c) Output **w/** ORB.

Figure 3.17: Visual comparison of the improvements achieved by utilizing the ORB-based loss (c) againt not using it (b). One can see that the reconstructed curbs in (c) are sharper and more straight than those in (b)

et al. (2018). The first choice is motivated by its performance on our inpainting test dataset: this method performs the best among the two non-learning based approaches. The second choice is however motivated by the fact that we have not been able to train the model by Iizuka et al. (2017) with our training data for a direct comparison. The evaluation and different results can be seen in Fig. 3.15.

The images inpainted by the method of Telea (2004) are usually very smooth and no features are extracted within the inpainted areas. The behaviour of ORB-SLAM



(a) Dynamic images.

(b) Dynamic images **w/o** ORB.

(c) Our images **w/** ORB.

(d) Static images.

Figure 3.18: The vertical axis shows the different sequences from our CARLA dataset, and the horizontal axes show the DSO ATE [m]. We have computed the boxplots' minimum, maximum and quartiles with the results from 10 repetitions of every test.

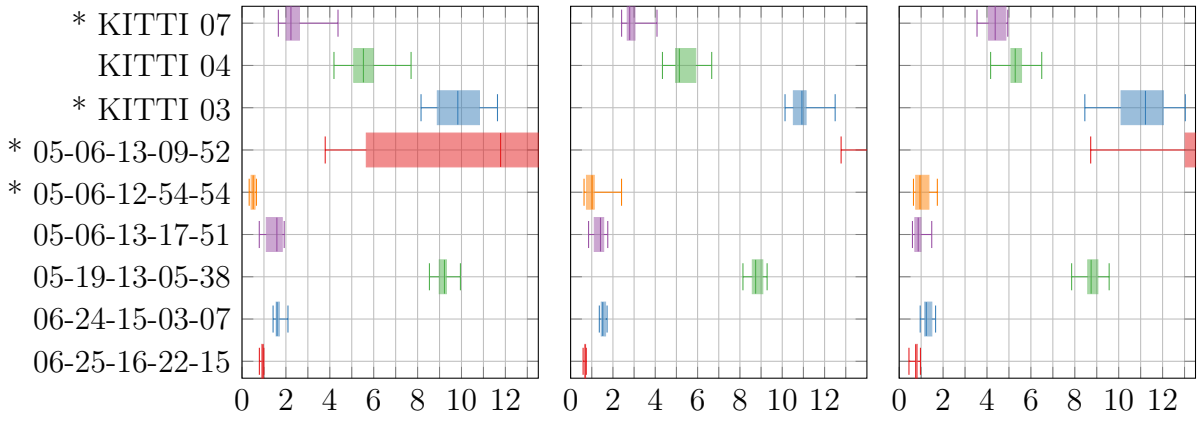
when using such sequences (Fig. 3.15d) is very similar to using DynaSLAM. However, the learning-based method byYu et al. (2018) tends to inpaint the images with low frequency patterns found in the image static content, generating many crispy artifacts (see examples in Fig. 3.12). A higher ATE is observed when using such images (Fig. 3.15e). Our method seems to be more suitable for the VO task: the inpainting is neither too smooth nor generates crispy artifacts.

Baselines for VO in Real World New Scenarios

For the evaluation of Empty Cities on real world environments w.r.t. visual odometry we have chosen the KITTI and the Oxford Robotcar datasets (Geiger et al. (2013), Maddern et al. (2017)). Dynamic objects in the KITTI odometry dataset do not represent a big inconvenience for camera pose estimation, as was shown in the previous chapter. Most of the vehicles that appear are not moving and lay in nearby scene areas, thus their features happen to be helpful to compute the sensor odometry. Also, the few moving pedestrians and cars along the sequences do not represent a big region within the images. The Oxford Robotcar dataset has though many sequences with representative moving objects (driving cars), having also sequences with only stationary objects (parked cars). Note that, the reported results are not for the whole sequence since the authors provide their VO solution as ground-truth, stating that it is accurate

over hundreds of metres. Hence, the sequences we use are between 100 and 300 m long.

To perform the KITTI experiment (Fig. 3.19) we have re-trained our network with 256×768 resolution images for a better adaptation. The CARLA camera intrinsics have also been modified to match the ones used in KITTI. In this case, we have shifted the previously used semantic segmentation model trained only on Cityscapes for the ERFNet model with encoder trained on ImageNet and decoder trained on Cityscapes train set, and have finetuned it with the KITTI semantic segmentation training dataset. The generator and discriminator have also been finetuned with such data as explained in Subsection 3.4.3.



(a) ORB-SLAM. (b) DynaSLAM. (c) Ours.

Figure 3.19: The vertical axis show the KITTI and Oxford Robotcar datasets sequences in which we have tested our model, and the horizontal axes show the boxplots of the absolute trajectory RMSE [m] with the results from 10 repetitions of every test. The asterisk in some sequence names means that most of the observed vehicles are parked.

Fig. 3.19 shows the evaluation of our method performance with the sequences from the Oxford Robotcar and the KITTI datasets. The asterisk at the beginning of some sequences names means that most of the observed vehicles are either not moving or parked. The other sequences though present many moving vehicles. In the former type of sequences (*) the highest accuracy should be observed in the case in which the raw images are used (Fig. 3.19a). Removing the features from such vehicles as in Fig. 3.19b leads to a lower accuracy since the most nearby features are no longer used. Inpainting the static scene behind these vehicles (Fig. 3.19c) would still remove nearby features but would create new static features a little bit further. That is, the visual odometry accuracy should be lower than in Fig. 3.19a but a little bit higher or similar to Fig. 3.19b. This is the case of our performance on the Oxford Robotcar sequences and the KITTI sequence 03. However, DynaSLAM achieves a result in the KITTI sequence 07 better the proposed Empty Cities. After the first half of the sequence, there are a few consecutive frames in which a truck covers almost 75 % of the image. The task of inpainting becomes especially difficult, thus worsening the estimation of the camera's trajectory. Regarding the performance of the second type of sequences, removing the features from moving vehicles and pedestrians should lead to a lower ATE (Fig. 3.19b compared to Fig. 3.19a). Empty Cities adds in these sequences an important number of features for pose estimation that usually leads to a slightly better trajectory estimation (Fig. 3.19c). Note that the results given in here for the KITTI sequences might not

match the ones reported by ORB-SLAM and DynaSLAM respectively because of the utilized images resolution (256×768).

3.5.2 Visual Place Recognition

Visual place recognition (VPR) is an important task for visual SLAM. Such algorithms are useful when revisiting places to perform loop closure and correct the accumulated drift along long trajectories. Bags of visual words (BoW) is the approach that is widely used to perform such task, as can be seen in ORB-SLAM (Mur-Artal & Tardós (2017)) and LDSO (Gao et al. (2018)). Lately, thanks to the boost of deep learning, learnt global image descriptors are also commonly used for VPR (Arandjelović et al. (2016)).

Baselines for Visual Place Recognition in our Synthetic Dataset

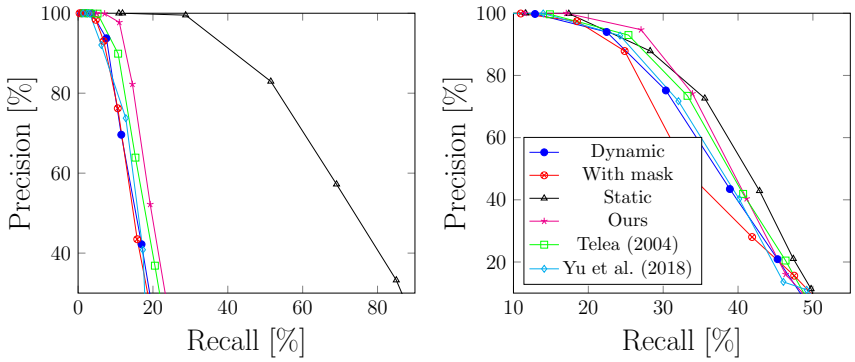
In this subsection we show a VPR experiment performed with the bag of words work by Mur-Artal & Tardós (2014) and Gálvez-López & Tardos (2012). It is ideal to test our model since is based on ORB features. We also show an experiment with one of the strongest learning-based baselines: NetVLAD (Arandjelović et al. (2016)), which is trained for the specific task of VPR. This comparison can provide a broader understanding of how and when end-to-end task-specific learning becomes more or less suitable than an explicit use of semantics-based visual description, which forms the primary pitch of this paper.

For this experiment we have generated two CARLA sequences with loop closures with and without dynamic objects. Two images are defined as the same place if they are less than 10 meters apart.

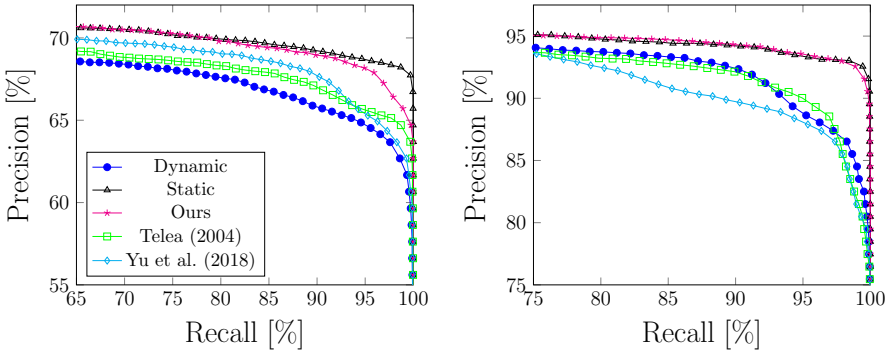
The precision-recall curves for the BoW experiments are depicted in Fig. 3.20a. We have extracted the visual words of every frame along the trajectories and have tried to match every two images as a function of the number of common visual words. For both trajectories the results obtained with the dynamic images with and without masks are similar. This is congruent with the idea that the database of visual words mostly contains static and long-term stable words. The first trajectory is a good example of how the VPR recall drops fast in presence of dynamic objects: a place is better represented with words from the whole static image. It leads to less false positives and less false negatives mostly. Even though our method slightly brings closer in the bag-of-words space images from the same place with different dynamic objects, we would have expected our results to be closer to those of the ground-truth static images. Our intuition behind these results is that the synthetic features, despite being useful for feature matching, do not fully fall on any visual words in the BoW space.

The precision-recall curves for the NetVLAD experiments are shown in Fig. 3.20b. We have extracted the learnt descriptors of every frame along the trajectories and tried to match every two images as a function of their descriptors' euclidean distance[2]. Despite NetVLAD's incredible performance on VPR face to illumination, view point and clutter changes, we show that their execution is seen slightly degraded by dynamic objects. Empty Cities brings closer together the descriptors of the same place, and pulls apart the descriptors of different places with the same dynamic objects, leading to higher precision and recall. That is, the hidden semantic representations of our model match the ones learnt by NetVLAD. We can conclude that our method brings

[2]with the Python-Tensorflow implementation by Cieslewski et al. (2018)



(a) Place recognition results for the BoW work by Mur-Artal & Tardós (2014) and Gálvez-López & Tardos (2012).



(b) Place recognition results for the learning-based method NetVLAD by Arandjelović et al. (2016).

Figure 3.20: (a) and (b) show the precision and recall curves for the VPR results with BoW and NetVLAD respectively. We report the results for the dynamic, the inpainted and ground-truth static sequences, as well as the results obtained when masking out the dynamic objects as in DynaSLAM (only in (a)). The precision and recall curves for the inpainting methods of Telea (2004) and of Yu et al. (2018) are also presented.

more relevant improvements in VPR if it is used in conjunction with a learning-based method. Finally, Fig. 3.21 shows a case in which NetVLAD fails at matching two dynamic frames of the same place, but manages to match them when dynamic objects are inpainted with our framework.

(a) Reference. (b) Query. (c) Empty Reference. (d) Empty Query.

Figure 3.21: (a) and (b) show the same location with different viewpoints and objects setups. NetVLAD (Arandjelović et al. (2016)) fails to match them, but it succeeds when our framework is previously employed –(c) and (d)–.

Baseline for Inpainting

We compare our approach to the inpainting methods by Telea (2004) and Yu et al. (2018)) w.r.t. the VPR metrics in Fig. 3.20. For the BoW curves the conclusion is similar to what we have seen in the previous experiment. Even if the extracted synthetic features were useful for matching, they seem to be of less help for place recognition with BoW. Few synthetic ORB features match any existing visual word in the BoW space. Our method though creates more useful visual words. As for the results with NetVLAD, the use of the geometric method by Telea (2004) brings little improvement. However, the learning-based method by Yu et al. (2018) decreases NetVLAD's performance on the second trajectory. NetVLAD can –up to some extent– ignore the dynamic classes clues, but cannot ignore the transformed regions. That is, the hidden semantic representations of these inpainted regions do not always match the static scene representation learnt by NetVLAD.

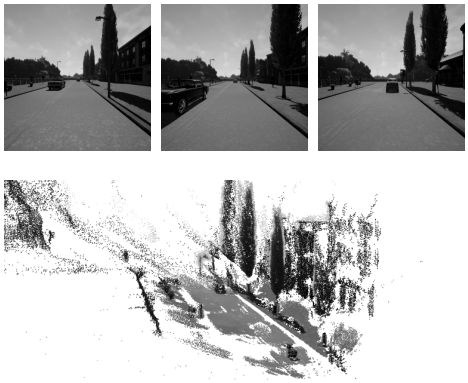
3.5.3 Mapping

Another important application of our framework is the creation of high-detail road maps. Our inpainting framework allows us to create long-term and reusable maps that do not either show dynamic objects or the empty spaces left by their occlusions. To this end we use COLMAP by Schönberger & Frahm (2016) and Schönberger, Zheng, Pollefeys & Frahm (2016), which is a general-purpose Structure-from-Motion (SfM) and Multi-View Stereo (MVS) software.

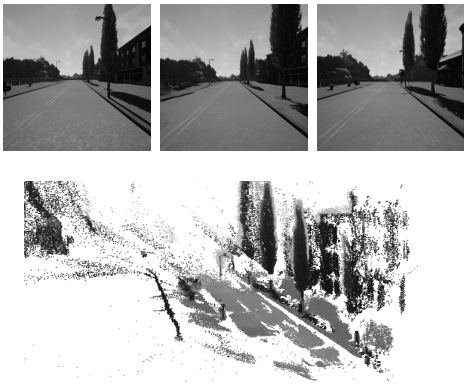
Baseline for Mapping in our Synthetic Dataset

Fig. 3.22 shows the dense map of a simulated city environment (CARLA) with the original dynamic sequence, with the images processed by our framework and with the ground-truth static images. The map seen in Fig. 3.22a might not be useful for future use since it shows dynamic objects that might not be there any more. Fig. 3.22e shows the map built with the ground-truth static images, and Fig. 3.22b shows the map computed with our generated images. The areas which have been consistently inpainted along the frames are mapped, even if they have never been seen. When inpainting fails or is not consistent along the sequence, the mapping photometric and geometric epipolar constraints are not met and such areas cannot be reconstructed. This idea makes our framework suitable to build stable maps.

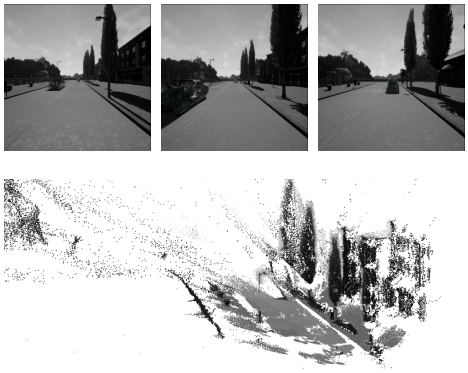
To give a quantitative experiment on the validity of our maps, we choose the standard Iterative Closest Point (ICP) algorithm (Besl & McKay (1992)) based on the Euclidean fitness score. That is, having the point cloud built with the ground-truth static images as a fix reference, for each of its points the algorithm searches for the closest point in the target point cloud and calculates the distance based on the result of the algorithm search. To have a baseline for our experiment, we compute the point cloud with the dynamic images and with the CARLA segmentation. That is, the pixels belonging to the dynamic objects have not been used in the multi-view-stereo pipeline. The results can be seen in Table 3.4. Since the map has no scale, the ICP Euclidean fitness score is also up-to-scale. Even though the similarity score is improved when masking out dynamic objects, the map built with our images has triangulated more 3D points from the inpainted regions, and such points have a low error.



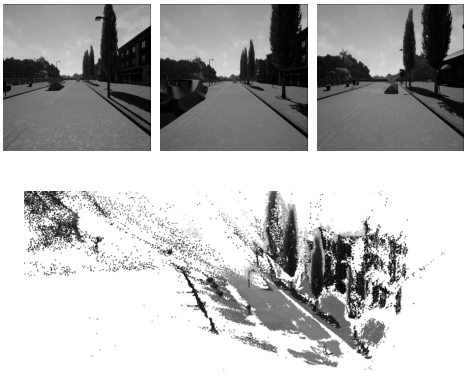
(a) Dynamic map.



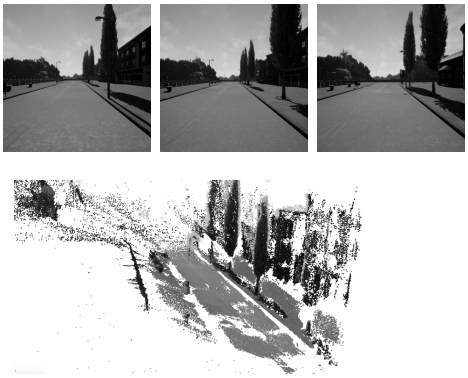
(b) Our map.



(c) Map of Yu et al. (2018).



(d) Map of Telea (2004).



(e) Ground-truth static map.

Figure 3.22: Dense maps of a CARLA city environment built with the COLMAP Multi-View-Stereo software (Schönberger, Zheng, Pollefeys & Frahm (2016)). The upper row shows some of the sequence images used to build such maps. (a) shows the case in which the original dynamic images have been used. (b) shows the resulting map with the images previously processed by our framework, and (e) uses the ground-truth static images to create the resulting map. (c) and (d) show the resulting maps with the images previously processed by the frameworks from Yu et al. (2018) and Telea (2004) respectively. All maps are computed with the ground-truth camera poses.

	Dynamic	Dynamic w/ masks	Ours	Telea (2004)	Yu et al. (2018)
ICP score	63.21	49.02	**33.34**	82.23	70.89

Table 3.4: Quantitative mapping results for Fig. 3.22. We report the Euclidean fitness score given by the ICP algorithm. This score has been computed between the different point clouds w.r.t. the point cloud built with the ground-truth static images. Note that, since the maps do not have scale, the reported scores are up-to-scale.

Baseline for Inpainting

We also compare our approach with the two other state-of-the-art inpainting methods –Telea (2004) and Yu et al. (2018)– w.r.t. the map quality. The qualitative results are depicted in Fig. 3.22, and the ICP scores are shown in Table 3.4. It can be seen that with both other inpainting methods the shadows of the dynamic objects are reconstructed, as well as their prolongation into the inpainted image regions. This leads to an ICP score that is higher than that of the map built with dynamic objects.

Baseline for Mapping in Real World Environments

Fig. 3.23 shows an example of the computed dense maps for both types of inputs with the sequence 04 from the KITTI dataset. These maps have been computed with the camera poses given by ORB-SLAM using respectively the dynamic and inpainted images. Our framework, when being able to inpaint a coherent context along the sequence, allows us to densely reconstruct unseen areas. When the inpainting task is not coherent along the sequence, the epipolar constraints are not met and therefore such areas cannot be reconstructed. Note that this map is shown in RGB only for visualization purposes[3]. We do recommend to use the gray-scale images for both localization and mapping purposes since a lower reconstruction error is usually achieved.

3.5.4 Timing Analysis

Reporting our framework efficiency is crucial to judge its suitability for autonomous driving and robotic tasks in general. The end-to-end pipeline runs at 100 fps on a nVidia Titan Xp 12GB with images of a 512×512 resolution. Out of the 10 ms it takes to process one frame, 8 ms are invested into obtaining its semantic segmentation, and 2 ms are used for the inpainting task. Other than to deal with dynamic objects, the semantic segmentation may be needed for many other tasks involved in autonomous navigation. In such cases, our framework would only add 2 extra ms per frame. Based on our analysis, we consider that the inpainting task is not at all a bottleneck.

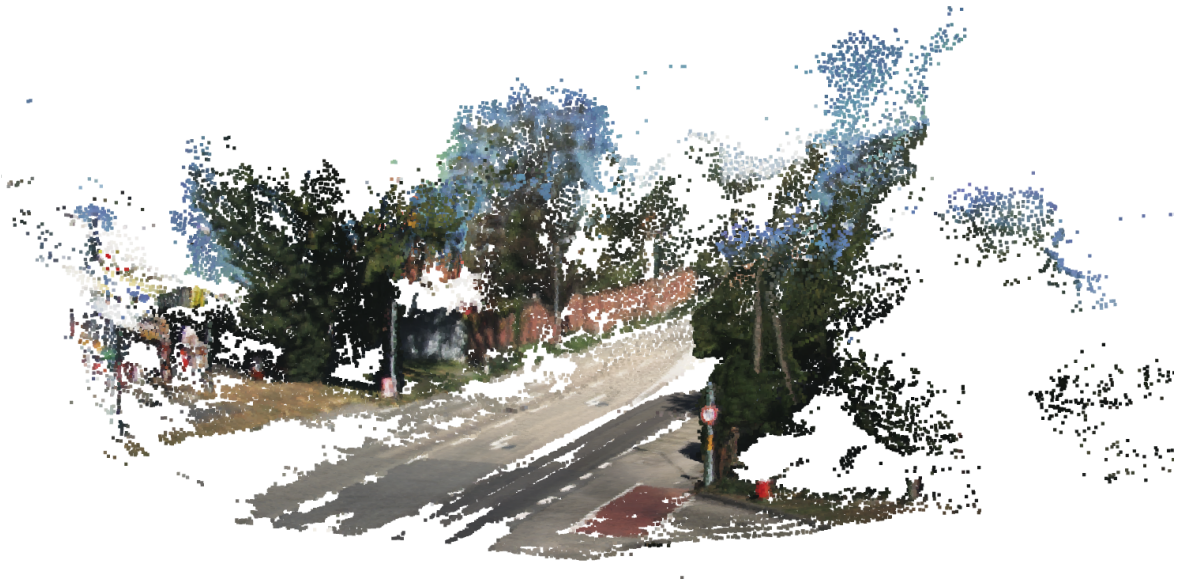
3.6 Failure Modes and Future Work

The aim of this section is to provide the reader with an understanding of the benefits and limitations of our proposal, and how to integrate it into a SLAM or MVS pipeline.

[3]Since our framework offers enough flexibility, we have re-trained our model with RGB images just as explained in Section 3.3. The only difference is that since the features have to be extracted in gray-scale images, we add a fixed convolutional layer to convert the RGB output images to gray scale.

(a) Input of our system: urban images with dynamic content.

(b) Output of our system: dynamic objects have been removed.



(c) Static map built with the images pre-processed by our framework.

Figure 3.23: The dynamic images are firstly converted one by one into static with an end-to-end deep learning model. Such images allow us to compute an accurate camera trajectory estimation that is not damaged by the dynamic objects' motion, as well as to build dense static maps that are useful for long-term applications.

Having a scene static representation leads to better visual odometry results than just excluding the features belonging to moving objects. The presented inpainting approach has though some weaknesses: the bigger the image dynamic region is, the lower inpainting quality the resulting image has. Empty Cities would be suitable in setups in which approximately less than 15 % of the camera field of view is covered by dynamic objects. In such setups, the reconstructed L1 error is acceptable and usually lies between 1 and 10 %. The L1 error goes above 10 % when more than 15 % of the image pixels are covered[4]. Work remains to be done to tackle extreme situations: developing a system that processes the sequence as a whole, rather than binning it into independent frames, would result in a more consistent image inpainting along time.

[4]As a practical example, the Oxford Robotcar sequence 2014-05-06-12-54-54 has 56 % of images with less than 5 % of covered pixels, 30 % with a percentage of dynamic pixels between 5 and 10 %, 12 % between 10 and 15 % and 2 % between 15 and 20 %. This sequence is not Manhattan at 11 am, but shows cars parked at both sides of the road and cars driving nearby.

Our system processes the image streams outside of the application pipeline and hence can be used naturally as a front end to many existing systems. We hereby want to discuss other application-dependent possibilities to boost its performance.

Visual Odometry: Removing the features that belong to static cars certainly damages VO. However, a car can from one frame to another change from static to dynamic. Had we a movement detector, we would use the features belonging to static cars and the inpainted ones behind the moving cars.

Place Recognition: Using the features belonging to parked cars damages the performance of place recognition algorithms. Two frames of the same place with a different setup of parked cars can be incorrectly tagged as a different place. Also, two frames of different places with the same parked cars setup can be wrongly matched as the same place. Only the features belonging to objects that remain stable in the long term (buildings, sidewalks, *etc.*) would benefit PR.

Mapping: A map containing information about parked cars would be useless for future reuse. Only the information belonging to objects that remain stable in the long term, as well as the most likely static representation of the static scene behind dynamic objects should be included in the map.

Ideally, one would use a movement detector to identify the status of the different observed instances. The features belonging to static instances would be used for visual odometry, and the corresponding inpainted features would be used for place recognition and mapping. This approach would bring the highest accuracy results but would entail a series of modifications to the existing pipeline. Our method would currently pose problems in the case in which one wanted to inpaint the static scene behind a car that is currently moving, and this static scene contained parked cars. Our model would fail in reconstructing the unseen parts of the parked cars. It would be interesting for future work to include such scenarios in our training data. Our suggestion is, for now, inpainting all instances, regardless of their current dynamic status.

3.7 Discussion

We have presented an end-to-end deep learning framework that translates images that contain dynamic objects within a city environment, such as vehicles or pedestrians, into a realistic image with only static content. These images are suitable for visual odometry, place recognition and mapping tasks thanks to a new loss based on steganalysis techniques and ORB features maps, descriptors and orientation. We motivate this extra complexity by showing quantitatively that the systems ORB-SLAM and DSO obtain a higher accuracy when utilizing the images synthesized with this loss. Also, mapping systems can benefit from this approach since not only they would not map dynamic objects but also would they map the plausible static scene behind them. Finally, an architectural nicety is that our system processes the image streams outside of the localization pipeline, either offline or online, and hence can be used naturally as a front end to many existing systems.

3.8 Related Publications

- Berta Bescos, Cesar Cadena and José Neira. "Empty Cities: a Dynamic-Object-Invariant Space for Visual SLAM". This work is currently conditionally accepted

for publication at IEEE Transactions on Robotics.

- Berta Bescos, Cesar Cadena and José Neira. "Dynamic-to-Static Image Translation for Visual SLAM". Oral and Poster Presentation within the Workshop Scene and Situation Understanding for Autonomous Driving at *IEEE Robotics Science and Systems*, June 2019. **Winner Toyota Research Institute (TRI) Best Contribution Award**.

- Berta Bescos, José Neira, Roland Siegwart , Cesar Cadena. "Empty Cities: Image Inpainting for a Dynamic-Object-Invariant Space". Oral presentation at *IEEE International Conference on Robotics and Automation*, May 2019.

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes". *IEEE Robotics and Automation Letters* and oral presentation at *IEEE International Conference on Intelligent Robots and Systems*, October 2018.

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "Detecting, Tracking and Eliminating Dynamic Objects in 3D Mapping using Deep Learning and Inpainting". Oral and Poster Presentation within the Workshop Representing a Complex World: Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding at *IEEE International Conference on Robotics and Automation*, May 2018.

- Berta Bescos, Jose M. Facil , Javier Civera , José Neira. "Robust and Accurate 3D Mapping by combining Geometry and Machine Learning to deal with Dynamic Objects". Poster Presentation within the Workshop Learning for Localization and Mapping at *IEEE International Conference on Intelligent Robots and Systems*, September 2017.

Chapter 4

Multi-Object Tracking

In recent years, a growing part of the robotics community has addressed the issue of visual SLAM in dynamic environments by incorporating the dynamics of moving objects into the problem: Li et al. (2018), Henein et al. (2018), Yang & Scherer (2019), Huang et al. (2019, 2020). The objective of this group is twofold: they not only focus on achieving an accurate ego-motion estimation but also provide information about the poses of other dynamic agents within the scene. Understanding surrounding dynamic objects is of crucial importance for the frontier requirements of emerging applications within AR/VR or autonomous systems navigation.

Although it is tolerable to discard minor movements in an almost static environment, most scenarios including autonomous driving, multi-robot collaboration and augmented/virtual reality require explicit motion information of the surroundings to help with decision making and scene understanding. In virtual reality, dynamic objects need to be explicitly tracked to enable interactions of virtual objects with moving instances in the real world. In autonomous driving scenarios, a car should not only localize itself but also reliably sense other moving cars and passersby to avoid collisions.

The vast majority of the literature that specifically addresses this issue do so by detecting moving objects and tracking them separately from the SLAM formulation using traditional multi-target tracking approaches (Wang et al. (2003), Miller & Campbell (2007), Rogers et al. (2010), Kundu et al. (2011), Bârsan et al. (2018), Rosinol et al. (2020)). Their accuracy is highly dependent on the camera pose estimation, which is more susceptible to failure in complex dynamic environments where the presence of reliable static structure can be limited. In the latest years, the robotics community has made its first steps towards addressing dynamic objects tracking jointly with visual SLAM, adding an extra layer of complexity to the problem (Henein et al. (2018), Huang et al. (2019, 2020), Zhang et al. (2020). These systems are often tailored for special use cases and multiple priors are exploited to constraint the solution spaces: road planar structure and planar object movement in autonomous driving scenarios (Li et al. (2018), Yang & Scherer (2019)), or even the use of known object models (Hosseinzadeh et al. (2019)).

In this chapter we introduce DynaSLAM II, an open-source stereo/RGB-D SLAM system for dynamic environments, which simultaneously estimates the poses of the camera, the map and the trajectories of the moving objects in the scene. We propose a bundle adjustment solution that tightly optimizes the scene structure, the camera poses and the objects trajectories in a local-temporal window. The objects' bounding boxes are also optimized in a decoupled formulation that allows to estimate the dimensions

and the 6 DoF poses of the objects without being geared to any particular use case. Examples of our output can be seen in Fig. 4.1.

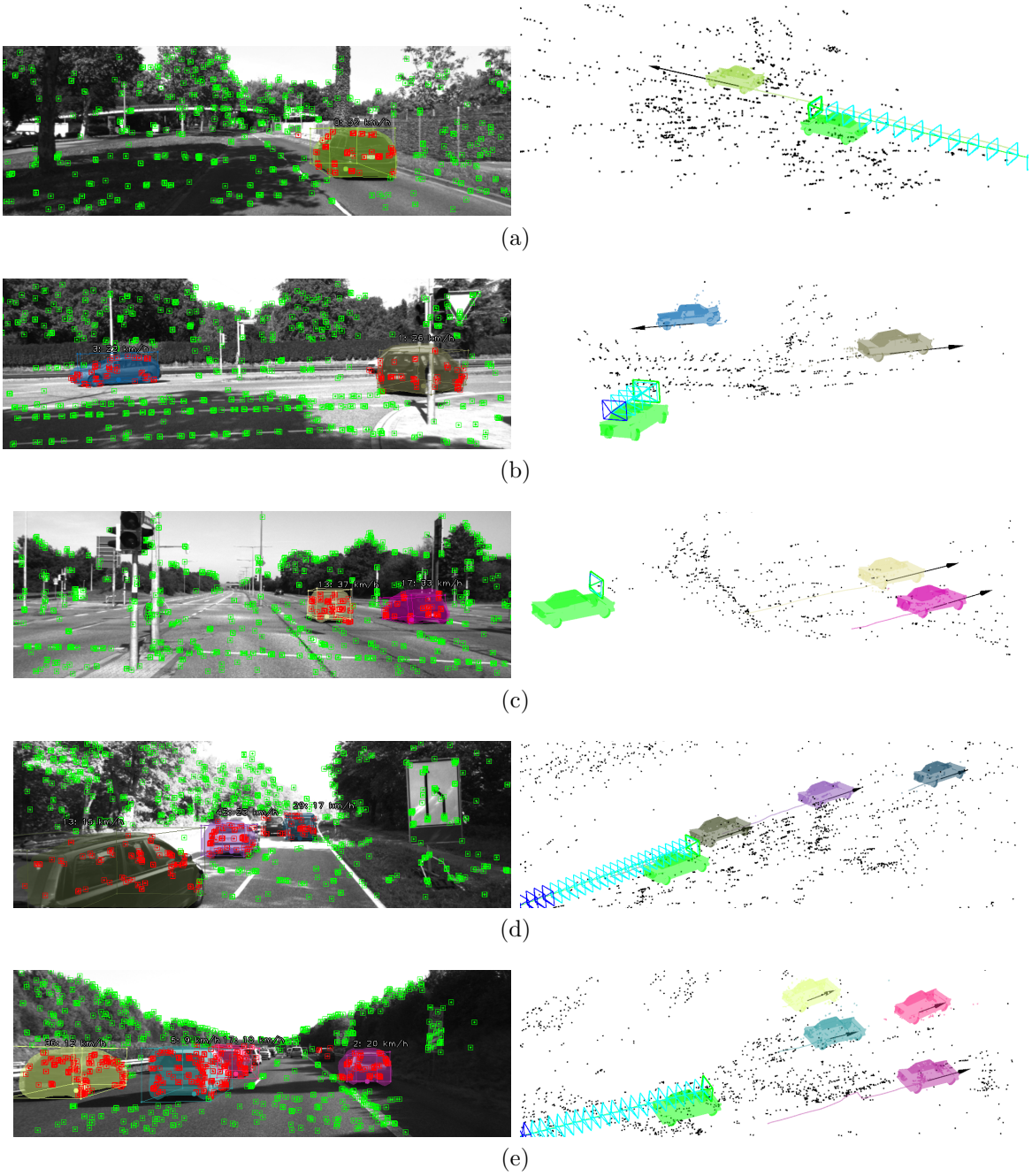


Figure 4.1: Qualitative results with the KITTI tracking dataset. One can see in the left images the current frame with the tracked objects highlighted with colors. The 2D projections of the different 3D bounding boxes are drawn in the frames together with the objects' track ids and their current velocity (better seen in digital format). In the right images one can see the 3D scene estimation with the ego-car representation in light green. The other dynamic objects are represented with their current pose, their trajectory and their current velocity vector. The black points depict the static scene.

4.1 Related Work

The traditional manner of addressing 3D multi-object tracking implies to detect the moving objects and track them separately from the SLAM formulation. Examples of these works in the bibliography are the following: Wang et al. (2003), Miller & Campbell (2007), Wangsiripitak & Murray (2009), Rogers et al. (2010), Kundu et al. (2011), Bârsan et al. (2018), Rosinol et al. (2020). The main drawback of these approaches is that their accuracy is highly correlated with the estimation of the camera position. Therefore, if the estimation of the camera pose fails, which is quite likely in complex dynamic environments where the presence of a reliable static structure is questionable, multi-object tracking also fails directly. Among them, Wang et al. (2003) derive the Bayesian formula of the SLAM with DATMO (Detection And Tracking of Moving Objects) problem, and provide a solid basis for understanding and solving this problem. Wangsiripitak & Murray (2009) more recently proposed the parallel implementation of SLAM with a 3D object tracker: the SLAM solution provides the object tracker with information to register objects to the map, and the object tracker allows the marking of features on objects. Rogers et al. (2010) apply an Expectation Maximization technique to a graph based SLAM approach and allow the landmarks points to be dynamic. In the work of Kundu et al. (2011) various modules of feature tracking, motion segmentation, visual SLAM and moving-object tracking are continuously interleaved and run online. More recently, Bârsan et al. (2018) present a stereo-based dense mapping algorithm for dynamic urban environments that simultaneously reconstructs the static background and the moving objects. There is a new work by Rosinol et al. (2020) that reconciles visual-inertial SLAM and dense mesh tracking, focusing mostly on humans. They show impressive object tracking results but uniquely with simulation data.

The idea of simultaneously estimating camera ego motion and multiple moving rigid objects motion originated from the SLAMMOT work (Wang et al. (2007)). They established a mathematical framework to integrate a filtering-based SLAM and moving object tracking and demonstrated that it satisfied both navigation and safety requirements in autonomous driving. Later on, works that use a RGB-D camera followed up this idea to densely reconstruct the static indoors scenes along with the moving objects utilizing pixel-wise instance segmentation, showing impressive results (Rünz & Agapito (2017), Runz et al. (2018), Xu et al. (2019)). Since it is of crucial importance for dense approaches to obtain a highly accurate segmentation mask, the works Mask-Fusion and MID-Fusion, by Runz et al. (2018) and Xu et al. (2019) respectively, refine it by making the assumption that –human-made– objects are largely convex.

Among the feature-based approaches, as ours, few aim to utilize information from static and dynamic objects into a single framework to improve the accuracy of the estimation. Henein et al. (2018) were among the first ones to combine the two problems of tracking dynamic objects and the camera ego motion. However, they only reported experiments on synthetic data showing limited real dataset results. Li et al. (2018) use a CNN trained in an end-to-end manner to estimate the 3D pose and dimensions of cars, which is further refined together with the camera poses. The use of data-driven approaches often provides excellent accuracy in 6 DoF object pose estimation, but also a loss of generality. Therefore, they can only track cars. Huge amounts of data would be required to track generic objects with their approach. The authors of CubeSLAM (Yang & Scherer (2019)) showed impressive results with only a monocular camera and new measurements to integrate multi-object tracking within the SLAM

formulation. They make use of KLT to track the dynamic features among frames, of a 3D bounding box proposal generation based on 2D bounding boxes and vanishing points. They assume that objects have a constant velocity motion within a hard-coded duration time interval and exploit object priors like car sizes, road planar structure and object planar non-holonomic wheel movement model. Furthermore, they only track objects whose 3D bounding box is observable, *i.e.*, only once two or more faces of the *cuboid-shape* object are seen. ClusterSLAM (Huang et al. (2019)) proposes a stereo SLAM back end with no scene prior to discover individual rigid bodies and compute their motions in dynamic environments. Since it acts as a back end instead of a full system, its performance relies heavily on the landmark tracking and association quality. The same authors recently developed the full system ClusterVO (Huang et al. (2020)), which models the object points with a probability of object belonging to deal with segmentation inaccuracies. Given that they assume no priors, they obtain good ego motion results in both indoor and outdoor scenarios, but the 3D bounding boxes they estimate are not completely accurate. There is a recent work called VDO-SLAM utilizing dense optical flow estimation to maximise the number of tracked points on moving objects (Zhang et al. (2020)). They implement –similarly to their previous work (Henein et al. (2020))– a bundle adjustment with cameras, objects and object points that gives good results but that is computationally complex to run in real time.

In light of these advances, it is apparent that the feature-based SLAM community is searching for the best optimization formulation to combine cameras, objects and structure points. In our proposal, we use a tightly-coupled bundle adjustment formulation with new measurements between cameras, objects and points giving special attention to its computationally complexity and number of parameters involved without introducing hard-coded priors. For this, we integrate instance semantic priors together with sparse image features. This formulation allows the estimation of both the camera, the map structure and the dynamic objects to be mutually beneficial at a low computational cost. On the other hand, part of the current literature focuses on the estimation of the point cloud structure of dynamic objects and of the trajectory of a random object reference (Huang et al. (2019), Zhang et al. (2020)), whereas another part of the literature seeks to find a common reference for objects of the same class as well as a more informative occupancy volume (Li et al. (2018), Yang & Scherer (2019), Huang et al. (2020)). We intend to carry out these two tasks independently in order to leverage on the benefits of both without their disadvantages.

4.2 DynaSLAM II

DynaSLAM II builds on top of the popular ORB-SLAM2 (Mur-Artal & Tardós (2017)) and takes synchronized and calibrated stereo / RGB-D images as input, and outputs the camera and the dynamic objects poses for each frame, as well as a spatial / temporal map containing the dynamic objects. For each incoming frame, pixel-wise semantic segmentation is computed and ORB features (Rublee et al. (2011)) are extracted and matched across stereo image pairs. We first associate the static and dynamic features with the ones from the previous frame and the map assuming a constant velocity motion for both the camera and the observed objects. Object instances are then matched based on the dynamic feature correspondences. The static matches are used to estimate the initial camera pose, and the dynamic ones yield the object's SE(3) transform. Finally,

the camera and objects trajectories, as well as the objects bounding boxes and 3D points are optimized over a sliding window with marginalization and a soft smooth motion prior. The different contributions and building blocks of DynaSLAM II, as well as the used notation, are explained and discussed in detail in the following subsections.

4.2.1 Notation

In this chapter, we will use the following notation: a stereo/RGB-D camera at time i, $\mathtt{C}_i$, has a pose $\mathbf{T}^i_{\mathtt{CW}} \in \mathrm{SE}(3)$ in the world coordinates $\mathtt{W}$ (see Fig. 4.2). The camera observes static map points $\mathbf{X}^l_{\mathtt{W}} \in \mathbb{R}^3 \mid \forall l \in \mathcal{MP}$ and dynamic objects points $\mathbf{X}^{j,k}_{\mathtt{O}} \in \mathbb{R}^3 \mid \forall j \in \mathcal{OP}, \forall k \in \mathcal{O}$. The set of object points $\mathcal{OP}$ belonging to the same instance let us define the dynamic object k, $\mathtt{O}^k_i$, which has a linear and angular velocity in object coordinates $\mathbf{v}^k_i, \mathbf{w}^k_i \in \mathbb{R}^3$ respectively, and a world coordinates pose $\mathbf{T}^{k,i}_{\mathtt{OW}} \in \mathrm{SE}(3)$.

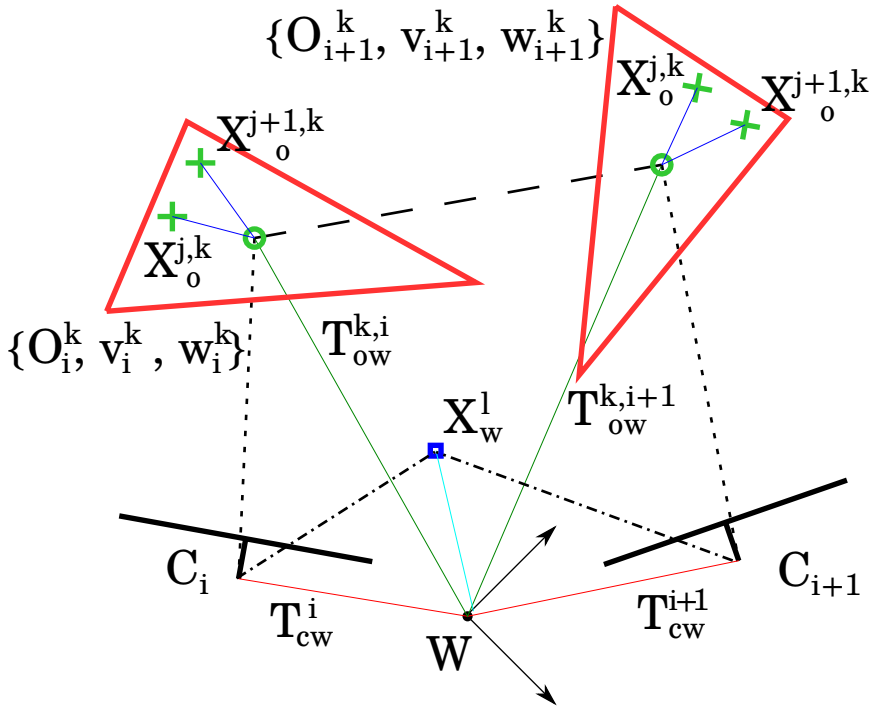


Figure 4.2: Depiction of the used notation to model dynamic objects and their points. The dotted line (···) depicts that the camera $\mathtt{C}_i$ is observing the object $\mathtt{O}^k_i$, and the dashed line (– – –) shows that the two object instances $\mathtt{O}^k_i$ and $\mathtt{O}^k_{i+1}$ are the same moving body at different times. The dash-dotted line (–·–·–) are static point observations.

4.2.2 Object Data Association

For each incoming frame, pixel-wise semantic segmentation is extracted and ORB features (Rublee et al. (2011)) are extracted and matched across stereo image pairs. If an instance belongs to a dynamic class –vehicles, pedestrians and animals– and contains a high number of new nearby key points, an object is created. The key points are then assigned to the instance and the corresponding object. We first associate the static features with the ones from the previous frame and the map to initially estimate the camera pose. Next, dynamic features are associated with the dynamic object points from the local map in two different ways: (a) if the velocity of the map objects

is known, the matches are searched by reprojection assuming a inter-frame constant velocity motion, (b) if the objects' velocity is not initialized or not enough matches are found following (a), we constraint the brute force matching to those features that belong to the most overlapping instance within consecutive frames. A higher level association between instances and objects is also required. If the majority of the key points assigned to a new object are matched with points belonging to one map object, the two objects are attributed the same track id. Also, to render this high level association more robust, a parallel instance-to-instance matching is performed based on the intersection over union (IoU) of the instances 2D bounding boxes.

The SE(3) pose of the first object of a track is initialized with the center of mass of the 3D points and with the identity rotation. To predict the poses of further objects from a track, we use a constant velocity motion model –null velocity if it is not initialized– and refine the object pose estimate by minimizing the reprojection error.

The common formulation of the reprojection error seen in multi-view geometry for a camera i with pose $\mathbf{T}^i_{\mathtt{CW}} \in \mathrm{SE}(3)$ and a point l with homogeneous position $\mathbf{X}^l_{\mathtt{W}} \in \mathbb{R}^4$ in the world reference $\mathtt{W}$ with a stereo key point correspondence $\mathbf{x}^l_i \in \mathbb{R}^3$ is

$$\mathbf{e}^{i,l}_{\mathbf{repr}} = \mathbf{x}^l_i - \pi_s(\mathbf{T}^i_{\mathtt{CW}}\mathbf{X}^l_{\mathtt{W}}), \tag{4.1}$$

where π_s is the reprojection function for a rectified stereo camera with focal length (f_x, f_y), optical center (c_x, c_y) and horizontal baseline b (Eqn. 4.2). Given these camera intrinsics, π_s projects a 3D homogeneous point in the camera coordinates $\mathbf{X}_{\mathtt{C}} = [X, Y, Z, 1]^T$ into the camera frame pixel $\mathbf{x} = [u_L, v_L, u_R]^T$. u_L and v_L are respectively the pixel column and row on the left image, and u_R is the pixel column on the right image. Note that, even though these equations have been already presented in Chapter 1 we hereby rewrite them for homogeneous coordinates points.

$$\mathbf{x} = \pi_s(\mathbf{X}_{\mathtt{C}}) = \begin{bmatrix} f_x\frac{X}{Z} + c_x \\ f_y\frac{Y}{Z} + c_y \\ f_x\frac{X-b}{Z} + c_x \end{bmatrix} \tag{4.2}$$

Unlike this formulation, which is valid for static representations, we propose to restate the reprojection error as

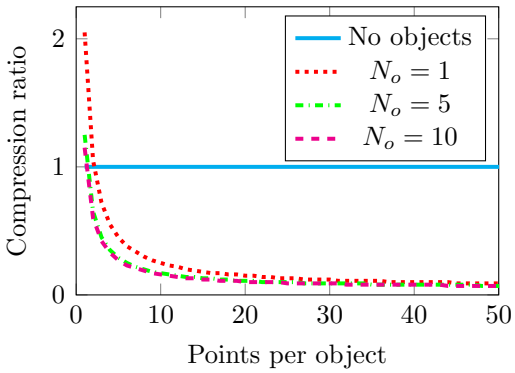
$$\mathbf{e}^{i,j,k}_{\mathbf{repr}} = \mathbf{x}^j_i - \pi_s(\mathbf{T}^i_{\mathtt{CW}}\mathbf{T}^{k,i}_{\mathtt{WO}}\mathbf{X}^{j,k}_{\mathtt{O}}), \tag{4.3}$$

where $\mathbf{T}^{k,i}_{\mathtt{WO}} \in \mathrm{SE}(3)$ is the inverse pose of the object k in the world coordinates when the camera i is observing it, and $\mathbf{X}^{j,k}_{\mathtt{O}} \in \mathbb{R}^4$ represents the 3D homogeneous coordinates of the point j in its object reference k with observation in the camera i $\mathbf{x}^j_i \in \mathbb{R}^3$. This formulation enables us to optimize either jointly or separately the poses of the cameras and of the different moving objects, as well as the positions of their 3D points.

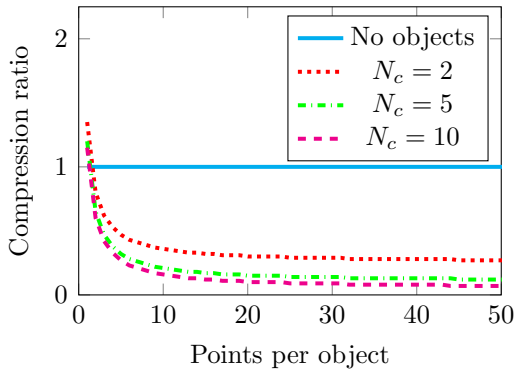
4.2.3 Object-Centric Representation

Given the extra complexity and mainly the extra number of parameters that the task of tracking moving objects implies on top of the usual SLAM ones, it is of high importance to keep this number as reduced as possible to maintain real-time performance. Modeling dynamic points as repeated 3D points by forming independent point clouds as in usual dynamic SLAM implementations results in a prohibitive amount of parameters. Given a set of N_c cameras, N_o dynamic objects with N_p 3D points each observed

in all cameras, the number of parameters needed to track dynamic objects turns to be $N = 6N_c + N_c \times N_o \times 3N_p$ as opposed to $N = 6N_c + 3N_p$ in conventional static SLAM representations. This number of parameters becomes prohibitive for long –and not so long– operations and deployment. If the concept of objects is introduced, 3D object points become unique and can be referred to their associated dynamic object. Therefore it is the pose of the object that is modelled along time and the number of required parameters shifts to $N' = 6N_c + N_c \times 6N_o + N_o \times 3N_p$. It can be seen in Fig. 4.3 the parameters compression ratio defined as $\frac{N'}{N}$ for a fixed number of cameras in Fig. 4.3a and for a fixed number of objects in Fig. 4.3b. If the number of observations and the number of dynamic points per object are sufficiently large, this modelling of dynamic objects and points brings a great saving in the number of parameters required.



(a) The number of cameras N_c is fixed to 10.



(b) The number of objects N_o is fixed to 10.

Figure 4.3: Motivation to delete the association between the 3D points that belong to dynamic objects and their observing key frame. Instead, objects are associated to their observing cameras and dynamic points are associated only to their object. For rigid objects, this last transformation remains fixed. These graphs show the relationship between the number of parameters that is needed when objects are used and when points belonging to objects are tracked independently (No objects). If the number of observations (N_c) and the number of points per object are sufficiently large, there is a great saving in the number of parameters that is required.

4.2.4 Bundle Adjustment with Objects

Bundle Adjustment (BA) is known to provide accurate estimates of camera poses and sparse geometrical reconstruction, given a strong network of matches and good initial guesses. We hypothesize that BA might bring similar benefits if object poses are also jointly optimized. The static map point 3D locations $\mathbf{X}^l_{\mathtt{W}}$ and camera poses in the world reference $\mathbf{T}^i_{\mathtt{CW}}$ are optimized by minimizing the reprojection error with respect to the matched key points $\mathbf{x}^l_i$ (Eqn. 4.1). In a similar way, for dynamic representations, the 3D object points $\mathbf{X}^{j,k}_{\mathtt{O}}$, the camera poses $\mathbf{T}^i_{\mathtt{CW}}$ and the object poses $\mathbf{T}^{k,i}_{\mathtt{WO}}$ can be refined by minimizing the reprojection error formulation presented in Eqn. 4.3.

In our implementation, a key frame can be inserted in the map for two different reasons: (a) if the camera tracking is weak, (b) if the tracking of any scene object is weak. The reasons for the former are the same ones than in ORB-SLAM2. The latter though happens if an object with a relatively large amount of features has few points

tracked in the current frame. In this case, a key frame is inserted and creates a new object and new object points. If the camera tracking is not weak, this key frame would not introduce new static map points, and if the rest of dynamic objects have a stable tracking, new objects for these tracks would not be created. As for the optimization, if a key frame is inserted only because the camera tracking is bad, the local BA optimizes the currently processed key frame, all the key frames connected to it in the covisibility graph, and all the map points seen by those key frames, following the implementation of ORB-SLAM2. Regarding dynamic data, if a key frame is inserted only because the tracking of an object is weak, the local BA optimizes the pose and velocity of this object along a temporal tail of 2 seconds together with its object points. Finally, if a key frame is inserted because the tracking of both the camera and objects are weak, camera poses, map structure, object poses, velocities and points are jointly optimized.

In order to force a smooth object trajectory estimation, we assume that an object has a constant velocity in consecutive observations. Denoted the linear and angular velocity of an object k at observation i as $\mathbf{v}_i^k \in \mathbb{R}^3$ and $\mathbf{w}_i^k \in \mathbb{R}^3$ respectively, we then define the following error term:

$$\mathbf{e}_{\mathbf{vcte}}^{i,k} = \begin{bmatrix} \mathbf{v}_{i+1}^k - \mathbf{v}_i^k \\ \mathbf{w}_{i+1}^k - \mathbf{w}_i^k \end{bmatrix} \tag{4.4}$$

An additional error term is needed to couple the objects velocities with the objects poses and their belonging 3D points. This error term can be seen in Eqn. 4.5, where $\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1}$ is the pose transformation that the object k suffers in the time interval $\Delta t_{i,i+1}$ between consecutive observations i and $i+1$.

$$\mathbf{e}_{\mathbf{vcte},\mathbf{XYZ}}^{i,j,k} = (\mathbf{T}_{\mathtt{WO}}^{k,i+1} - \mathbf{T}_{\mathtt{WO}}^{k,i}\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1})\mathbf{X}_{\mathtt{O}}^{j,k} \tag{4.5}$$

The term $\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1}$ is defined from the linear and angular velocity of the object k in the instant i ($\mathbf{v}_i^k$ and $\mathbf{w}_i^k$) as in Eqn. 4.6, where $\mathrm{Exp} : \mathbb{R}^3 \rightarrow \mathrm{SO}(3)$ is the exponential map for SO(3).

$$\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1} = \begin{pmatrix} \mathrm{Exp}(\mathbf{w}_i^k \Delta t_{i,i+1}) & \mathbf{v}_i^k \Delta t_{i,i+1}) \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix} \tag{4.6}$$

The derivative of the term $\Delta\mathbf{T}_{\mathtt{O}_k}^{i,i+1}(\mathbf{w}_i^k, \mathbf{v}_i^k)$ with respect to the angular velocity is approximated as follows. Let us apply an additive perturbation $\delta\mathbf{w}_i^k$ to the object angular velocity (Eqn. 4.7). We write only the rotational part of the pose transformation for simplicity.

$$\Delta\mathbf{R}_{\mathtt{O}_k}^{i,i+1}(\mathbf{w}_i^k + \delta\mathbf{w}_i^k) = \mathrm{Exp}((\mathbf{w}_i^k + \delta\mathbf{w}_i^k)\Delta t) \tag{4.7}$$

We use the first order approximation (Eqn. 4.8), where the term $\mathbf{J}_r$ is the right Jacobian of SO(3) (Forster et al. (2016)) and relates additive increments in the tangent space to multiplicative increments applied on the right-hand side

$$\Delta\mathbf{R}_{\mathtt{O}_k}^{i,i+1}(\mathbf{w}_i^k + \delta\mathbf{w}_i^k) \approx \mathrm{Exp}(\mathbf{w}_i^k \Delta t)\mathrm{Exp}(\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t) \tag{4.8}$$

Assuming that the term $\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t$ is small, we can rewrite the equation as:

$$\Delta\mathbf{R}_{\mathtt{O}_k}^{i,i+1}(\mathbf{w}_i^k + \delta\mathbf{w}_i^k) \approx \mathrm{Exp}(\mathbf{w}_i^k \Delta t)(\mathbf{I} + [\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\delta\mathbf{w}_i^k \Delta t]_\times), \tag{4.9}$$

where $[\cdot]_\times$ is the skew operator that transforms a vector in $\mathbb{R}^3$ into a skew symmetric matrix. Therefore, the partial derivative of $\Delta\mathbf{R}_{\mathtt{O}_k}^{i,i+1}$ with respect to the angular velocity can be written as follows:

$$\frac{\partial \Delta\mathbf{R}_{\mathtt{O}_k}^{i,i+1}}{\partial \delta \mathbf{w}_i^k} = \mathrm{Exp}(\mathbf{w}_i^k \Delta t)[\mathbf{J}_r(\mathbf{w}_i^k \Delta t)\Delta t]_\times \tag{4.10}$$

Finally, our local BA optimization for n cameras, m dynamic points, p objects and q map points is the following:

$$\hat{\theta} = \arg\min_\theta \sum_{i=1}^{n}(\sum_{j=1}^{m}\sum_{k=1}^{p} \mathbf{e}_{\mathbf{dyn}}^{i,j,k} + \sum_{l=1}^{q} \rho(\|\mathbf{e}_{\mathbf{repr}}^{i,l}\|_{\Sigma_i^l}^2)), \tag{4.11}$$

where both error terms optimize the camera pose and, more specifically, the term $\mathbf{e}_{\mathbf{repr}}^{i,l}$ is in charge of optimizing the static structure and the term $\mathbf{e}_{\mathbf{dyn}}^{i,j,k}$ (Eqn. 4.12) optimizes the dynamic agents' structure and trajectory.

$$\mathbf{e}_{\mathbf{dyn}}^{i,j,k} = \rho(\|\mathbf{e}_{\mathbf{repr}}^{i,j,k}\|_{\Sigma_i^j}^2) + \rho(\|\mathbf{e}_{\mathbf{vcte,XYZ}}^{i,j,k}\|_{\Sigma_{\Delta t}}^2) + \rho(\|\mathbf{e}_{\mathbf{vcte}}^{i,k}\|_{\Sigma_{\Delta t}}^2) \tag{4.12}$$

ρ is the robust Huber cost function to downweight outlier correspondences and Σ is the covariance matrix. In the case of the reprojection error Σ is associated to the scale of the key point in the camera i observing the points l and j respectively. For the two other error terms Σ is associated to the time interval between two consecutive observations of an object, *i.e.*, the longer time the more uncertainty there is about the constant velocity assumption. $\theta = \{\mathbf{T}_{\mathtt{CW}}^i, \mathbf{T}_{\mathtt{WO}}^{k,i}, \mathbf{X}_{\mathtt{W}}^l, \mathbf{X}_{\mathtt{O}}^{j,k}, \mathbf{v}_i^k, \mathbf{w}_i^k\}$ are the parameters to be optimized per iteration, and their optimized representation is $\hat{\theta} = \{\hat{\mathbf{T}}_{\mathtt{CW}}^i, \hat{\mathbf{T}}_{\mathtt{WO}}^{k,i}, \hat{\mathbf{X}}_{\mathtt{W}}^l, \hat{\mathbf{X}}_{\mathtt{O}}^{j,k}, \hat{\mathbf{v}}_i^k, \hat{\mathbf{w}}_i^k\}$.

On the one hand, the factor graph representation for this bundle adjustment problem with dynamic objects can be seen in Fig. 4.4. On the other hand, Fig. 4.5 shows the boolean Hessian matrix ($\mathbf{H}$) of the described problem. The Hessian can be built from the Jacobian matrices associated to each edge in the graph. In order to have a non-zero (i, j) block matrix, there must be an edge between i and j node in the factor graph. For this example we have defined a set of 5 key frames (KFs), 1 moving object (Objects) that has 10 3D points (OPs) visible in all 5 key frames, and 10 static map points (MPs) also visible in all the key frames. Notice the difference in the sparsity patterns of the static map points and the dynamic object points. The size of the matrix $\mathbf{H}$ is dominated by the number of map points and object points, which in typical problems is several orders of magnitude larger than the number of cameras and objects.

4.2.5 Bounding Boxes

Some approaches in the current literature would consider the tracking of dynamic objects to be complete with the contributions so far. That is, for every dynamic object in the scene we have an estimation of the trajectory of the centroid of its map points when it was first observed, as well as a point cloud representation. Examples of these works are ClusterSLAM (Huang et al. (2019)) and VDO-SLAM (Zhang et al. (2020)). However, we believe that it is also of key importance to find a common spatial reference for objects of the same semantic class, as well as an estimate of their dimensions and most important their space occupancy.

Alternatively, the basis of the system CubeSLAM by Yang & Scherer (2019) and of the work by Li et al. (2018) is the discovery of object bounding boxes. Only once

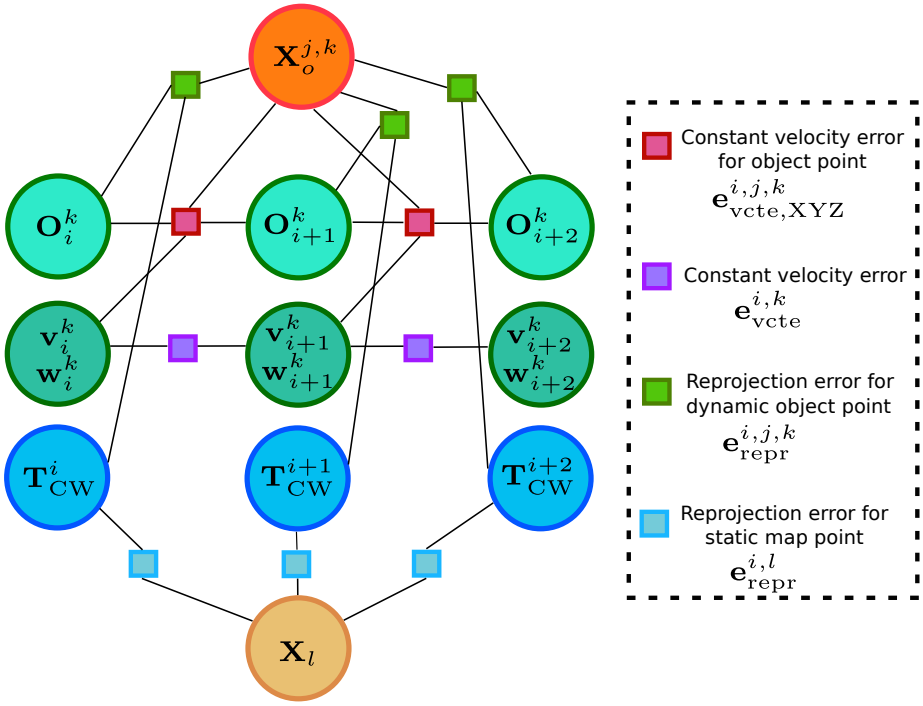


Figure 4.4: BA factor graph representation with dynamic objects.

bounding boxes are discovered, objects are tracked along frames. That is, if the camera viewing angle does not allow to estimate an object's bounding box (back view or partial view), the tracking of the object does not take place. Whereas this is not a problem for Li et al. (2018) because CNNs are by nature robust to partial views of objects, CubeSLAM struggles to initialize bounding boxes from views of occluded objects.

We propose to decouple the estimation of the trajectories and the bounding boxes of the observed dynamic objects. The former provides the system tracking with rich clues for ego-motion estimation, and the conjunction of both are useful to understand the dynamics of the surroundings. The output of the data association and the bundle adjustment stages contains the camera poses, the structure of the static scene and the dynamic objects, and the 6 DoF trajectory of one point for each object. This one point is the center of mass of the object's 3D points when it is first observed. Even though the center of mass changes along time with new object point's observations, the object pose that is tracked and optimized is referred to this first center of mass. To have a full understanding of the moving surroundings, it is of high importance to know the object's dimensions and more important their space occupancy. Tackling the two problems independently allows us to track dynamic objects from the first frame in which they appear independently of the camera-object view point.

We initialize an object bounding box by searching two perpendicular planes that fit roughly the majority of the object points. We hypothesize that even though objects are not always perfect cuboids, many can approximately fit a 3D bounding box. In the case in which only one plane is found, we add a prior on the rough dimensions of the non-observable direction that is related to the object class. This procedure is done within a RANSAC scheme: we choose the computed 3D bounding box that has the largest intersection over union (IoU) of its image projection with the CNN 2D bounding box. This bounding box is computed once for every different object track.

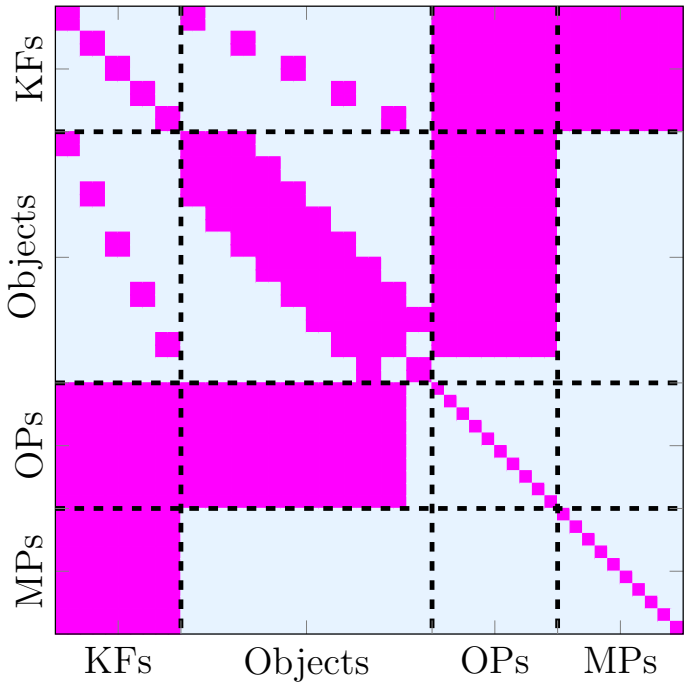


Figure 4.5: Hessian matrix for 5 key frames (KFs), 1 dynamic object (Objects) with 10 object points (OPs) visible in all 5 key frames, and 10 static map points (MPs). Key frames and points have 6 and 3 variables respectively representing their pose, and each object has 12 variables that represent its 6 DoF pose and 6 DoF velocity.

To refine this bounding box estimation –both the dimensions and the 6 DoF pose relative to the object tracking reference–, an image-based optimization is performed within a temporal window. This optimization seeks to minimize the distance between the image projection of the 3D bounding box and the CNN 2D bounding box prediction. Given that this problem is not observable for less than three views of an object, this is only performed once an object has at least three observing key frames. Also, to constraint the solution space in case the view of an object makes this problem non-observable (*e.g.*, a car observed from the back or from one lateral), a soft prior about the object dimensions is included. Finally, the initial bounding box 6 DoF pose is set as a prior so that the optimization solution remains close. Since this prior is tightly related to the object class coming from the CNN, we believe that adding this soft prior does not mean a loss of generality.

One can see in Fig. 4.6 the effect of the different errors and priors we make use of in our optimization. First, all three objects (Figs. 4.6a, 4.6b and 4.6c) yield the same 2D image projection so, unless an object is observed in at least three frames at different view points, more constraints are required to render the problem observable. Second, forcing the object points to be near the planes found within the point clouds constraints most cases. Third, a prior about the object's dimensions is needed, otherwise, cases such as the ones in Figs. 4.6b and 4.6c are not fully constrained.

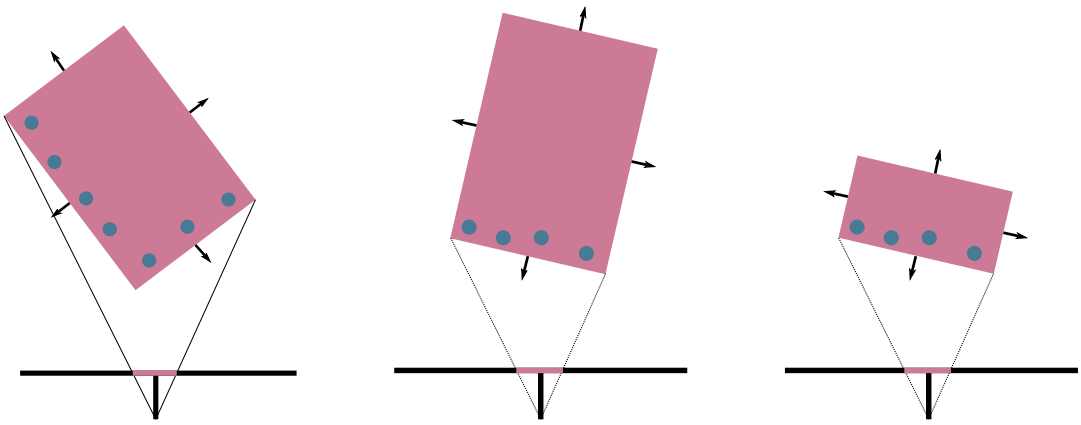


Figure 4.6: Examples of camera and bounding box configurations.

4.3 Experiments

In this section we detail the experiments that we have conducted to prove the validity of DynaSLAM II. We divide this section in two main building blocks: one that evaluates the advantages and influence of tracking dynamic objects in the camera ego-motion estimation (Subsection 4.3.1), and one that analyzes and discusses the performance of the multi-object tracking (Subsection 4.3.2).

4.3.1 Visual Odometry

For the visual odometry experiments we have chosen the KITTI raw (Table 4.1), tracking (Table 4.2) and odometry (Table 4.3) datasets by Geiger et al. (2013). These three datasets contain sequences of urban and road scenes recorded from a car perspective with multiple vehicles and pedestrians in circulation. The camera ego-motion trajectory is provided through a GPS tracker. The KITTI sequences of the raw and odometry datasets have been recorded with a stereo gray-scale camera, as opposed to the ones in the tracking dataset, which have also been recorded with a stereo RGB camera.

seq	ORB-SLAM2			DynaSLAM			ClusterSLAM			ClusterVO			Ours		
	AT	RP_t	RP_R	AT	RP_t	RP_R	AT	RP_t	RP_R	AT	RP_t	RP_R	AT	RP_t	RP_R
0926-0009	0.83	1.85	0.01	0.81	1.80	0.01	0.92	2.34	0.03	0.79	2.98	0.03	**0.78**	**2.11**	**0.01**
0926-0013	0.32	1.04	0.01	0.30	0.99	0.01	2.12	5.50	0.07	**0.26**	1.16	0.01	0.28	**0.98**	**0.00**
0926-0014	0.50	1.22	0.01	0.60	1.62	0.01	0.81	2.24	0.03	**0.48**	**1.04**	**0.01**	0.55	1.51	**0.01**
0926-0051	0.38	1.16	0.00	0.46	1.17	0.00	1.19	1.44	0.03	0.81	2.74	0.02	**0.45**	**1.21**	**0.00**
0926-0101	2.97	13.63	0.03	3.52	15.14	0.03	4.02	**12.43**	**0.02**	**3.18**	12.78	**0.02**	4.72	18.82	0.04
0929-0004	0.62	1.38	0.01	0.56	1.36	0.01	1.12	2.78	0.02	**0.40**	1.77	0.02	0.62	**1.40**	**0.01**
1003-0047	20.49	32.59	0.08	2.87	5.95	0.02	10.21	8.94	0.06	4.79	**6.54**	0.05	**3.37**	9.17	**0.03**

Table 4.1: Egomotion comparison on the KITTI raw dataset. The absolute trajectory error (AT) is given in m, the translation relative pose error (RP_t) in m too and the rotational relative pose error (RP_R) is given in radians.

One can find in all three tables a comparison of our system's performance against ORB-SLAM2 and DynaSLAM. ORB-SLAM2 is the base SLAM system on which we build DynaSLAM II, and does not specifically address dynamic objects. DynaSLAM

seq	ORB-SLAM			DynaSLAM			VDO-SLAM			Ours		
	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R
0000	1.32	0.04	0.06	1.35	0.04	0.06	-	0.05	**0.05**	1.35	**0.04**	0.06
0001	1.95	0.05	0.04	2.42	0.05	0.04	-	0.12	0.04	2.14	**0.05**	0.04
0002	0.95	0.04	0.03	1.04	0.04	0.03	-	0.04	0.02	0.85	0.04	0.02
0003	0.74	0.07	0.04	0.78	0.07	0.04	-	0.09	0.04	0.63	**0.06**	0.04
0004	1.44	0.07	0.06	1.52	0.07	0.06	-	0.11	**0.05**	1.45	**0.07**	0.06
0005	1.23	0.06	0.03	1.22	0.06	0.03	-	0.10	**0.02**	1.30	**0.06**	0.03
0006	0.19	0.02	0.04	0.19	0.02	0.04	-	0.02	0.05	0.17	0.02	**0.04**
0007	2.47	0.05	0.07	2.69	0.05	0.07	-	-	-	3.53	0.05	0.07
0008	1.40	0.08	0.04	1.29	0.08	0.04	-	-	-	1.55	0.07	0.04
0009	4.00	0.06	0.05	3.55	0.06	0.05	-	-	-	4.52	0.06	0.05
0010	1.68	0.07	0.04	1.84	0.07	0.04	-	-	-	1.30	0.07	0.04
0011	0.97	0.04	0.03	1.05	0.04	0.03	-	-	-	0.97	0.04	0.03
0012	0.01	0.00	0.01	0.01	0.00	0.01	-	-	-	0.01	0.00	0.01
0013	1.18	0.04	0.05	1.18	0.04	0.05	-	-	-	1.02	0.04	0.05
0014	0.13	0.03	0.08	0.13	0.03	0.08	-	-	-	0.11	0.03	0.08
0015	0.23	0.02	0.02	0.22	0.02	0.02	-	-	-	0.21	0.03	0.03
0016	0.02	0.00	0.01	0.02	0.00	0.01	-	-	-	0.02	0.01	0.02
0017	0.04	0.00	0.01	0.04	0.00	0.01	-	-	-	0.04	0.00	0.02
0018	0.89	0.05	0.03	1.00	0.05	0.03	-	0.07	**0.02**	1.09	**0.05**	0.03
0019	2.31	0.05	0.03	2.35	0.05	0.03	-	-	-	2.23	0.05	0.03
0020	16.80	0.11	0.07	1.10	0.05	0.04	-	0.16	**0.03**	2.15	**0.05**	0.04

Table 4.2: Egomotion comparison on the KITTI tracking dataset. The absolute trajectory error (ATE) is given in m, and the relative pose error for translation (RPE_t) and rotation (RPE_R) are given respectively in m/frame and °/frame.

provides ORB-SLAM2 with the capability to detect dynamic objects also with Mask R-CNN but uniquely ignores the features belonging to dynamic classes and does not track them. The difference in the results of ORB-SLAM2 and DynaSLAM gives an idea of how dynamic each sequence is. Theoretically, if dynamic objects are representative in the scene and they are in circulation, DynaSLAM's performance is better, as can be seen in the sequence 1003-0047 in Table 4.1, 0020 in Table 4.2 and 04 in Table 4.3. However, if dynamic objects are representative in the scene but they are not in motion, *e.g.*, parked vehicles, DynaSLAM shows a higher trajectory error. This happens because the tracking thread does not make use of the features belonging to the static vehicles, which are useful for pose estimation and usually lay in nearby scene regions. This can be seen for example in the sequence 0001 in Table 4.2 and in most sequences in Table 4.3. Besides that, DynaSLAM II achieves a performance better than both ORB-SLAM and DynaSLAM in these two types of scenarios in many of the evaluated sequences. On the one hand, when dynamic instances are moving, DynaSLAM II successfully estimates the velocity of the corresponding objects and provide the bundle adjustment with rich clues for camera pose estimation when the static representation is not sufficient. This often occurs when dynamic objects occlude nearby regions of the scene and therefore static features can only provide valuable hints for accurately estimating the camera rotation. On the other hand, when instances belonging to dynamic classes are static, *e.g.*, parked vehicles, DynaSLAM II tracks their features estimating that their velocity is approximately close to zero. Consequently, these object points act much like 3D static map points.

Tables 4.1 and 4.2 present our ego motion results compared to those of state-

seq	ORB-SLAM Mur-Artal et al. (2015)			DynaSLAM Bescos et al. (2018)			Ours		
	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R	ATE	RPE_t	RPE_R
00	**1.38**	**1.30**	**0.90**	1.43	1.32	0.92	**1.38**	1.42	1.02
01	10.79	1.89	**0.40**	**9.74**	**1.87**	**0.40**	10.05	1.88	0.48
02	**5.59**	**1.15**	**0.51**	6.75	1.28	0.74	7.85	1.36	0.91
03	**0.64**	**0.77**	0.38	0.65	0.78	**0.36**	0.92	1.02	0.47
04	0.85	1.22	0.60	0.78	1.10	0.52	**0.18**	**0.65**	**0.23**
05	0.72	**0.74**	**0.42**	**0.71**	0.75	0.43	0.89	0.80	0.46
06	0.57	0.73	**0.37**	**0.56**	0.69	0.41	0.72	**0.63**	0.65
07	0.56	**0.76**	0.60	0.56	0.77	**0.58**	**0.47**	0.93	0.76
08	**3.58**	**1.79**	**0.69**	4.21	1.80	0.71	**3.58**	1.87	0.71
09	2.90	**1.01**	**0.41**	3.33	1.02	0.50	**2.12**	1.12	0.64
10	**1.27**	0.99	**0.58**	**1.27**	**0.97**	**0.58**	1.58	0.99	0.60

Table 4.3: Egomotion comparison on the KITTI odometry dataset. The absolute trajectory error (ATE) is given in m, and the relative pose error for translation (RPE_t) and rotation (RPE_R) are given respectively in m/100m and °/100m.

of-the-art systems that also track dynamic objects within a joint SLAM framework. ClusterSLAM (Huang et al. (2019)) acts as a SLAM back end rather than a whole SLAM system and is therefore highly dependent on the camera poses initial estimates. ClusterVO (Huang et al. (2020)) and VDO-SLAM (Zhang et al. (2020)) are SLAM systems as ours with the multi-object tracking capability. The former can handle stereo and RGB-D data, whereas the former only handles RGB-D data. The reported errors are given with different metrics so that we can directly use the values that the authors provide in their papers. DynaSLAM II achieves in all sequences a translational relative error (RPE_t) which is lower than that of VDO-SLAM. However, VDO-SLAM usually achieves a lower rotational pose error. Since far map points are the ones that provide the richest clues for rotation estimation, we believe that this difference in accuracy does not depend on the object tracking performance and is therefore due to the underlying camera pose estimation algorithm and the sensor suite. Regarding the performance of ClusterVO, it achieves an accuracy which is in most sequences quite similar to ours.

4.3.2 Multi-Object Tracking

Once demonstrated the utility and the value of tracking dynamic objects for ego motion estimation, we have chosen once again the KITTI tracking dataset (Geiger et al. (2013)) to validate our multi-object tracking results . This dataset has 21 sequences of urban and road scenes recorded from a car perspective with multiple vehicles and pedestrians in circulation. The ego car is equipped with a LIDAR sensor, a GPS tracker and a stereo RGB camera. The trajectory of the ego car is provided through the GPS data, and the trajectories and the 3D bounding boxes of the dynamic objects are provided thanks to expensive manual annotations on the LIDAR 3D point clouds.

First of all, we want to refer the reader to Fig. 4.1 to have a look at some of our multi-object tracking qualitative results in this dataset. For example, in Fig. 4.1c the two tracked cars show perfectly parallel trajectories (lines in the map view) and a good estimate of the occupied space. Furthermore their absolute velocities (written in the frame view) are similar and are coherent with the driving context. Also, we would like to draw the reader's attention to Fig. 4.1d, where the 3D bounding box of the car on

the left is well estimated despite its partial view. This scene is also challenging because the front car is far from the camera and it is though correctly tracked.

In the last decade Bernardin & Stiefelhagen (2008) introduced the CLEAR MOT metrics, two intuitive and general metrics (MOTP and MOTA) to allow for objective comparison of tracker characteristics, focusing on their precision in estimating object locations, their accuracy in recognizing object configurations and their ability to consistently label objects over time. Whereas these metrics are well established in the computer vision and robotics communities and provide valuable insights about the per-frame performance of trackers, they do not take into account the quality of the tracked objects trajectories (see Fig. 4.7)). Therefore, we suggest that to correctly evaluate multi-object tracking within a SLAM framework, one needs to report the CLEAR MOT metric MOTP [1] as well as the common trajectory error metrics. Most related works on SLAM and multi-object tracking only report the CLEAR MOT metric MOTP (Yang & Scherer (2019), Huang et al. (2020), Li et al. (2018)). Besides that, the authors of VDO-SLAM (Zhang et al. (2020)) uniquely report the relative pose error of all the objects trajectories of one sequence as a single ensemble. We believe that this metric should be instead reported for individual trajectories to provide a clearer understanding and an easier comparison.

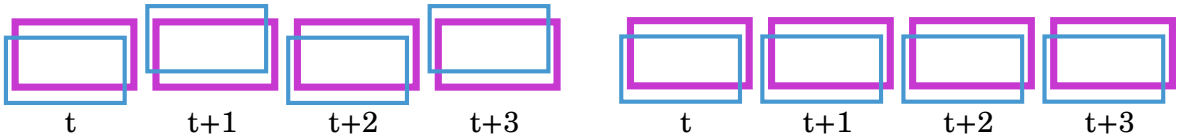


Figure 4.7: Required metrics for multi-object tracking. These two images represent the ground-truth pose of an object along time (magenta) and the estimation of a random tracker of this object's pose (blue). Both setups of estimation and ground truth deliver the same CLEAR MOT metrics, however, the trajectory estimated in (a) is of a worse quality than the estimated in (b). In a nutshell, these metrics have to be accompanied with the relative objects trajectories evaluation to provide comprehensive results.

One can find in Table 4.4 the evaluation of our results for the multi-object tracking task in the KITTI tracking dataset. For some of the sequences in this dataset we have chosen the objects that we considered to be the most suitable for a trajectory evaluation. These chosen objects are labeled with their ground-truth object id. For each of these ground truth trajectories we look for the most overlapping bounding boxes in our estimations. In the case of the trajectory metrics (ATE and RPE) and the 2D MOTP, this overlapping has to be of at least 25 % to count as a true positive, and we compute it as the intersection over union (IoU) of the 3D bounding boxes projected over the current frame. For the other two evaluations (BV and 3D), the overlapping has to be also of at least 25 % to count as a true positive, and is computed as the IoU of the bounding boxes in bird view and in 3D respectively.

This evaluation gives an idea of the tracking performance of our framework and of the quality of our bounding boxes. Regarding the percentage of true positives (TP), we can see that objects are tracked for the majority of their trajectory. Missing detections usually occur because the objects lay far from the camera and the stereo matching

[1] MOTP stands for multiple object tracking precision. It is defined as the total precision of the predictions –computed with any given cost function– over the number of true positives.

	sequence	0003		0006	0010	0017	0018			0020			
	object id	0	1	4	0	6	1	2	3	0	2	3	9
	class	car	car	car	car	person	car	car	car	car	car	car	car
	ATE $[m]$	0.22	0.31	0.40	0.86	0.96	0.84	0.67	0.26	0.30	0.39	0.18	0.77
	RPE_t [m/m]	0.13	0.23	0.26	0.26	1.12	0.25	0.30	0.27	0.31	0.49	0.15	0.19
	RPE_R [°/m]	1.47	11.24	2.21	15.29	9.13	12.12	14.30	25.88	1.30	10.18	1.22	0.96
2D	# TP	48	50	25	163	65	100	189	160	122	36	61	88
	TP (%)	63	41	53	55	45	53	72	56	61	65	86	94
	MOTP [%]	80	79	83	73	63	59	86	66	63	63	75	79
BV	# TP	48	48	25	135	24	75	150	92	120	36	67	88
	TP (%)	63	39	53	46	17	39	57	32	60	65	94	94
	MOTP [%]	69	59	63	52	32	54	47	45	46	53	66	47
3D	# TP	48	48	25	124	0	57	125	63	110	30	67	83
	TP (%)	63	39	53	42	0	30	47	22	55	55	94	88
	MOTP [%]	56	47	49	45	0	54	41	38	35	33	50	35

Table 4.4: Objects motion comparison on the KITTI tracking dataset

does not provide accurate enough features for a rich tracking. It is important to notice that the accuracy of the passersby tracking is much lower than that of the cars due to their non-rigid shape –see the column for sequence 0017–. The trajectory errors of the cars are acceptable, however, these results are far from the ego-motion estimation performance. Our intuition is that the own feature-based nature of our algorithm limits its tracking accuracy. A larger amount of 3D points per object would always provide richer clues for object tracking.

So that we can provide an evaluation about all object detections and not only of those with long trajectories –for example, also the evaluation of the bounding box estimation of parked cars–, we also compute the MOTP of the whole KITTI tracking dataset with the KITTI 3D object detection benchmark. This allows us to directly compare our multi-object tracking results to those of other state-of-the-art similar systems (Table 4.5). The CNNs of Chen et al. (2017) and specially of Li et al. (2018) achieves amazing results thanks to the single-view network accuracy itself and the multi-view refinement approach of the latter one, to the detriment of a loss of generality. On the other hand, the accuracy of Bârsan et al. (2018) and Huang et al. (2020) in detecting bounding boxes is remarkable, but it is very sensitive to object truncation and occlusion. Our results show that we can handle objects truncation and occlusion with a minor loss in precision. However, less bounding boxes are usually discovered than with the other methods. Our intuition is that the feature-based nature of our system renders this step specially challenging, opposite to that of the work by Bârsan et al. (2018), which computes dense stereo matching to discover bounding boxes.

4.3.3 Timing Analysis

To complete the evaluation of our proposal, Table 4.6 shows the average computational time for its different building blocks. The timing of DynaSLAM II is obviously highly dependent on the number of objects to be followed. In sequences like the KITTI tracking 0003 there are only two objects at a time as maximum and it can therefore run at 12 fps. However, the sequence 0020 can have up to 20 objects at a time and its performance is seen slightly compromised, but it still achieves a real time performance at approximately 10 fps. The local mapping thread also runs with a real-time performance in most cases bringing important improvements to the trajectories estimation.

	$MOTP_{BV}$			$MOTP_{3D}$		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Chen et al. (2017)	81.34 %	70.70 %	66.32 %	80.62 %	70.01 %	65.76 %
Li et al. (2018)	**88.07 %**	**77.83 %**	**72.73 %**	**86.57 %**	**74.13 %**	**68.96 %**
Bârsan et al. (2018)	71.83 %	47.16 %	40.30 %	**64.51** %	43.70 %	37.66 %
Huang et al. (2020)	**74.65 %**	49.65 %	45.62 %	55.85 %	38.93 %	33.55 %
Ours	64.69 %	**58.75 %**	**58.36 %**	53.14 %	**48.66 %**	**48.57 %**

Table 4.5: MOTP evaluation on the KITTI tracking dataset. All ground-truth 3D detections are divided into three categories: Easy, Moderate and Hard, based on the height of 2D reprojected bounding box and the occlusion and truncation level.

Sequence	Building block	Time [ms]
KITTI tracking 0003	Tracking thread	80.10 ±0.78
	Local BA	61.37 ±6.70
	Bounding Boxes BA	0.07 ±0.01
KITTI tracking 0005	Tracking thread	86.46 ±0.82
	Local BA	69.90 ±16.87
	Bounding Boxes BA	0.17 ±0.02
KITTI tracking 0020	Tracking thread	94.56 ±1.27
	Local BA	65.03 ±17.72
	Bounding Boxes BA	0.60 ±0.05

Table 4.6: DynaSLAM II average computational time.

We do not include within these numbers the computational time of the semantic segmentation building block since the 2D object detection time depends on the GPU power and CNN model complexity. Many algorithms such as YOLACT by Bolya et al. (2019) can run in real time and provide high-quality instance masks.

Finally, Table 4.7 collects the average timing results for the systems that jointly perform SLAM and multi-object tracking in the KITTI dataset. DynaSLAM II is the only system that can provide at present a real-time solution.

	Li et al. (2018)	Zhang et al. (2020)	Huang et al. (2019)	Huang et al. (2020)	Ours
fps	5.8	5 - 8	7	8	**10 - 12**

Table 4.7: DynaSLAM II frames per second compared against the state of the art.

4.4 Failure Modes and Future Work

The feature-based core of our system limits its applicability for 3D object tracking when objects do not have a high texture or when they lay far from the camera. We would therefore like to improve the object temporal correlation by fully exploiting the dense visual information. We would also like to explore the –even more– challenging task of multi-object tracking and SLAM with only a monocular camera. This is an interesting direction since dynamic object tracking can provide rich clues about the scale of the map and correct the usual scale drift that occurs in monocular systems.

4.5 Discussion

In this chapter we have proposed a 3D objects and ego-motion tracking system for stereo and RGB-D cameras. We integrate instances semantic priors with sparse feature measurements into a tightly-coupled optimization framework. This allows the estimation of both the camera and the dynamic objects to be mutually beneficial. We propose an object level SLAM with novel measurement functions between cameras, objects, 3D points. We decouple the problem of object tracking from that of bounding boxes estimation and, differently from other works, we do not make any assumptions about the objects motion, pose or model. This renders our framework suitable for a large number of applications. We demonstrate with our experiments that DynaSLAM II achieves a state-of-the-art accuracy at near real time performance.

4.6 Related Publications

1. Berta Bescos, Carlos Campos, Juan Tardós , José Neira. "DynaSLAM II: Tightly-Coupled Multi-Object Tracking and Visual SLAM". This work is in preparation and will be submitted in October 2020 for publication at *IEEE Robotics and Automation Letters* and presentation at *IEEE International Conference on Intelligent Robots and Systems*.

Chapter 5

Conclusions and Future Prospects

In this thesis we have addressed the problem of visual Simultaneous Localization and Mapping in real-world environments populated with dynamic objects. We would like to provide a brief summary of the major findings of this research and also draft our views on future developments. We believe that important research directions and applications are emerging right now in topics tightly related to our work.

5.1 Detection of Dynamic Objects

In this part of the thesis, we have addressed the core problem of detecting dynamic objects in localization and mapping frameworks. Our contributions enhance the performance of VO and SLAM systems in challenging real-world populated environments and enable robots to localize reliably in long-term applications thanks to the creation of long-term stable and reusable 3D maps. The proposed methods make it possible to localize robots in highly dynamic environments with a state-of-the-art accuracy that is similar to that usually achieved when the scene is completely static. To this end, we have leveraged semantic segmentation from deep learning to detect *a priori* dynamic classes, together with multi-view geometry algorithms that detect scene motion. We perceive the following directions as the possible next developments in the field.

- Semantic Understanding: Intuitively, when memorizing a place, humans do not pay attention to the specific cars or passersby but only pay attention to the stable and representative scene landmarks such as buildings. As we have demonstrated with our research, understanding the semantic information of the scene allows to ignore the dynamic objects from the scene that will most likely not reappear again. Whereas the current state-of-the-art solutions to visual SLAM in dynamic environments as ours explicitly blend hand-crafted features with semantic segmentation, we believe that these hand-crafted features and algorithms we use nowadays will shift over time towards semantically labeled primitives and objects, since they are more reliable to detect and less vulnerable to changes in viewpoint and illumination. That is, a good feature-detector aimed at visual odometry and place recognition should not extract information from the dynamic objects in the scene, and should therefore only contain clues about the static scene representation, which is exactly what humans do when memorizing a place.

- End-to-End Localization: In the recent years, due to the rapid development of deep learning and the outstanding contributions of CNNs in the field of image

classification (Krizhevsky et al. (2012)) and segmentation (Badrinarayanan et al. (2017)), data driven approaches for end-to-end visual odometry have begun to be explored. These learning-based methods do not require expensive feature extraction and matching or complex geometric operations. The recent publications on this topic of Wang et al. (2017) and Jiao et al. (2019) among others demonstrate initial promising results. Intuitively, these methods would learn to ignore clues coming from dynamic classes or instances even without an explicit use of a semantic visual description or an optical flow representation. However, even the end-to-end state-of-the-art methods still struggle to generalize well to various environments and deliver an acceptable accuracy. Moreover, to obtain these far from satisfactory results, enormous amounts of training data are required. While it is not clear whether data-driven approaches will eventually surpass traditional methods, it is certain that blended learning and geometry systems are increasingly representing the state of the art in visual SLAM in dynamic environments.

5.2 Static Background Inpainting

In this second part of the thesis, we have presented approaches that address the challenge of realistically inpainting in images the plausible static background occluded by dynamic objects. To the best of our knowledge, we are the first ones to use this approach in the context of visual localization and mapping. The proposed method is novel in a robotics context and brings improvements and ease of use mainly in visual place recognition and mapping, and also –although less notorious– in visual odometry in presence of dynamic objects. With this contribution we have paved the way to research localization in dynamic environments following this new direction. It is clear, particularly in view of recent progress in deep learning, that image and video-based inpainting will continue to evolve and improve, providing new and better tools for inpainting in robotics. As following lines of research on this topic, we would like to extend our solutions to also include inpainting in depth maps rather then only in color images. This would enhance the quality of our results, mainly in indoor scenes where depth information is more easily accessible, and would create new research opportunities.

From a broader perspective, we envision that place recognition and mapping solutions dealing with dynamic and spurious data would follow these directions.

- End-to-End Place Recognition: Whereas deep-learning-based visual odometry approaches struggle to achieve an accuracy comparable to that of conventional methods, deep-learning-based visual place recognition significantly outperforms non-learnt image representations. Despite the success of methods trained end-to-end for the place recognition task in scenes with large amount of clutter like the one by Arandjelović et al. (2016), we have demonstrated with our place recognition experiments in Subsection 3.5.2 that they see their performance significantly boosted with explicit static scene representations. We can then foresee that the key to achieving state-of-the-art performance in visual place recognition in dynamic environments is the use of end-to-end trained methods through an explicit use of the static scene semantic description.

- Dense Deep Mapping: Conventionally, dense representations of the environment are known to require a large number of parameters. However, in the recent years,

the computer vision and robotics communities have seen attempts to undermine this claim as in the works by Bloesch et al. (2018) and Czarnowski et al. (2020). The geometry of natural scenes is not a random collection of occupied and unoccupied space but exhibits a high degree of order. In a depth map, the values of neighbouring pixels are highly correlated and can often be accurately represented by well known geometric smoothness primitives. But more strongly, if a higher level of understanding is available, a scene could be decomposed into a set of semantic objects together with some internal parameters and a pose, as in the work by Salas-Moreno et al. (2013). Regarding dynamic environments, explicitly modifying these high level space representations not to contain either dynamic or non-stable objects is a much more simple assignment, and we believe that it might allow to infer static and complete dense 3D environment representations.

5.3 Multi-Object Tracking

In this third and final part of the thesis, we have undertaken the challenging task of conjointly solving visual SLAM and multi-object tracking. Our contributions further advance the state of the art of the –not so explored– task of simultaneous localization, mapping and multi-object tracking, and establish some of the foundations of this approach. We propose a bundle adjustment solution that tightly optimizes the scene structure, the camera poses and the objects trajectories. The objects' bounding boxes are also optimized in a decoupled formulation that allows to estimate the 6 DoF pose of the objects without being tailored for special use cases.

We believe that our research and development efforts demonstrate the value of simultaneous localization, mapping and multi-object tracking technology in robotics. The tasks of estimating the trajectory of the ego camera and the trajectory of the scene dynamic agents complement each other and renders the technology more versatile. We consider that in the years to come this approach will gain popularity due to the importance of understanding surrounding dynamic objects for frontier requirements of emerging applications within AR/VR or autonomous systems navigation.

Traditionally, the problem of 6 DoF object pose estimation is tackled by matching feature points between 3D models and images (Lowe (1999), Rothganger et al. (2006), Collet et al. (2011)). However, these methods require that there are rich textures on the objects in order to detect feature points for matching. As a result, they are unable to handle texture-less objects. On the other hand, in the latest years, deep learning has shown to be effective for robust and real-time monocular pose estimation (Xiang et al. (2017), Peng et al. (2019), Zakharov et al. (2019)). Whereas these methods outperform traditional hand-crafted image representations regardless of the objects texture and with a great robustness to occlusions and partial views, they need to be trained for specific object classes with the consequent loss of generality. We believe that achieving successful results for a joint estimation of camera and objects pose with end-to-end approaches is far from being the case today. The two independent problems have to be first solved without a generality loss. However, blended learning and geometry systems already represent the state of the art for specific use cases. We envision that the next steps in this direction of research will bring increasing performance and decreasing loss of generality by balancing the advantages and disadvantages of learning and geometry.

Chapter 6

Summary of Results

6.1 Research Stays

During the years of the thesis I have made the following research stays abroad:

- I was at the Autonomous Systems Laboratory of ETH Zürich (Switzerland) from February 1, 2018 to June 30, 2018. I had the opportunity to meet and work together with Dr. Cesar Cadena and Prof. Dr. Roland Siegwart, as well as to share the laboratory and the working hours with their wonderful students.

- During the summer of 2019 I did an internship at the Facebook Reality Labs also in Zürich, Switzerland (June 24, 2019 - September 13, 2019). My direct supervisor during this stay was Dr. Mariano Jaimez. I also enjoyed the supervision of SLAM experts such as Dr. Christian Forster, among others.

6.2 Supervision of Students

Aside from research, I have also participated in teaching, imparting the "Computer Vision" and "Machine Learning" courses together with my supervisor Prof. José Neira at the University of Zaragoza. Also, during the doctoral studies a significant effort was spent on supervising student projects. Below, all supervised students and projects are listed. For projects that resulted in a publication a citation is provided.

- Guoxiang Zhou (Semester Project in ETH Zurich – Spring 2018): "Dynamic Objects Segmentation for Visual Localization in Urban environments" (Zhou, Bescos, Dymczyk, Pfeiffer, Neira & Siegwart (2018)).

- Laura Delfau Luesma (Bachelor Thesis at University of Zaragoza – 2019): "A Dynamic-Object-Invariant Space via CycleGANs".

- Daniel Cay (Bachelor Thesis at University of Zaragoza – Spring 2020): "Multi-Object Tracking and Segmentation with Learnt Descriptors".

- Victor Sisqués (Bachelor Thesis at University of Zaragoza – Spring 2020): "Monocular Camera Pose Estimation Robust to Dynamic Objects".

6.3 Dissemination

6.3.1 Peer-Reviewed Publications

The research developed in this thesis has resulted in the following peer-reviewed publications in journals, conferences and workshops.

In preparation

- Berta Bescos, Carlos Campos, Juan Tardós , José Neira. "DynaSLAM II: Tightly-Coupled Multi-Object Tracking and Visual SLAM". This work is in preparation and will be submitted in October 2020 for publication at *IEEE Robotics and Automation Letters* and presentation at *IEEE International Conference on Robotics and Automation*.

Journals

- Berta Bescos, Cesar Cadena , José Neira. "Empty Cities: A Dynamic-Object-Invariant Space for Visual SLAM". *IEEE Transactions on Robotics*, accepted in September 2020.

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "DynaSLAM: Tracking, Mapping and Inpainting in Dynamic Scenes". *IEEE Robotics and Automation Letters* and oral presentation at *IEEE International Conference on Intelligent Robots and Systems*, October 2018.

Conferences

- Berta Bescos, José Neira, Roland Siegwart , Cesar Cadena. "Empty Cities: Image Inpainting for a Dynamic-Object-Invariant Space". *IEEE International Conference on Robotics and Automation*, May 2019.

Workshops

- Berta Bescos, Cesar Cadena , José Neira. "Dynamic-to-Static Image Translation for Visual SLAM". Oral and Poster Presentation within the Workshop Scene and Situation Understanding for Autonomous Driving at *IEEE International Conference on Robotics, Science and Systems*, June 2019. (**Winner Toyota Research Institute (TRI) Best Contribution Award**).

- Guoxiang Zhou, Berta Bescos, Marcin Dymczyk, Mark Pfeiffer, José Neira , Roland Siegwart. "Dynamic Objects Segmentation for Visual Localization in Urban Environments". Poster Presentation within the Workshop From freezing to jostling robots: Current challenges and new paradigms for safe robot navigation in dense crowds at *IEEE International Conference on Intelligent Robots and Systems*, October 2018.

- Berta Bescos, Jose M. Facil, Javier Civera and José Neira. "Detecting, Tracking and Eliminating Dynamic Objects in 3D Mapping using Deep Learning and Inpainting". Oral and Poster Presentation within the Workshop Representing a

Complex World: Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding at *IEEE International Conference on Robotics and Automation*, May 2018.

- Berta Bescos, Jose M. Facil , Javier Civera , José Neira. "Robust and Accurate 3D Mapping by combining Geometry and Machine Learning to deal with Dynamic Objects". Poster Presentation within the Workshop Learning for Localization and Mapping at *IEEE International Conference on Intelligent Robots and Systems*, September 2017.

6.3.2 Open-Source Software

We have released the following open-source software:

- **DynaSLAM** (`https://github.com/BertaBescos/DynaSLAM`)
- **Empty Cities** (`https://github.com/BertaBescos/EmptyCities_SLAM`)
- We plan to release the code for **DynaSLAM II** when we will submit this work for RA-L and ICRA (October 2020).

6.3.3 Videos

Demonstrating videos of DynaSLAM:

- `https://youtu.be/EabI_goFmQs`

Demonstrating videos of Empty Cities:

- `https://youtu.be/douCaMUVPxY`

Demonstrating videos of DynaSLAM II will also be available soon.

6.3.4 Conference and Research Seminar Attendance

I participated in the following courses or seminars offered outside the PhD program:

- I participated in the Robotic Vision Summer School (RVSS), organized by the Australian Center for Robotic Vision, during March 2017 in Kioloa, Australia.
- Poster presentation within the workshop at the IEEE conference Intelligent Robots and Operative Systems (IROS) 2017: *Learning for Localization and Mapping.*
- Oral and poster presentation within the workshop at the IEEE International Conference on Robotics and Automation (ICRA) 2018: *Representing a Complex World: Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding.*
- Oral and poster presentation within the conference IEEE conference Intelligent Robots and Operative Systems (IROS) 2018.

- Poster presentation within the workshop at the IEEE conference Intelligent Robots and Operative Systems (IROS) 2018: *From freezing to jostling robots: Current challenges and new paradigms for safe robot navigation in dense crowds.*
- I attended the IEEE Conference on Robot Learning (CoRL) 2018.
- Poster presentation within the IEEE International Conference on Robotics and Automation (ICRA) 2019.
- Oral and poster presentation within the workshop at IEEE conference on Robotics, Science and Systems (RSS) 2019: *Scene and Situation Understanding for Autonomous Driving.*

6.4 Peer Reviews

In addition, during the doctoral studies a significant effort was devoted to the review of journals and conference papers.

- Reviewer of 19 submissions to IEEE RA-L.
- Reviewer of 9 submissions to IEEE ICRA.
- Reviewer of 8 submissions to IEEE IROS.
- Reviewer of 2 submissions to IEEE RSS.
- Reviewer of 2 submissions to IEEE T-RO.
- Reviewer of 1 submission to IEEE CVPR.
- Reviewer of 1 submission to IEEE WACV.

Bibliography

Alahi, A., Ortiz, R. & Vandergheynst, P. (2012), Freak: Fast retina keypoint, in '2012 IEEE Conference on Computer Vision and Pattern Recognition', Ieee, pp. 510–517.

Alcantarilla, P. F., Yebes, J. J., Almazán, J. & Bergasa, L. M. (2012), On combining visual SLAM and dense scene flow to increase the robustness of localization and mapping in dynamic environments, in 'ICRA'.

Ambrus, R., Folkesson, J. & Jensfelt, P. (2016), Unsupervised object segmentation through change detection in a long term autonomy scenario, in 'Humanoid Robots (Humanoids)', IEEE.

Arandjelović, R., Gronat, P., Torii, A., Pajdla, T. & Sivic, J. (2016), NetVLAD: CNN architecture for weakly supervised place recognition, in 'IEEE Conference on Computer Vision and Pattern Recognition'.

Badrinarayanan, V., Kendall, A. & Cipolla, R. (2017), 'Segnet: A deep convolutional encoder-decoder architecture for image segmentation', IEEE transactions on pattern analysis and machine intelligence **39**(12), 2481–2495.

Bailey, T. & Durrant-Whyte, H. (2006), 'Simultaneous localization and mapping (slam): Part ii', IEEE robotics & automation magazine **13**(3), 108–117.

Barnes, D., Maddern, W., Pascoe, G. & Posner, I. (2018), Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments, in '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1894–1900.

Bârsan, I. A., Liu, P., Pollefeys, M. & Geiger, A. (2018), Robust dense mapping for large-scale dynamic environments, in '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 7510–7517.

Bay, H., Ess, A., Tuytelaars, T. & Van Gool, L. (2008), 'Speeded-up robust features (surf)', Computer vision and image understanding **110**(3), 346–359.

Bernardin, K. & Stiefelhagen, R. (2008), 'Evaluating multiple object tracking performance: the clear mot metrics', EURASIP Journal on Image and Video Processing **2008**, 1–10.

Bertalmio, M., Bertozzi, A. L. & Sapiro, G. (2001), Navier-stokes, fluid dynamics, and image and video inpainting, in 'Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001', Vol. 1, IEEE, pp. I–I.

Bescos, B., Fácil, J. M., Civera, J. & Neira, J. (2018), 'DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes', IEEE Robotics and Automation Letters **3**(4), 4076–4083.

Bescos, B., Neira, J., Siegwart, R. & Cadena, C. (2019), Empty Cities: Image Inpainting for a Dynamic-Object-Invariant Space, in '2019 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 5460–5466.

Besl, P. J. & McKay, N. D. (1992), Method for registration of 3-d shapes, in 'Sensor fusion IV: control paradigms and data structures', Vol. 1611, International Society for Optics and Photonics, pp. 586–606.

Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S. & Davison, A. J. (2018), Codeslam—learning a compact, optimisable representation for dense visual slam, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2560–2568.

Bolya, D., Zhou, C., Xiao, F. & Lee, Y. J. (2019), Yolact: Real-time instance segmentation, in 'Proceedings of the IEEE international conference on computer vision', pp. 9157–9166.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I. & Leonard, J. J. (2016), 'Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age', IEEE Transactions on robotics **32**(6), 1309–1332.

Chen, X., Kundu, K., Zhu, Y., Ma, H., Fidler, S. & Urtasun, R. (2017), '3d object proposals using stereo imagery for accurate object class detection', IEEE transactions on pattern analysis and machine intelligence **40**(5), 1259–1272.

Chu, C., Zhmoginov, A. & Sandler, M. (2017), 'CycleGAN, a master of steganography', arXiv preprint arXiv:1712.02950 .

Cieslewski, T., Choudhary, S. & Scaramuzza, D. (2018), Data-efficient decentralized visual slam, in '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2466–2473.

Collet, A., Martinez, M. & Srinivasa, S. S. (2011), 'The moped framework: Object recognition and pose estimation for manipulation', The international journal of robotics research **30**(10), 1284–1306.

Concha, A. & Civera, J. (2015), DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence, in 'IEEE/RSJ IROS'.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S. & Schiele, B. (2016), The cityscapes dataset for semantic urban scene understanding, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3213–3223.

Czarnowski, J., Laidlow, T., Clark, R. & Davison, A. J. (2020), 'Deepfactors: Real-time probabilistic dense monocular SLAM', IEEE Robotics and Automation Letters **5**(2), 721–728.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E. & Darrell, T. (2014), Decaf: A deep convolutional activation feature for generic visual recognition, in 'International conference on machine learning', pp. 647–655.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V. (2017), CARLA: An open urban driving simulator, in 'Proceedings of the 1st Annual Conference on Robot Learning', pp. 1–16.

Durrant-Whyte, H. & Bailey, T. (2006), 'Simultaneous localization and mapping: part i', IEEE robotics & automation magazine **13**(2), 99–110.

Dymczyk, M., Stumm, E., Nieto, J., Siegwart, R. & Gilitschenski, I. (2016), Will it last? learning stable features for long-term visual localization, in '2016 Fourth International Conference on 3D Vision (3DV)', IEEE, pp. 572–581.

Efros, A. A. & Freeman, W. T. (2001), Image quilting for texture synthesis and transfer, in 'Proceedings of the 28th annual conference on Computer graphics and interactive techniques', ACM, pp. 341–346.

Engel, J., Koltun, V. & Cremers, D. (2017), 'Direct sparse odometry', IEEE Transactions on Pattern Analysis and Machine Intelligence .

Engel, J., Schöps, T. & Cremers, D. (2014), LSD-SLAM: Large-scale direct monocular SLAM, in 'ECCV', Springer, pp. 834–849.

Forster, C., Carlone, L., Dellaert, F. & Scaramuzza, D. (2016), 'On-manifold preintegration for real-time visual–inertial odometry', IEEE Transactions on Robotics **33**(1), 1–21.

Forster, C., Pizzoli, M. & Scaramuzza, D. (2014), SVO: Fast semi-direct monocular visual odometry, in '2014 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 15–22.

Fridrich, J. & Kodovsky, J. (2012), 'Rich models for steganalysis of digital images', IEEE Transactions on Information Forensics and Security **7**(3), 868–882.

Gaidon, A., Wang, Q., Cabon, Y. & Vig, E. (2016), 'Virtual worlds as proxy for multi-object tracking analysis', Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition .

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

Gao, X., Wang, R., Demmel, N. & Cremers, D. (2018), LDSO: Direct sparse odometry with loop closure, in '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 2198–2204.

Gauthier, J. (2014), 'Conditional generative adversarial nets for convolutional face generation', Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester **2014**(5), 2.

Geiger, A., Lenz, P., Stiller, C. & Urtasun, R. (2013), 'Vision meets robotics: The KITTI dataset', IJRR **32**(11), 1231–1237.

Gerlach, N. L., Meijer, G. J., Kroon, D.-J., Bronkhorst, E. M., Bergé, S. J. & Maal, T. J. J. (2014), 'Evaluation of the potential of automatic segmentation of the mandibular canal using cone-beam computed tomography', British journal of oral and maxillofacial surgery **52**(9), 838–844.

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 580–587.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014), Generative adversarial nets, in 'Advances in neural information processing systems', pp. 2672–2680.

Graber, G., Pock, T. & Bischof, H. (2011), Online 3D reconstruction using convex optimization, in 'ICCV Workshops', IEEE.

Granados, M., Kim, K. I., Tompkin, J., Kautz, J. & Theobalt, C. (2012), Background inpainting for videos with dynamic objects and a free-moving camera, in 'European Conference on Computer Vision', Springer, pp. 682–695.

Guerrero, R., Qin, C., Oktay, O., Bowles, C., Chen, L., Joules, R., Wolz, R., Valdés-Hernández, M., Dickie, D., Wardlaw, J. et al. (2018), 'White matter hyperintensity and stroke lesion segmentation and differentiation using convolutional neural networks', NeuroImage: Clinical **17**, 918–934.

Guizilini, V. & Ramos, F. (2015), 'Online self-supervised learning for dynamic object segmentation', The International Journal of Robotics Research **34**(4-5), 559–581.

Hartmann, W., Havlena, M. & Schindler, K. (2014), Predicting matchability, in 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 9–16.

He, K., Gkioxari, G., Dollár, P. & Girshick, R. (2017), Mask R-CNN, in 'Proceedings of the IEEE international conference on computer vision', pp. 2961–2969.

He, K., Zhang, X., Ren, S. & Sun, J. (2016), Deep residual learning for image recognition, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 770–778.

Henein, M., Kennedy, G., Mahony, R. & Ila, V. (2018), 'Exploiting rigid body motion for slam in dynamic environments', environments **18**, 19.

Henein, M., Zhang, J., Mahony, R. & Ila, V. (2020), 'Dynamic SLAM: The Need For Speed', arXiv preprint arXiv:2002.08584 .

Hinton, G. E. & Salakhutdinov, R. R. (2006), 'Reducing the dimensionality of data with neural networks', science **313**(5786), 504–507.

Hosseinzadeh, M., Li, K., Latif, Y. & Reid, I. (2019), Real-time monocular object-model aware sparse SLAM, in '2019 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 7123–7129.

Hosseinzadeh, S., Shakeri, M. & Zhang, H. (2018), Fast shadow detection from a single image using a patched convolutional neural network, in '2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 3124–3129.

Huang, J., Yang, S., Mu, T.-J. & Hu, S.-M. (2020), 'Clustervo: Clustering moving instances and estimating visual odometry for self and surroundings', arXiv pp. arXiv–2003.

Huang, J., Yang, S., Zhao, Z., Lai, Y.-K. & Hu, S.-M. (2019), ClusterSLAM: A SLAM Backend for Simultaneous Rigid Body Clustering and Motion Estimation, in 'Proceedings of the IEEE International Conference on Computer Vision', pp. 5875–5884.

Iizuka, S., Simo-Serra, E. & Ishikawa, H. (2017), 'Globally and locally consistent image completion', ACM Transactions on Graphics (TOG) **36**(4), 107.

Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. (2017), Image-to-image translation with conditional adversarial networks, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1125–1134.

Jiao, J., Jiao, J., Mo, Y., Liu, W. & Deng, Z. (2019), 'Magicvo: An end-to-end hybrid cnn and bi-lstm method for monocular visual odometry', IEEE Access **7**, 94118–94127.

Judd, K. M. & Gammell, J. D. (2019), 'The Oxford multimotion dataset: Multiple SE (3) motions with ground truth', IEEE Robotics and Automation Letters **4**(2), 800–807.

Khoshelham, K. & Elberink, S. O. (2012), 'Accuracy and resolution of kinect depth data for indoor mapping applications', Sensors **12**(2), 1437–1454.

Kim, D.-H. & Kim, J.-H. (2016), 'Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment', IEEE T-RO .

Kingma, D. P. & Welling, M. (2013), 'Auto-encoding variational bayes', International Conference on Learning Representations .

Klein, G. & Murray, D. (2007), Parallel tracking and mapping for small AR workspaces, in 'ISMAR', pp. 225–234.

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), Imagenet classification with deep convolutional neural networks, in 'Advances in neural information processing systems', pp. 1097–1105.

Kundu, A., Krishna, K. M. & Jawahar, C. (2011), Realtime multibody visual SLAM with a smoothly moving monocular camera, in '2011 International Conference on Computer Vision', IEEE, pp. 2080–2087.

Lajoie, P.-Y., Hu, S., Beltrame, G. & Carlone, L. (2019), 'Modeling perceptual aliasing in slam via discrete–continuous graphical models', IEEE Robotics and Automation Letters **4**(2), 1232–1239.

Lamarca, J., Parashar, S., Bartoli, A. & Montiel, J. (2019), 'DefSLAM: Tracking and Mapping of Deforming Scenes from Monocular Sequences', arXiv preprint arXiv:1908.08918 .

Latif, Y., Cadena, C. & Neira, J. (2013), 'Robust loop closing over time for pose graph slam', The International Journal of Robotics Research **32**(14), 1611–1626.

Leutenegger, S., Chli, M. & Siegwart, R. Y. (2011), Brisk: Binary robust invariant scalable keypoints, in '2011 International conference on computer vision', Ieee, pp. 2548–2555.

Li, P., Qin, T. et al. (2018), Stereo vision-based semantic 3d object and ego-motion tracking for autonomous driving, in 'Proceedings of the European Conference on Computer Vision (ECCV)', pp. 646–661.

Li, S. & Lee, D. (2017), 'RGB-D SLAM in Dynamic Environments Using Static Point Weighting', IEEE RA-L **2**(4), 2263–2270.

Li, Y., Qi, H., Dai, J., Ji, X. & Wei, Y. (2017), Fully convolutional instance-aware semantic segmentation, in 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 2359–2367.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. & Zitnick, C. L. (2014), Microsoft coco: Common objects in context, in 'ECCV'.

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A. & Catanzaro, B. (2018), Image inpainting for irregular holes using partial convolutions, in 'The European Conference on Computer Vision (ECCV)'.

Lowe, D. G. (1999), Object recognition from local scale-invariant features, in 'Proceedings of the seventh IEEE international conference on computer vision', Vol. 2, Ieee, pp. 1150–1157.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J. J., Cox, D., Corke, P. & Milford, M. J. (2015), 'Visual place recognition: A survey', IEEE Transactions on Robotics **32**(1), 1–19.

Maddern, W., Pascoe, G., Linegar, C. & Newman, P. (2017), '1 Year, 1000km: The Oxford RobotCar Dataset', The International Journal of Robotics Research (IJRR) **36**(1), 3–15.
URL: *http://dx.doi.org/10.1177/0278364916679498*

Miller, I. & Campbell, M. (2007), Rao-blackwellized particle filtering for mapping dynamic environments, in 'Proceedings 2007 IEEE International Conference on Robotics and Automation', IEEE, pp. 3862–3869.

Mur-Artal, R., Montiel, J. M. M. & Tardos, J. D. (2015), 'ORB-SLAM: a versatile and accurate monocular SLAM system', IEEE transactions on robotics **31**(5), 1147–1163.

Mur-Artal, R. & Tardós, J. D. (2014), Fast relocalisation and loop closing in keyframe-based SLAM, in '2014 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 846–853.

Mur-Artal, R. & Tardós, J. D. (2017), 'ORB-SLAM2: An open-source slam system for monocular, stereo, and RGB-D cameras', IEEE T-RO .

Newcombe, R. A., Lovegrove, S. J. & Davison, A. J. (2011), DTAM: Dense tracking and mapping in real-time, in 'ICCV', IEEE.

Oquab, M., Bottou, L., Laptev, I. & Sivic, J. (2014), Learning and transferring mid-level image representations using convolutional neural networks, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1717–1724.

Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T. & Efros, A. A. (2016), Context encoders: Feature learning by inpainting, in 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 2536–2544.

Paz, L. M., Piniés, P., Tardós, J. D. & Neira, J. (2008), 'Large-scale 6-dof slam with stereo-in-hand', IEEE transactions on robotics **24**(5), 946–957.

Peng, S., Liu, Y., Huang, Q., Zhou, X. & Bao, H. (2019), Pvnet: Pixel-wise voting network for 6dof pose estimation, in 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 4561–4570.

Peris, M., Martull, S., Maki, A., Ohkawa, Y. & Fukui, K. (2012), Towards a simulation driven stereo vision system, in 'Pattern Recognition (ICPR), 2012 21st International Conference on', IEEE, pp. 1038–1042.

Pinheiro, P. O., Lin, T.-Y., Collobert, R. & Dollár, P. (2016), Learning to refine object segments, in 'European conference on computer vision', Springer, pp. 75–91.

Porav, H., Maddern, W. & Newman, P. (2018), 'Adversarial training for adverse conditions: Robust metric localisation using appearance transfer', IEEE International Conference on Robotics and Automation .

Ren, S., He, K., Girshick, R. & Sun, J. (2015), Faster r-cnn: Towards real-time object detection with region proposal networks, in 'Advances in neural information processing systems', pp. 91–99.

Ren, X. & Malik, J. (2003), Learning a classification model for segmentation, in 'nternational Conference on Computer Vision', IEEE, p. 10.

Riazuelo, L., Montano, L. & Montiel, J. M. M. (2017), 'Semantic visual SLAM in populated environments', ECMR .

Rogers, J. G., Trevor, A. J., Nieto-Granda, C. & Christensen, H. I. (2010), SLAM with expectation maximization for moveable object tracking, in '2010 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 2077–2082.

Romera, E., Alvarez, J. M., Bergasa, L. M. & Arroyo, R. (2017), 'ERFNet', https://github.com/Eromera/erfnet.

Romera, E., Alvarez, J. M., Bergasa, L. M. & Arroyo, R. (2018), 'ERFNet: Efficient Residual Factorized ConvNet for Real-Time Semantic Segmentation', IEEE Transactions on Intelligent Transportation Systems **19**(1), 263–272.

Ronneberger, O., Fischer, P. & Brox, T. (2015), U-net: Convolutional networks for biomedical image segmentation, in 'International Conference on Medical image computing and computer-assisted intervention', Springer, pp. 234–241.

Rosinol, A., Gupta, A., Abate, M., Shi, J. & Carlone, L. (2020), '3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans', arXiv preprint arXiv:2002.06289 .

Rosten, E. & Drummond, T. (2006), Machine learning for high-speed corner detection, in 'European conference on computer vision', Springer, pp. 430–443.

Rothganger, F., Lazebnik, S., Schmid, C. & Ponce, J. (2006), '3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints', International journal of computer vision **66**(3), 231–259.

Rublee, E., Rabaud, V., Konolige, K. & Bradski, G. (2011), ORB: An efficient alternative to SIFT or SURF, in 'Computer Vision (ICCV), 2011 IEEE international conference on', IEEE, pp. 2564–2571.

Rünz, M. & Agapito, L. (2017), Co-fusion: Real-time segmentation, tracking and fusion of multiple objects, in '2017 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 4471–4478.

Runz, M., Buffier, M. & Agapito, L. (2018), Maskfusion: Real-time recognition, tracking and reconstruction of multiple moving objects, in '2018 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)', IEEE, pp. 10–20.

Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. & Davison, A. J. (2013), Slam++: Simultaneous localisation and mapping at the level of objects, in 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 1352–1359.

Schönberger, J. L. & Frahm, J.-M. (2016), Structure-from-motion revisited, in 'Conference on Computer Vision and Pattern Recognition (CVPR)'.

Schönberger, J. L., Zheng, E., Frahm, J.-M. & Pollefeys, M. (2016), Pixelwise view selection for unstructured multi-view stereo, in 'European Conference on Computer Vision', Springer, pp. 501–518.

Schönberger, J. L., Zheng, E., Pollefeys, M. & Frahm, J.-M. (2016), Pixelwise view selection for unstructured multi-view stereo, in 'European Conference on Computer Vision (ECCV)'.

Schorghuber, M., Steininger, D., Cabon, Y., Humenberger, M. & Gelautz, M. (2019), SLAMANTIC-Leveraging Semantics to Improve VSLAM in Dynamic Environments, in 'Proceedings of the IEEE International Conference on Computer Vision Workshops', pp. 0–0.

Scona, R., Jaimez, M., Petillot, Y. R., Fallon, M. & Cremers, D. (2018), StaticFusion: Background reconstruction for dense RGB-D SLAM in dynamic environments, in '2018 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 1–9.

Skinner, J., Garg, S., Sünderhauf, N., Corke, P., Upcroft, B. & Milford, M. (2016), High-fidelity simulation for evaluating robotic vision performance, in 'Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on', IEEE, pp. 2737–2744.

Song, Y., Yang, C., Lin, Z. L., Li, H., Huang, Q. & Kuo, C.-C. J. (2017), 'Image inpainting using multi-scale feature image translation', CoRR **abs/1711.08590**.

Strasdat, H., Montiel, J. & Davison, A. J. (2010), 'Scale drift-aware large scale monocular SLAM', Robotics: Science and Systems VI **2**(3), 7.

Stühmer, J., Gumhold, S. & Cremers, D. (2010), Real-time dense geometry from a handheld camera, in 'Joint Pattern Recognition Symposium'.

Sturm, J., Engelhard, N., Endres, F., Burgard, W. & Cremers, D. (2012), A benchmark for the evaluation of RGB-D SLAM systems, in '2012 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 573–580.

Sun, Y., Liu, M. & Meng, M. Q.-H. (2017), 'Improving RGB-D SLAM in dynamic environments: A motion removal approach', RAS .

Sünderhauf, N. & Protzel, P. (2012), Switchable constraints for robust pose graph slam, in '2012 IEEE/RSJ International Conference on Intelligent Robots and Systems', IEEE, pp. 1879–1884.

Tan, W., Liu, H., Dong, Z., Zhang, G. & Bao, H. (2013), Robust monocular SLAM in dynamic environments, in 'ISMAR', pp. 209–218.

Telea, A. (2004), 'An image inpainting technique based on the fast marching method', Journal of graphics tools **9**(1), 23–34.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W. & Abbeel, P. (2017), Domain randomization for transferring deep neural networks from simulation to the real world, in 'Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on', IEEE, pp. 23–30.

Triggs, B., McLauchlan, P. F., Hartley, R. I. & Fitzgibbon, A. W. (1999), Bundle adjustment—a modern synthesis, in 'International workshop on vision algorithms', Springer, pp. 298–372.

Uittenbogaard, R., Sebastian, C., Vijverberg, J., Boom, B., Gavrila, D. M. et al. (2019), Privacy protection in street-view panoramas using depth and multi-view imagery, in 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 10581–10590.

Ulyanov, D., Vedaldi, A. & Lempitsky, V. (2018), Deep image prior, in 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition', pp. 9446–9454.

Wang, C.-C., Thorpe, C. & Thrun, S. (2003), Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas, in '2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)', Vol. 1, IEEE, pp. 842–849.

Wang, C.-C., Thorpe, C., Thrun, S., Hebert, M. & Durrant-Whyte, H. (2007), 'Simultaneous localization, mapping and moving object tracking', The International Journal of Robotics Research **26**(9), 889–916.

Wang, S., Clark, R., Wen, H. & Trigoni, N. (2017), Deepvo: Towards end-to-end visual odometry with deep recurrent convolutional neural networks, in '2017 IEEE International Conference on Robotics and Automation (ICRA)', IEEE, pp. 2043–2050.

Wang, Y. & Huang, S. (2014), Motion segmentation based robust RGB-D SLAM, in 'WCICA', IEEE, pp. 3122–3127.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P. et al. (2004), 'Image quality assessment: from error visibility to structural similarity', IEEE transactions on image processing **13**(4), 600–612.

Wangsiripitak, S. & Murray, D. W. (2009), Avoiding moving outliers in visual SLAM by tracking moving objects, in 'ICRA', IEEE, pp. 375–380.

Xiang, Y., Schmidt, T., Narayanan, V. & Fox, D. (2017), 'PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes', arXiv preprint arXiv:1711.00199 .

Xiao, L., Wang, J., Qiu, X., Rong, Z. & Zou, X. (2019), 'Dynamic-SLAM: Semantic monocular visual localization and mapping based on deep learning in dynamic environment', Robotics and Autonomous Systems **117**, 1–16.

Xu, B., Li, W., Tzoumanikas, D., Bloesch, M., Davison, A. & Leutenegger, S. (2019), Mid-fusion: Octree-based object-level multi-instance dynamic SLAM, in '2019 International Conference on Robotics and Automation (ICRA)', IEEE, pp. 5231–5237.

Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O. & Li, H. (2017), High-resolution image inpainting using multi-scale neural patch synthesis, in 'The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 1, p. 3.

Yang, H., Antonante, P., Tzoumas, V. & Carlone, L. (2020), 'Graduated non-convexity for robust spatial perception: From non-minimal solvers to global outlier rejection', IEEE Robotics and Automation Letters **5**(2), 1127–1134.

Yang, S., Maturana, D. & Scherer, S. (2016), Real-time 3D scene layout from a single image using convolutional neural networks, in '2016 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 2183–2189.

Yang, S. & Scherer, S. (2019), 'CubeSLAM: Monocular 3D object SLAM', IEEE Transactions on Robotics **35**(4), 925–938.

Yang, S., Song, Y., Kaess, M. & Scherer, S. (2016), Pop-up SLAM: Semantic monocular plane slam for low-texture environments, in '2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)', IEEE, pp. 1222–1229.

Yau, H.-T., Yang, T.-J. & Jian, H.-Z. (2013), 'A region-growing algorithm using parallel computing for surface reconstruction from unorganized points', Advances in Engineering Software **59**, 29–37.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X. & Huang, T. S. (2018), 'Generative Image Inpainting with Contextual Attention', Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition .

Zakharov, S., Shugurov, I. & Ilic, S. (2019), Dpod: 6d pose object detector and refiner, in 'Proceedings of the IEEE International Conference on Computer Vision', pp. 1941–1950.

Zhang, J., Henein, M., Mahony, R. & Ila, V. (2020), 'VDO-SLAM: A Visual Dynamic Object-aware SLAM System', arXiv preprint arXiv:2005.11052 .

Zhou, B., Lapedriza, A., Khosla, A., Oliva, A. & Torralba, A. (2017), 'Places: A 10 million image database for scene recognition', IEEE Transactions on Pattern Analysis and Machine Intelligence .

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A. & Oliva, A. (2014), Learning deep features for scene recognition using places database, in 'Advances in neural information processing systems', pp. 487–495.

Zhou, G., Bescos, B., Dymczyk, M., Pfeiffer, M., Neira, J. & Siegwart, R. (2018), 'Dynamic objects segmentation for visual localization in urban environments', arXiv preprint arXiv:1807.02996 .

Zhou, P., Han, X., Morariu, V. I. & Davis, L. S. (2018), 'Learning rich features for image manipulation detection', Proceedings of the IEEE conference on Computer Vision and Pattern Recognition .

Zhu, J.-Y., Park, T., Isola, P. & Efros, A. A. (2017), Unpaired image-to-image translation using cycle-consistent adversarial networks, in 'Computer Vision (ICCV), 2017 IEEE International Conference on'.

Figures

1.1 Sparse and dense maps comparison . . . 5
1.2 Examples of SLAM with deep priors . . . 8
1.3 Front end and back end in a SLAM pipeline . . . 8
1.4 Example of camera tracking failure . . . 10
1.5 Example of a corrupted map . . . 11
1.6 Example of challenging place recognition. . . . 12
1.7 *A priori* dynamic objects and moving objects. . . . 13

2.1 Multi-view geometry approach to detect movement. . . . 20
2.2 Precision-recall curves for the different descriptors. . . . 21
2.3 Optimal threshold selection. . . . 22
2.4 Detection and segmentation of dynamic objects. . . . 23
2.5 Semi-supervised segmentation results. . . . 26
2.6 Static $3D$ map built with dynamic input frames. . . . 27
2.7 Multi-object detection pipeline. . . . 28
2.8 Example of our camera trajectory accuracy. . . . 31

3.1 Inpainting examples . . . 39
3.2 Geometric inpainting visual results. . . . 41
3.3 Multi-object detection and inpainting pipeline. . . . 42
3.4 Generator architecture. . . . 43
3.5 Empty Cities pipeline. . . . 44
3.6 Noise detection results. . . . 45
3.7 Discriminator diagram. . . . 46
3.8 SRM kernels. . . . 47
3.9 FAST detector kernels. . . . 48
3.10 CARLA inpainting comparison. . . . 53
3.11 CityScapes inpainting results. . . . 54
3.12 Oxford Robotcar inpainting results. . . . 55
3.13 Qualitative dynamic-to-static and static-to-dynamic translation. . . . 57
3.14 Quantitative dynamic-to-static and static-to-dynamic translation. . . . 58
3.15 Baseline for VO experiments. . . . 60
3.16 VO influence of ORB loss. . . . 61
3.17 Qualitative results with the ORB loss. . . . 61
3.18 Influence of ORB loss on VO. . . . 62
3.19 VO experiments with real-world data. . . . 63
3.20 Place recognition comparison. . . . 65
3.21 Place recognition visual example. . . . 65
3.22 Dense maps qualitative comparison. . . . 67

3.23 Map with inpainted images. 69

4.1 Qualitative multi-object tracking results. 74
4.2 Notation to model dynamic objects and points. 77
4.3 Parameters saving ratio. 79
4.4 BA factor graph 82
4.5 Hessian boolen matrix. 83
4.6 Examples of camera and bounding box configurations. 84
4.7 Metrics for multi-object tracking. 87

Tables

2.1 Ablation study of DynaSLAM. 30
2.2 Comparison against the SLAM baseline. 31
2.3 Comparison against the state of the art. 32
2.4 Monocular results in TUM dataset. 32
2.5 Stereo results in KITTI dataset. 33
2.6 Monocular results in KITTI dataset. 33
2.7 Timing Analysis of DynaSLAM. 34

3.1 Ablation study of DynaSLAM with background inpainting. 42
3.2 Ablation study of Empty Cities. 52
3.3 Comparison against the state-of-the-art inpainting baseline. 54
3.4 Dense maps quantitative comparison. 68

4.1 Egomotion comparison on the KITTI raw dataset 84
4.2 Egomotion comparison on the KITTI tracking dataset. 85
4.3 Egomotion comparison on the KITTI odometry dataset. 86
4.4 Objects motion comparison on the KITTI tracking dataset 88
4.5 MOTP evaluation on the KITTI tracking dataset. 89
4.6 DynaSLAM II average computational time. 89
4.7 DynaSLAM II timing comparison. 89