Departamento de
Informática e Ingeniería
de Sistemas
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Tesis de Fin de Máster
Máster en Ingeniería de Sistemas e Informática
Curso 2010/2011

Dense Mapping for Indoor Environments

Yasir Latif

Directores: Jose Neira, Pedro Piniés

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Noviembre de 2011

Abstract

This thesis deals with the problem of indoor environment modelling using depth cameras. We propose a system that allows to traverse an environment with a hand held camera, and no other sensor, and compute a dense 3D textured geometric reconstruction. In the front-end, camera motion is computed by detecting interest points in the images and matching them across frames. We propose a loop closing or place recognition algorithm that is robust over time, and thus allows the system to reconsider past loop closing decisions once additional information becomes available. The backend of the system is the g2o graph optimization algorithm. We test our system both in simulations and with real data coming from a Kinect sensor. Results show that thus system is viable and precise. Our future goal is to incorporate robust outlier detection algorithms that will allow the system to ignore dynamic objects, such as people, and avoid the inclusion of these elements in the model.

Contents

1 Introduction **2**
1.1 Problem Statement 2
1.2 State of the Art 3

2 The Proposed System **5**
2.1 Visual Odometry 5
2.1.1 KeyFrames 7
2.2 Loop Closing over Time 8
2.2.1 Hypothesis generation 8
2.2.2 Hypothesis verification 9
2.3 Global Pose Refinement 12

3 Simulations **13**
3.1 Simulations 13

4 Conclusions and Future Work **19**

Chapter 1

Introduction

1.1 Problem Statement

We live in a colorful and richly textured world. Our senses have evolved to make decisions about localization, navigation and recognition based on visual information perceived from our surroundings. Creating maps of the world for autonomous vehicles to enable them to make similar decisions has been an on going effort in the robotics community, where this problem is known as Simultaneous Localization and Mapping or SLAM for short. Efforts have been going on to create maps that make sense to robots and are therefore based on sparse features extracted from the environment such as laser scans or monocular cameras (Davison et al., 2007; Klein and Murray, 2009). On the other hand, techniques have been developed to create dense models of the environment (Newcombe et al., 2011; Henry et al., 2010). This project is an attempt at the second category.

Building textured models of the world is an expensive task to undertake. While very precise 3D laser scanners are available, they are out of the reach of a normal person because of their cost. With the recent introduction of consumer grade depth cameras such as Microsoft Kinect (Microsoft, 2010), the task of building accurate 3D models is being explored by the robotics community both because of the affordable cost and the wide availability of the sensor.

Building models of indoor environment presents a number of challenges. While nature is rich in texture, indoor environments tend to be planar and less textured. Moreover, due to artificial lighting, the illumination is not consistent even over a small distance. This makes local feature based algorithms very unstable. Visual similarity of structures makes the identification of previously visited places, known as *loop closing*, a difficult problem. A typical

case to consider is the hallways in a multi-storey building, which all may look the same. Only using visual information for loop closing can then lead to wrong loop closures that can adversely effect the accuracy of the model.

1.2 State of the Art

Simultaneous Localization And Mapping (SLAM) is a well studied problem in the robotics community. Most of the initial algorithms were propose for a robot moving in a plane. More recently there has been a shift in the community towards vision based approaches and very robust system have been developed using monocular cameras as sensors. (Davison et al., 2007) developed a real-time SLAM system using a single camera. (Klein and Murray, 2009) developed a tracking and mapping system which works in real time but is limited to small work spaces. (Newcombe and Davison, 2010) proposed a dense mapping system using a single camera but it requires consistent illumination.

Recently, a few systems have employed depth cameras for creating dense models. (Henry et al., 2010) proposed the first work utilizing such cameras. (Du et al., 2011) developed a mapping system that ensures coverage and robustness but requires user interaction. (Newcombe et al., 2011) proposed a dense mapping algorithm which works in real time using information from a depth camera. Very recently, (Izadi et al., 2011) demonstrated a real-time dense mapping system that takes into account the dynamic nature of the environment being mapped. One important aspect of SLAM is data association, the process to determine which features in the map correspond to each sensor measurement. For data association in general, (Neira and Tardós, 2001) presented an algorithm that considers all the hypothesis jointly using an interpretation tree. (Civera et al., 2010) presented a hypothesis and test algorithm using one point in the dataset within a Kalman Filter framework. For loop closing in particular, (Olson, 2009) finds a subset of mutually consistent hypothesis from a set of proposed hypothesis using spectral clustering and (Sunderhauf and Protzel, 2011) detect and reject wrong loop closure using switch factors within a Pose Graph formulation while (Cadena et al., 2010) use Conditional Random Fields.

In this work, we propose a system that can robustly model the 3D textured environment with the help of a customer grade depth camera. This work focuses on the problem of robust loop closing over time. Loop closing is an important module of any mapping system and helps in improving the accuracy of the map. We propose a method of selecting loop closure hypotheses that are consistent over time and therefore will get less "confused" in a

scenario stated above. The application of such a system is not restricted to efficient robotic navigation but can also be used to create models for object recognition and for making models of the environment for use in games and virtual worlds.

Chapter 2

The Proposed System

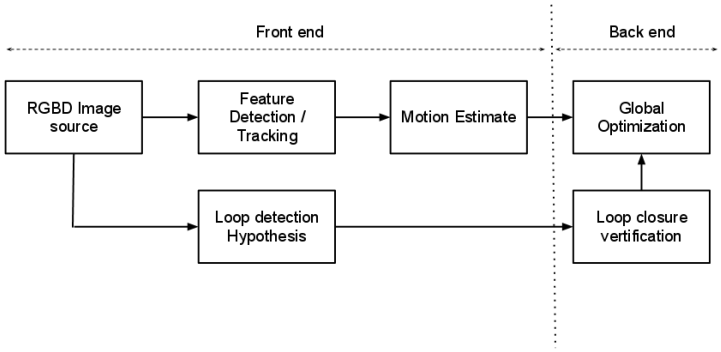
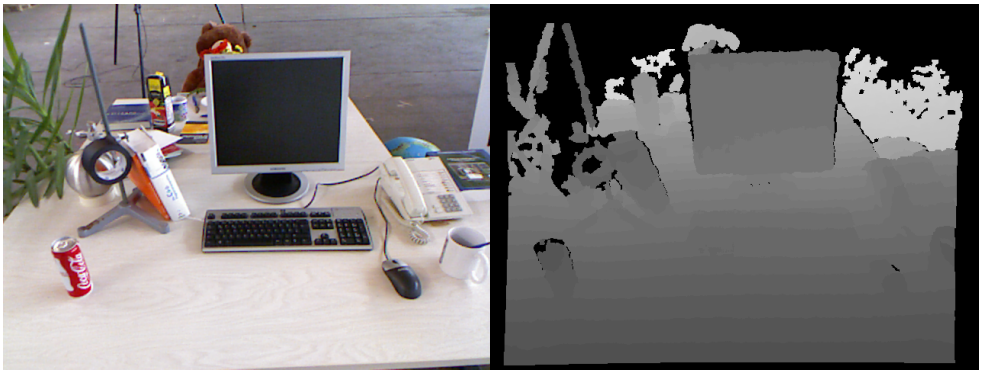


Figure 2.1: System Overview

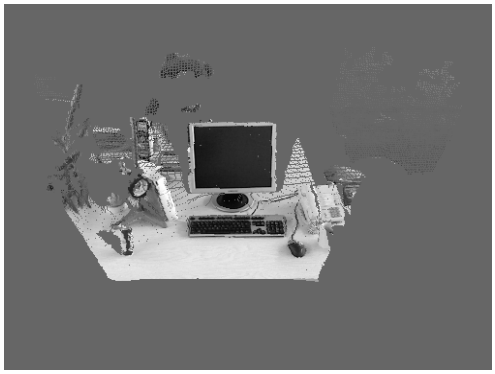
Fig. 2.1 gives an overview of the complete system. As can be seen, we make a distinction between the front end and the back end of the system. The front end is responsible for providing the Visual Odometry and loop closure hypothesis. The back end selects loop closure hypothesis that are consistent with each other and incorporates them into the global pose refinement problem. In the following, we present a detailed account of the involved modules.

2.1 Visual Odometry

Given motion commands, mobile robots provide an estimate of the actual motion carried out which is know as odometry. The estimation of motion with a visual sensor such as a camera is therefore termed visual odometry



(a) RGB Image (b) Depth Image



(c) Resulting point cloud

Figure 2.2: Data received from Kinect

(Nistér et al., 2004). For a robot restricted to a plane, the odometry consists of three degree of freedom (x, y, θ), while a handheld camera, has six degrees of freedom; three for rotation and three for translation.

In monocular vision, visual odometry is a difficult problem to solve because of the scale ambiguity problem. A certain baseline needs to be traversed before we can calculate the depth of observed points for calculating the 3D rigid body transformation of the sensor. In case of depth cameras, this problem is readily solved because of the estimate of depth provided by the camera. The problem reduces to finding matching features between the two images, calculating the corresponding point in 3D and estimating the 3D rigid body transformation that reduces the reprojection error in the least square sense. This requires that the camera being used be calibrated. This work assumes that the camera being used to carry out VO is already calibrated.

The information provided by Microsoft Kinect consists of two images,

one RGB and the other depth image as shown in fig. 2.2. The depth image contains the depth per pixel of the corresponding RGB image. Therefore, knowing the calibration parameters of the camera and the point in the RGB image, we can calculate the 3D location of a point. Given a point $p_i = [x_i, y_i, 1]^T$ in an image and the calibration parameters f_x, f_y, c_x, c_y where the depth of the point is d_i, the corresponding 3D point P_i can be calculated using:

$$P_i = d_i \begin{bmatrix} 1/f_x & & -c_x/f_x \\ & 1/f_y & -c_y/f_y \\ & & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \tag{2.1}$$

Given at least three such points in two images, the 3d rigid body transformation can be calculated between the two images. Let $P_{1,i}$ be the point in the first image and $P_{2,i}$ be the corresponding point in the second image, with $i = 1, 2, \ldots, n$ and $n \geq 3$. To calculate the rigid body transformation, we follow a hypothesis and test method based on (Horn, 1987). As a result we get a transformation $T_2^1 = (R, t)$ consisting of rotation and translation such that $P_{2,i} \approx RP_{1,i} + t$. This rigid body transformation T is the estimated motion between the two frames.

For two consecutive such motion, T_2^1 and T_3^2 the overall pose of frame 3 is then given by $T_3^1 = T_2^1 \oplus T_3^2$. We assume the first frame to be the origin of the world and maintain all the poses with respect to the first frame. So the pose for any frame f is given by T_f^w where w is the origin of the world.

2.1.1 KeyFrames

Although visual odometry calculates the poses between every consecutive frames, we do not store all the frames, instead we take an approach commonly used in Keyframe based approach (Klein and Murray, 2008). We identify and store keyframes, i.e. intermediate frames from the image sequence where the number of tracked features fall below a certain threshold. When the system starts, we initialize new features in the image and track them in the subsequent images. As the camera moves along the trajectory, the number of tracked features decreases due to the change in the scene. When they fall below a threshold, this indicates the we have moved sufficiently far from our initial position and should now include this new information in the system. Therefore we initialize a new keyframe and introduce new feature into the system for tracking.

The pose of this new keyframe is provided by the underlying visual odometry system. We also match features from this keyframe with the previous keyframe in order to use them in the global pose refinement step.

2.2 Loop Closing over Time

Uncertainty in the model grows during an exploratory trajectory. This uncertainty is reduced when the sensor enters some part of the environment that has been observed before. This is called loop closing and helps in correcting the map. Our system deals with loop closure in two steps. The first step is detecting whether the current place is a reobservation of something we have seen before. We call this loop closure hypothesis generation and is done by the front end of the system. The second step consist of verifying the loop closure hypothesis and including them in the global optimization and is carried out by the back end.

2.2.1 Hypothesis generation

The main problem to consider in this step is to be able to identify whether we have visited the place we are in before. Since we are mainly dealing with images, we would like to be able to detect loop closure based on the visual information that we receive from the camera. While certain methods achieve this through learning either online or before-hand a bag-of-words representation of the environment (Angeli et al., 2008), our system follows a simpler method which can generate hypothesis for loop closure based on matching a single or a few feature descriptors.

The basic idea is derived from (Sunderhauf and Protzel, 2011) in which each keyframe is described by as little as a single BRIEF descriptor (Calonder et al., 2010). Instead of BRIEF descriptor, our system uses BRISK (Leutenegger et al., 2011) which gives rotation and scale invariance while being computationally as expensive as BRIEF. For each keyframe, the corresponding image is downsampled to a size comparable to the BRISK descriptor patch size and the descriptor is calculated at the center of this patch. This then represents the whole keyframe with just a single descriptor. Another approach is to divide the image into $m \times m$ such patches and calcuate $\mathbf{m^2}$ descriptors. In (Sunderhauf and Protzel, 2011), the reported times for detection, calculation and matching of a single descriptor are 1ms, 0.05ms and 0.0001ms respectively.

When a new keyframe is created, the necessary descriptors are calculated for this keyframe and are matched against all the previous keyframes in the system. A loop closing hypothesis is generated when the distance between the two descriptor sets is less that a certain predefined threshold. This threshold is a tunable parameter of the system and can be set to control the number of hypothesis generated. Features are matched between the two keyframes and an estimation of the transform between them is calculated. This transfor-

mation is the required loop closure hypothesis that relates one camera pose to another. Since all of these hypothesis may not be correct, we describe a method of hypothesis verification in the next section.

2.2.2 Hypothesis verification

In traditional systems, loop closing is a very conservative business. Loops are closed when the system is (almost) absolutely sure about the loop closing, either using temporal consistency or geometry-based checks. Once a loop is closed, it is never reconsidered. Even with this much amount of care, wrong loops may be closed due to the inherent repeatability in man-made structures. This is specially a problem in indoor environments where, for example, all the floors on a certain building might look the same. Both for a robot moving in a 2D world, and a hand held camera with six degrees of freedom such scenarios might happen very often.

For this reason, there is a need for a loop closure verificiation system that can take into account not just the information available at the present moment, but also hypothesis that have been considered in the past, and take meaningful decision on them for correctly closing the loop. This would also enable us to undo loops in the past that might have been incorrectly closed. In effect, we are ensuring the consistency of loop closure hypothesis over time.

Given a set of loop closure hypothesis, we try to find a subset of hypothesis that are mutually consistent with each other. Consider a set of hypotheses $H = [h_1, h_2, \ldots, h_N]$ generated by the front end, where each $\mathrm{h_i}$ relates some keyframe to some other keyframe with a transformation $\mathrm{T_i}$. The first step is to determine which hypotheses are individually compatible with each other.

For this purpose, consider a loop closing hypothesis generated by the front end from node j and i denoted by L_i^j. We assume that j is greater than i, because loops can only be closed with already observed places. Then the transformation from node i to j ($L_j^i = \ominus L_i^j$) should equal $T_{i+1}^i \oplus T_{i+2}^{i+1} \oplus \ldots T_j^{j-1}$, if there is no noise in the system. We can also state that

$$L_i^j \oplus (T_{i+1}^i \oplus T_{i+2}^{i+1} \oplus \ldots T_j^{j-1}) = 0 \tag{2.2}$$

or equivalently,

$$L_j^i = (T_{i+1}^i \oplus T_{i+2}^{i+1} \oplus \ldots T_j^{j-1}) \tag{2.3}$$

The system is not perfect and we do not get the identity transform because of noise. Therefore, we need to have a measure of how far the transformations are from each other.We can calculate this by using the Mahalanobis distance.

Let $h_i = x_1 \oplus x_2 \oplus \ldots \oplus x_n$ denote the transformation obtained from pose composition in the above equation for the i^{th} loop closing, where each term in the composition has an associated uncertainity, denoted by P_i for the i^{th} transformation, and the uncertainity associated with the j^{th} hypothesis is denoted by R_j. The Jacobian Matrix of h_i is given by

$$\mathbf{H_i} = \left[\left.\frac{\partial h}{\partial x_1}\right|_{\hat{x_i}} \left.\frac{\partial h}{\partial x_2}\right|_{\hat{x_i}} \cdots \left.\frac{\partial h}{\partial x_n}\right|_{\hat{x_i}} \right]$$

where each term in the Jacobian is calcualted using

$$\left.\frac{\partial \mathbf{h}}{\partial \mathbf{x_k}}\right|_{\hat{\mathbf{x}}_\mathbf{i}} = \left.\frac{\partial \mathbf{h}}{\partial(\mathbf{x_1} \oplus \mathbf{x_2} \oplus \ldots \oplus \mathbf{x_k})}\right|_{\hat{\mathbf{x}}_\mathbf{i}} \cdot \left.\frac{\partial(\mathbf{x_1} \oplus \mathbf{x_2} \oplus \ldots \oplus \mathbf{x_n})}{\partial \mathbf{x_k}}\right|_{\hat{\mathbf{x}}_\mathbf{i}}$$

The covariance Matrix can then be calculated using

$$S_{H_i} = H_i P_i H_i^T + R_j$$

and finally, the Mahalanobis distance is given by

$$D^2 = (L_j - h_i)^T S_{H_i}^{-1} (L_j - h_i) \leq \chi^2_{d,1-\alpha}$$

which should be less that $\chi^2_{2,1-\alpha}$ in order to consider the loop closing hypothesis L_j compatible with transformation h_i. This is shown in algorithm 1 for **n** loop closing hypothesis and **m** transformations.

Algorithm 1 Individual Compatibility

```
IC ← []
for i = 1 → n do
  for j = 1 → m do
    S_Hj = H_j P_j H_j^T + R_i
    D^2 = (L_j − h_i)^T S_Hj^{-1} (L_j − h_i)
    IC_ij = D^2 ≤ χ^2_{d,1−α}
  end for
end for
```

Having established which loop closures are individually compatible, the next step is to find a subset of hypotheses that support each other. We do this by taking into account all the hypothesis that are jointly compatible with each other. We search recursively through an interpretation tree using a branch and bound traversal. The algorithm for doing this is given in 2.

Our method gives us the largest subset of hypotheses that are jointly consistent. The evalution of our algorithm is presented in the results section.

Algorithm 2 Maximally supported subset

$MaximalSupport(L_{1...m}, H_{1...n})$:
Best = []
$RecursiveSearch(Best, 1)$
return Best
$RecursiveSearch(S, i)$: {find support for loop closure L_i}
if $i > m$ **then**
  if pairings(S) > pairings(Best) **then**
    Best ←S
  end if
else
  for $i = 1 \rightarrow n$ **do**
    if IC_{ij} **and** jointly_compatible(S,i,j) **then**
      $RecursiveSearch([S\ j], i + 1)$ {Add to the current subset}
    end if
  end for
  if $pairings(H) + m - i > pairings(Best)$ **then**
    $RecursiveSearch([S, 0], i + 1)$ {Can do better?}
  end if
end if

2.3 Global Pose Refinement

Visual odometry provides an estimate of the camera pose with respect to the environment but this estimate suffers from drift over time. In order to get a better estimate, global pose refinement needs to be done which can consider all the information of the poses and the environment features together in order to calculate a better approximation of the structure of the environment as well as poses of the cameras.

The classical approach to global pose refinement is termed as bundle adjustment (Triggs et al., 2000). The method involves the simultaneous computation of camera poses and the points in 3D space given an initial estimate of the two. In our system, the initial estimation is provided by visual odometry. In order to solve this bundle adjustment problem we use g2o (Kummerle et al., 2011) as the back end.

General Graph Optimization or g2o addresses the class of nonlinear least square minimization problems that can be represented as a graph. These include many problems including SLAM and bundle adjustment. G2o exploits the structure of underlying problems to efficiently compute the solution. The general formulation of such problems is given by

$$\mathbf{F}(\mathbf{x}) = \sum_{\langle \mathbf{i},\mathbf{j} \rangle \in \mathbf{C}} \mathbf{e}(\mathbf{x_i}, \mathbf{x_j}, \mathbf{z_{ij}})^{\mathbf{T}} \mathbf{\Omega_{ij}} \mathbf{e}(\mathbf{x_i}, \mathbf{x_j}, \mathbf{z_{ij}})$$

$$\mathbf{x}^{\star} = \arg\min_{\mathbf{x}} \mathbf{F}(\mathbf{x})$$

Where, $x = [x_1^T, x_2^T, \ldots, x_n^T]^T$ is a vector of parameters, z_{ij} represents the mean of the constraint relating x_i to x_j with Ω_{ij} the related information matrix. e(.) is an error function that meansures how well the parameters blocks x_i and x_j satisfy the constraint z_{ij}.

We will use global pose refinement to compute the final sequence of poses once all loop closings have been decided.

Chapter 3

Simulations

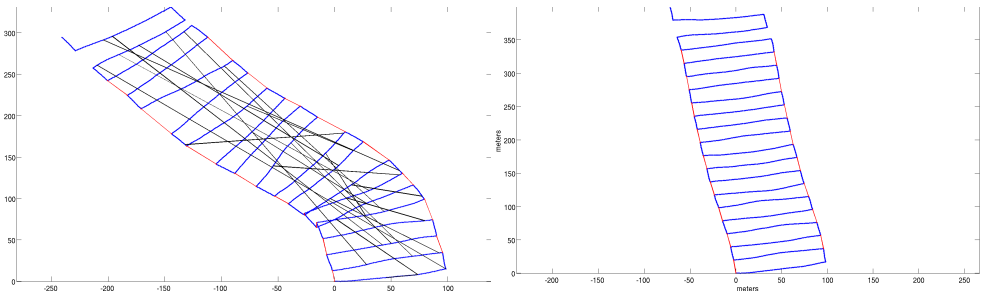
In this chapter we first present a set of simulations that demostrate the funtionality of loop closing verification module. Then we present experimental results for the dense mapping system.

3.1 Simulations

In man made environments, perceptual aliasing can give rise to wrong loop closures and only one such wrong loop closure can lead to a wrong solution in the underlying optimization problem. A representative case is show in fig. 3.1 in which a robot moving through a corridor can very easily get 'confused' and suggest wrong loop closures.

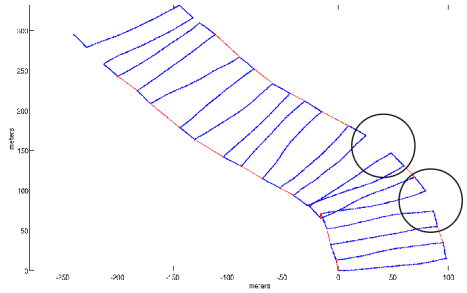
We have simulated two scenarios that are representative of SLAM trajectories. Both simulate the so called 'lawn mower' trajectory as show in figs. 3.2 and 3.3. The trajectory is corrupted with noise. We also simulate a set of correct and wrong loop closure hypotheses. Each simulation is run 100

Figure 3.1: A example of perceptual aliasing

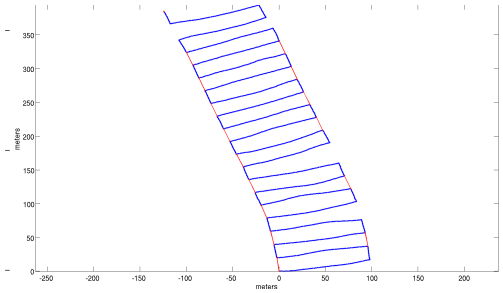


(a) All Hypothesis

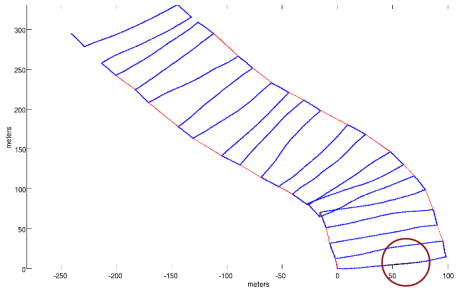
(b) Correct Hypothesis after optimization



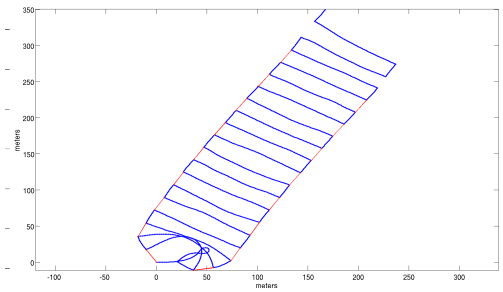
(c) Proposed Method



(d) Proposed Method after optimization

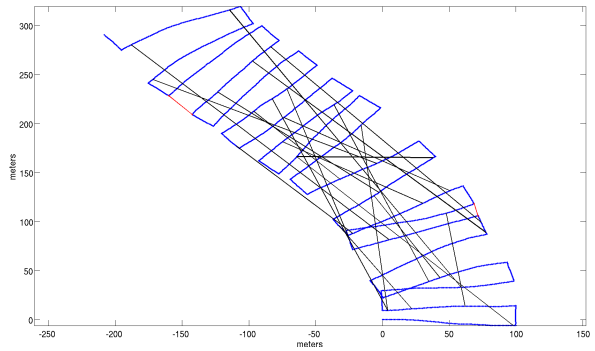


(e) Olson's Method

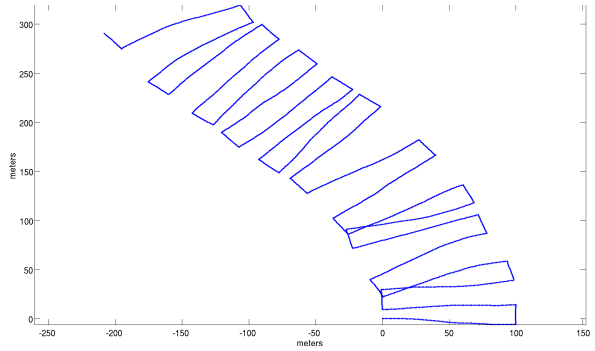


(f) Olson's Method after optimization

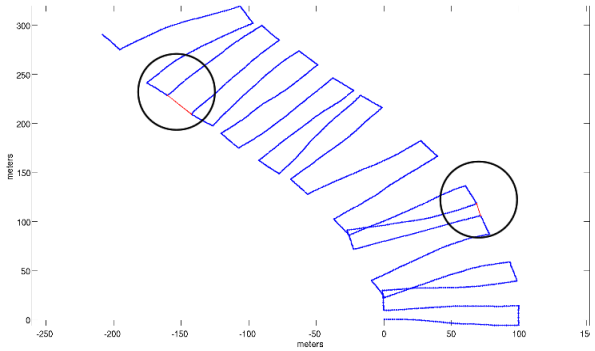
Figure 3.2: Lawn Mower Simulation - 50-50



(a) All Hypothesis

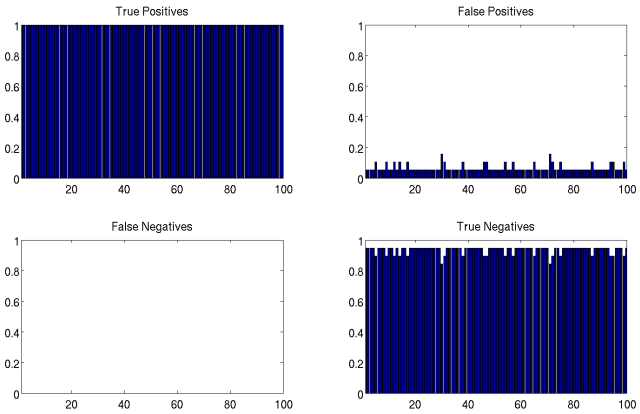


(b) Olson's Method

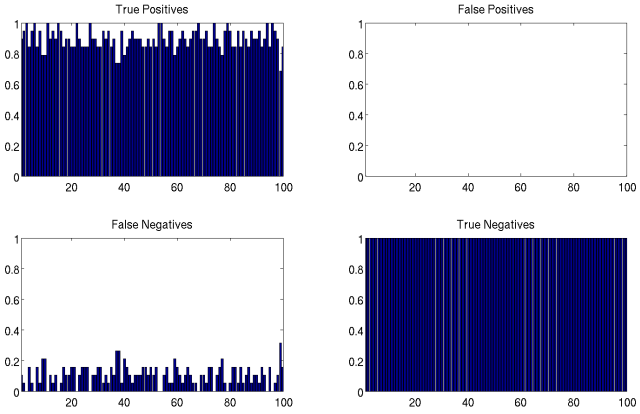


(c) Proposed Method

Figure 3.3: Lawn Mower Simulation - Needle in a hay stack

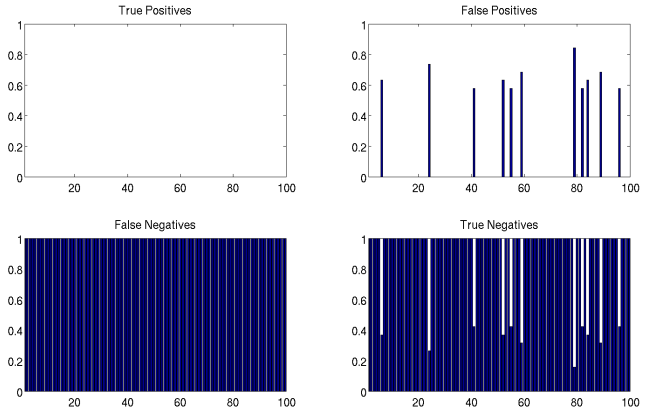


(a) Olson's Method

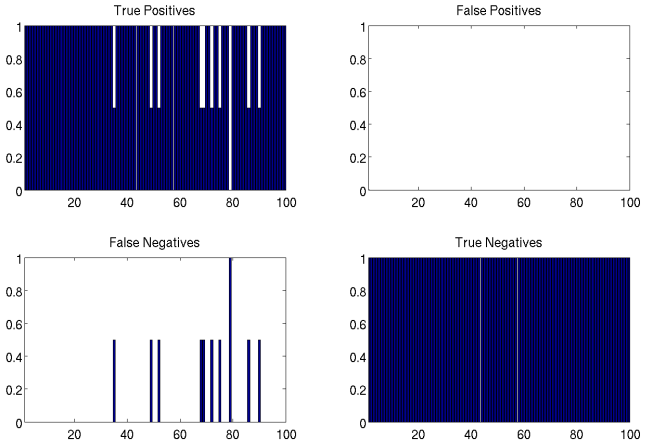


(b) Proposed Method

Figure 3.4: Lawn Mower Simulation - 50-50 Statistics



(a) Olson's Method



(b) Proposed Method

Figure 3.5: Lawn Mower Simulation - Needle in a hay stack Statistics

times with the same loop closure hypotheses to investigate the effect of noise on our method.

In the first simulation, we generate the same number of correct and wrong loop closures. The simulated trajectory along with the generated loop closing hypothesis are shown in fig. 3.2a. Our method correctly identifies and rejects all the wrong loop closure hypothesis while (Olson, 2009) shows a tendency towards accepting false positives. Although it accepts all the correct hypotheses, at the same time, it accepts many false positives (fig. 3.2e. It has to be noted that in the case of a loop closing, a hypotheses wrongly accepted as being correct (false positive) is much more harmful than a correct hypotheses classified as being wrong (false negative). In that respect, our method is more robust to false positive, although we do reject some correct hypothesis 3.2c. The statistics for this scenario are given in fig. 3.4. All the plots represent ratios of the hypotheses accepted or rejected.

In the second simulation (fig. 3.3), we simulate a very noisy front end hypothesis generator, that produces a lot of wrong hypotheses but very few correct hypotheses. We term this finding a needle in the hay stack. This would give us an idea of the performance of our algorithm in situations where only a few hypotheses are correct and drowned by a large number of wrong hypotheses. In this case as well, our method accepts no false positives and manages to find the few correct hypotheses most of the time (fig. 3.3c). Olson's method either rejects all the hypotheses or accepts wrong hypotheses as being correct. Our method is able to correctly identify the two loop closures that are consistent with each other (encircled in fig. 3.3c). The statistics for the scenario are given in 3.5.

These experiments show the superiority of our method in terms of robustness to false positives. Moreover, our method is able to discover the smallest possible consistent subset in a noisy set of loop closing hypothesis.

Chapter 4

Conclusions and Future Work

We have presented a framework for mapping indoor environments using a depth camera and a state of the art graph optimization method as the back-end with an effective loop closure verification system that can ensure the consistency of loop closures over time. We have also presented a comparison of our work with the start of the art loop closing over time method. Immediate future work is implementating the visual odometry system and a place detection algorithm to find loop closing candidates in the depth images and carry out an indoor experiment. We plan to submit the loop closing over time algorithm to the following conference:

Twenty-Sixth AAAI Conference on Artificial Intelligence
Special Track on Robotics
Toronto, Ontario, Canada, July 22-26, 2012

In the planned system, the dense model of the environment is maintained as a point cloud, which is both inefficent in terms of memory and in its present state, can not be used for any decision making. For very large maps there will be a need to represent this point cloud in a more meaningful and effiecent way. One direction to take would be to represent the point clouds as voxels (Henry et al., 2010), which is a more efficent than point clouds. Another approach would be convert the point cloud into a mesh model and therefore reducing the number of point to be maintained in the map by a great factor.

The current system makes the assumption that the environment being mapped is static and does not change over time. In future we would like to be able to deal with dynamic environment in the short term so the effects of people moving around or other dynamic agents can be detected and safely removed from the map. In the long term, we would like to detect the changes

the took place since the place was last seen. This could include removal or addition of objects in the scence. A typical case would be the constantly appearing or disappearing items on an office desk. Detecting such information would not only enable us to do better localization in a dynamic environment, but will also enable us to maintain accurate maps over long time.

Bibliography

Angeli, A., D. Filliat, S. Doncieux, and J. Meyer (2008). Fast and incremental method for loop-closure detection using bags of visual words. *Robotics, IEEE Transactions on 24*(5), 1027–1037.

Cadena, C., D. Gálvez-López, F. Ramos, J. Tardós, and J. Neira (2010). Robust place recognition with stereo cameras. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 5182–5189. IEEE.

Calonder, M., V. Lepetit, C. Strecha, and P. Fua (2010). Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, 778–792.

Civera, J., O. Grasa, A. Davison, and J. Montiel (2010). 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *Journal of Field Robotics 27*(5), 609–631.

Davison, A., I. Reid, N. Molton, and O. Stasse (2007). Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 29*(6), 1052–1067.

Du, H., P. Henry, X. Ren, M. Cheng, D. Goldman, S. Seitz, and D. Fox (2011). Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing*, pp. 75–84. ACM.

Henry, P., M. Krainin, E. Herbst, X. Ren, and D. Fox (2010). Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*.

Horn, B. (1987). Closed-form solution of absolute orientation using unit quaternions. *JOSA A 4*(4), 629–642.

Izadi, S., R. Newcombe, D. Kim, O. Hilliges, D. Molyneaux, S. Hodges, P. Kohli, J. Shotton, A. Davison, and A. Fitzgibbon (2011). Kinectfusion: real-time dynamic 3d surface reconstruction and interaction. In *ACM SIGGRAPH 2011 Talks*, pp. 23. ACM.

Klein, G. and D. Murray (2008). Improving the agility of keyframe-based slam. *Computer Vision–ECCV 2008*, 802–815.

Klein, G. and D. Murray (2009). Parallel tracking and mapping on a camera phone. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pp. 83–86. Ieee.

Kummerle, R., G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard (2011). g2o: A general framework for graph optimization. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3607–3613. IEEE.

Leutenegger, S., M. Chli, and R. Siegwart (2011, Nov.). Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), IEEE International Conference on*.

Microsoft (2010). Microsoft kinect. http://www.xbox.com/en-US/kinect.

Neira, J. and J. Tardós (2001). Data association in stochastic mapping using the joint compatibility test. *Robotics and Automation, IEEE Transactions on 17*(6), 890–897.

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

Newcombe, R., S. Lovegrove, and A. Davison (2011). Dtam: Dense tracking and mapping in real-time. In *Proc. of the Intl. Conf. on Computer Vision (ICCV), Barcelona, Spain*, Volume 1.

Nistér, D., O. Naroditsky, and J. Bergen (2004). Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, Volume 1, pp. I–652. IEEE.

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

Sunderhauf, N. and P. Protzel (2011, sept.). Brief-gist - closing the loop by simple means. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 1234 –1241.

Triggs, B., P. McLauchlan, R. Hartley, and A. Fitzgibbon (2000). Bundle adjustment a modern synthesis. *Vision algorithms: theory and practice*, 153–177.