University of Zurich
Department of Informatics

ROBOTICS & PERCEPTION GROUP

Hans Muster

RPG Thesis Template

Semester Thesis

Robotics and Perception Group
University of Zurich

Supervision

First Supervisor
Second Supervisor

January 2013

Contents

Abstract

Compress the introduction in a few key sentences. No more than half a page.

Nomenclature

Notation

$\mathbf{J}$	Jacobian
$\mathbf{H}$	Hessian
$\mathbf{T}_{WB}$	coordinate transformation from B to W
$\mathbf{R}_{WB}$	orientation of B with respect to W
${}_W\mathbf{t}_{WB}$	translation of B with respect to W, expressed in coordinate system W

Scalars are written in lower case letters (a), vectors in lower case bold letters ($\mathbf{a}$) and matrices in upper case bold letters ($\mathbf{A}$).

Acronyms and Abbreviations

RPG	Robotics and Perception Group
DoF	Degree of Freedom
IMU	Inertial Measurement Unit
MAV	Micro Aerial Vehicle
ROS	Robot Operating System

Chapter 1

Introduction

Real-time monocular Visual Odometry (VO) algorithms can be used to estimate the 6 DoF pose of a camera relative to its surroundings. This is attractive for applications such as mobile robotics (mostly aerial vehicles where not much power is available) and Augmented Reality (AR) because cameras are small and self-contained and therefore easy to attach to autonomous robots or AR displays. Further, they are cheap, and are now often pre-integrated into mobile computing devices such as PDAs, phones and laptops.

SVO (Semi-direct Bisual Odometry) [3] is a very fast VO algorithm able to run at more than 300 frames per second on a consumer laptop. It builds a map based on keyframes and salient points. Most monocular VO are feature-based where scale and rotation invariant descriptors (SIFT, SURF...) are extracted and matched in order to recover the motion from frame to frame while finally refining the pose with reprojection error minimization with the map. SVO uses a different approach by using direct methods. Instead of matching descriptors, it uses intensity gradient to minimize the error between patches around detected salient points over the frame to frame transformation. Finally, it uses Bundle Adjustment to align with the map and avoid or minimize derive.

The main problem with most existing monocular VO implementations (including SVO) is a lack of robustness. Rapid camera motions, occlusion, and motion blur (phenomena which are common in all but the most constrained experimental settings) can often cause tracking to fail. While this is inconvenient with any tracking system, tracking failure is particularly problematic for VO systems: not only is camera pose lost, but the estimated map could become corrupted as well.

This problem is accentuated during a fast agile maneuver (e.g., a flip) and so a good relocalization is important when these are intended to be performed. The envisaged relocalization scheme proceeds as follows:

- Ina training stage, the vehicle explores the environment where the relocalization is supposed to occur. During this stage, an appropriate representation of the scene is created.

- The vehicle executes an agile maneuver during which vision-based tracking is no feasible.
- During the actual relocalization phase, the 6 DoF pose in the built map must be estimated.

1.1 Related Work

Place Recognition

Klein and Murray presents in [5] the relocalization method used in PTAM [4]. PTAM is a VO algorithm based on keyframes that are used during the relocalization. The relocalization method consists of two steps. First, given the current frame, the most similar keyframe is retrieved, and its know pose is used as a baseline. As measure of similarity the difference between subsampled, blurred and zero-mean images is used. This measure is a cross correlation. The small blurry images are storied every time there is a new keyframe and the small blurry image of a new frame is computed during the relocalization to be compared with the keyframes.

Other methods can be used for image retrieval, for example using bag of words. Nistér and Stewenius [6] proposes to use a tree structure to store words in order to handle much larger vocabulary or have a much faster retrieval. Every node of the tree would have k child nodes which are the clustering results of k-means. The tree would be build by recursive k-means.

This structure is expensive to build because k-means is very resource consuming. During the online process, new words can be appended to the final leaves.

Özuysal et al. [7] proposes a simplified random forest classifier which relates image patches to objects. It is simplified because instead of using a tree structure they use a linear structure applying all the binary tests to the patch. The result of the tests is a binary descriptor, the list of binary tests is called Fern. Every object is trained with multiple random warps of the known view to introduce information from possible different views of the object. In the end every object can be represented with many binary descriptors and every descriptor should output a probability distribution of possible objects represented. Evaluating multiple Ferns and joining the yelled distributions the final classification is achieved.

Pose Estimation

During the second step of the relocalization of PTAM, the transformation from the retrieved frame is calculated. This transformation will be finally appended to the know keyframe pose. To do so, an image alignment algorithm, Efficient Second-Order Minimization method (ESM) [2], is employed. ESM is a Gauss-Newton gradient descent algorithm which can be used with different image warp functions, it is a Lucas-Kanade [1] algorithm that uses Second-order functions; therefore, results in a faster convergence.

Geometric methods are typically used to find the transformation from the fond keyframe using the classic pipeline of salient points detection, feature extraction and matching. The 5pt algorithm can then be used to find the 6 DoF transformation or the 3pt algorithm if depth is known.

Joint Place and Pose estimation

One approach to solve the relocalization problem was proposed by Williams [10]. In their implementation, they use Random Forest classifiers to characterize a salient object in space. To do so, the classifier needs to be trained with as many as possible representations of the object (multiple views). Therefore, the first time an object is found, multiple warps of the of the patch are used to initialize its presence in the classifier. On later encounters with the object, the classifier is incrementally trained with additional data. During the relocalization phase salient points are classified using the trained classifier and the 3pt algorithm is used to recover the 6 DoF position. This method is is memory expensive and requires of GPU power to generates the patch warps.

Shotton et al. [8] also propose a method using random forests. RGB-D data is used to train the classifier. In this case, all the information is encoded in the classifier so no previous data storing or computing (salient point detection, descriptor extraction, etc...) is needed. The classifier is trained to an individual RGB-D pixel, and an RGB-D pixel query will output a probability distribution over the position in $\mathbb{R}^3$. This can be applied to all pixels of a frame or to a sparse subset selection of them. Ideally, the camera pose can be inferred from only three pixels, but as the output of the classifier can be very noisy, a second step is applied. From the output from many pixels an energy function is minimized using preemptive RANSAC in order to find a pose that agrees with most of the distributions.

To train this method a very complete dataset of RGB-D images with 6 DoF poses from the environment associated to them is needed. That makes it difficult to be used with SLAM problems where the map get populated incrementally. An online training method should be developed.

Chapter 2

Approach

Two kind of approaches will be proposed. One one side, local approaches based on the PTAM implementation, where two steps are performed. The first step has been named *Place Recognition* and the second *Real Pose Recognition*. Multiple methods will be proposed to solve the second step. Then, on the other side, a global approach will be proposed. In this case machine learning methods (*ferns*) will be used to recognize points in space.

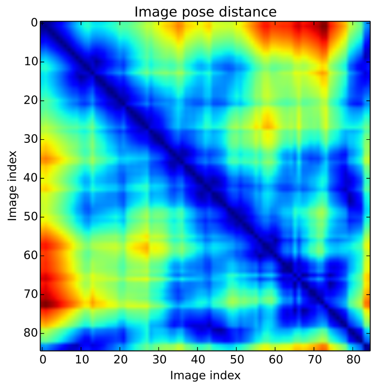
2.1 PTAM method

PTAM is a VO algorithm based on keyframes and so the relocalization method proposed is based on keyframes as well. Every keyframe has asociated with it a camera pose that will be used to relocalize. During the relocalization there are two steps involved. We have called the first step *Place Recognition* and the second *Real Pose Finder*.

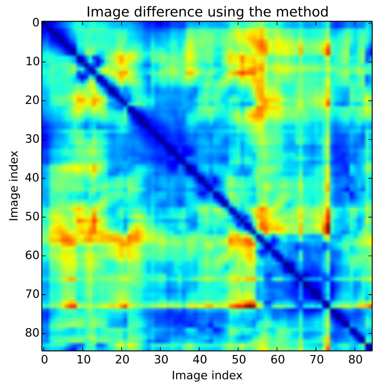
2.1.1 Place Recognition

During this step, the algorithm tries to find the keyframe image most similar to the last acquired image. The pose associated with the most similar keyframe is used as an initial rough estimation of the current pose. The similarity score should be resistant to view point because the new acquired image will, most probably, never be taken from the same pose as any of the keyframes. Also it should be fast to compute.

The used similarity score is the Cross Correlation between images meaning the sum of the squared error between two zero-mean images. To make to computation faster both images are resized become 40×30. Then, to make the images more resistant to view point changes it is blurred with a 3×3 Gaussian kernel with $\sigma = 2.5$. The resulting storied image is a resized, blurred and zero-mean



(a) Image to image real distance between the keyframe pose



(b) Image to image distance approximated using the Cross Correlation value

Figure 2.1

image called *small-blurry-image*.

During the normal map building pipeline this image is computed and stored every time a new keyframe is added to the map. And then, during the relocalization, the sum of squared difference between every stored *small-blurry-image* and the last acquired frame is computed to find the most similar keyframe and then use its pose as an initial estimation of the current camera pose.

Method evaluation

To evaluate the method the real distance between two frames is going to be compared with the Cross Correlation value described above. Far away frames should be dissimilar and close by frames should be more similar and have a lower CC value. In figure 2.1a can be seen the pair distances between image using the real pose while in figure 2.1b there is the approximated pair distances using the CC value as distance. It can be seen that they have a similar distribution. Also the correlation of 0.4337 shows that one explains the other in most cases.

Finally, to show that this method can be used, for every image the most CC similar image was taken being it real K closest image. Ideally the most CC similar image should always be the closest image. In figure 2.2 there is the count of occurrences of each K. It can be seen that most images resolve to the first or second closest image using this method.

Real Pose Finder

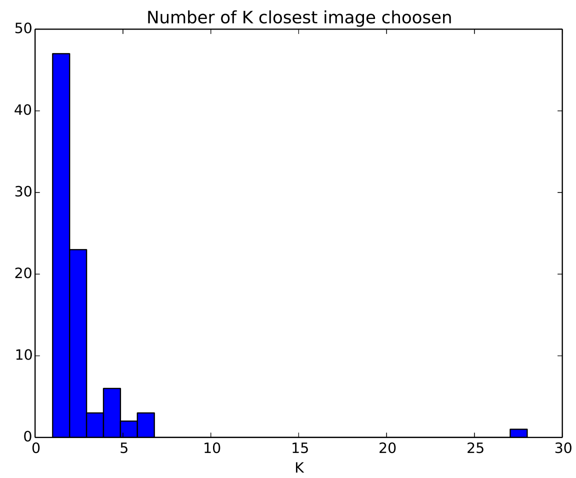

Number of K closest image choosen
50
40
30
20
10
0
0
5
10
15
20
25
30
K

Figure 2.2

Chapter 3

To be removed

Describe the main steps in your algorithm. An illustration is always helpful.

Here are some LaTeX tips:

3.1 Headings

Your report can be structured using several different types of headings. Use the commands `\chapter{.}`, `\section{.}`, `\subsection{.}`, and `\subsubsection{.}`. Use the asterisk symbol `*` to suppress numbering of a certain heading if necessary, for example, `\section*{.}`.

3.2 References

References to literature are included using the command `\cite{.}`. For example [4, 9]. Your references must be entered in the file `bibliography.bib`. Making changes or adding new references in the bibliography file can be done manually or by using specialized software such as *JabRef* which is free of charge.

Cross-referencing within the text is easily done using `\label{.}` and `\ref{.}`. For example, this paragraph is part of chapter 2; more specifically on page 7.

3.3 Writing Equations

The most common way to include equations is using the `equation` environment. Use `\eqref{.}` to reference an equation, e.g. (3.1).

$$\begin{aligned} C(\mathbf{x}) &= \frac{1}{2} \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}_i} \mathbf{e}_{i,k}(\mathbf{x})^T \ \mathbf{W}_{i,k} \ \mathbf{e}_{i,k}(\mathbf{x}) \\ \hat{\mathbf{x}}^{LS} &= \operatorname{argmin}_{\mathbf{x}} \ C(\mathbf{x}), \end{aligned} \tag{3.1}$$

$$\mathtt{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{p}_i \\ 0 & 1 \end{bmatrix} \qquad \text{with} \quad \mathbf{R}_i \in SO(3), \quad \mathbf{p} \in \mathbb{R}^3. \tag{3.2}$$

3.4 Including Graphics

The easiest way to include figures in your document is to use pdf figures if you use `pdflatex` to compile. Figure 3.1 was created with the use of the open source program `ipe`.

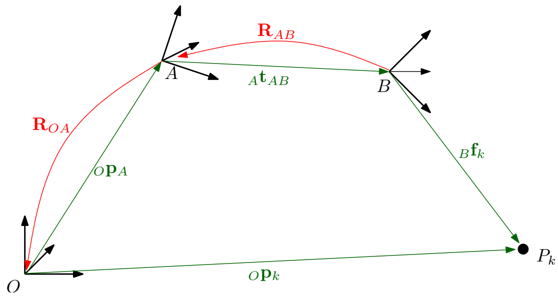


Figure 3.1: Example of a figure.

3.5 Including Code in your Document

You may include samples from your Matlab code using the `lstlistings` environment, for example

Listing 3.1: Matlab Example

```
% Evaluate y = 2x
for i = 1:length(x)

  y(i) = 2*x(i);

end
```

Listing 3.2: C++ Example

```
% sum all elements in a list
int sum=0;
for(list<int>::iterator it=mylist.begin(); it!=mylist.end(); ++it)
  sum += *it;
```

Chapter 4

Experiments

Provide numerical results, plots and timings. Interpret the data.

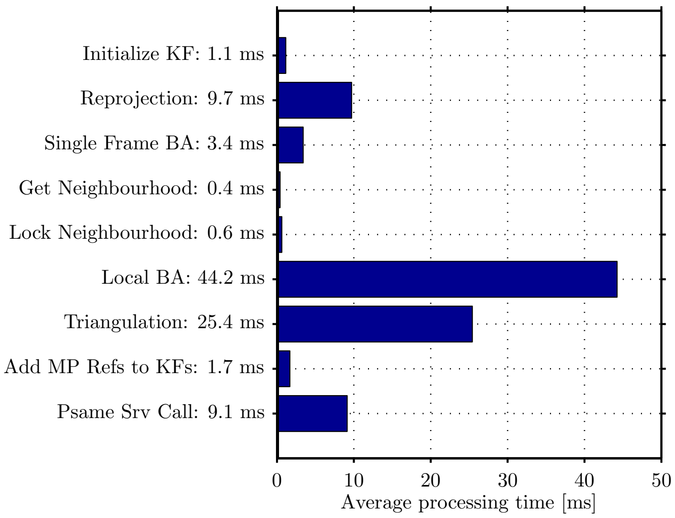


Figure 4.1: Example of a figure.

Chapter 5

Discussion

Explain both, the advantages and limitations of your approach.

5.1 Future Work

How would you extend the work? Can you propose another approach?

Appendix A

Something

In the appendix you can provide some more data, a tutorial on how to run your code, a detailed proof etc.

Bibliography

[1] Simon Baker and Iain Matthews. Lucas-Kanade 20 Years On: A Unifying Framework. *International Journal of Computer Vision*, 56(3):221–255, February 2004.

[2] S Benhimane and E Malis. Homography-based 2D visual servoing. *Robotics and Automation, 2006. ICRA ...*, 2006.

[3] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast Semi-Direct Monocular Visual Odometry. *Proc. IEEE Intl. Conf. on Robotics ...*, 2014.

[4] Georg Klein and David Murray. Parallel Tracking and Mapping for Small AR Workspaces. *IEEE and ACM International Symposium on Mixed and Augmented Reality*, November 2007.

[5] Georg Klein and David Murray. Improving the agility of keyframe-based SLAM. *Computer VisionECCV 2008*, 2008.

[6] D Nister and H Stewenius. Scalable recognition with a vocabulary tree. *... Vision and Pattern Recognition, 2006 ...*, 2006.

[7] M Ozuysal and M Calonder. Fast keypoint recognition using random ferns. *Pattern Analysis and ...*, 2010.

[8] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2930–2937, 2013.

[9] Hauke Strasdat, J M M Montiel, and Andrew J Davison. Real-time Monocular SLAM: Why Filter? *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, pages 2657 – 2664, 2010.

[10] Brian Williams, Georg Klein, and Ian Reid. Real-time SLAM relocalisation. *... Vision, 2007. ICCV 2007. IEEE 11th ...*, pages 1–8, 2007.

ROBOTICS & PERCEPTION GROUP

Title of work:

RPG Thesis Template

Thesis type and date:

Semester Thesis, January 2013

Supervision:

First Supervisor
Second Supervisor

Student:

Name:	Hans Muster
E-mail:	muster@student.ethz.ch
Legi-Nr.:	97-906-739

Statement regarding plagiarism:

By signing this statement, I affirm that I have read the information notice on plagiarism, independently produced this paper, and adhered to the general practice of source citation in this subject-area.

Information notice on plagiarism:

`http://www.ethz.ch/students/semester/plagiarism_s_en.pdf`

Zurich, 19. 5. 2014: ____