ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Learned Single-Shot Image-Based Camera Localization

Master Thesis

Moyuan Zhou

September 15, 2019

Advisors: Dr. Bugra Tekin, Dr. Johannes L. Schönberger
Supervisor: Prof. Marc Pollefeys
Department of Computer Science, ETH Zürich

Abstract

This example thesis briefly shows the main features of our thesis style, and how to use it for your purposes.

Contents

Chapter 1

Introduction

This is version `v1.4` of the template.

We assume that you found this template on our institute's website, so we do not repeat everything stated there. Consult the website again for pointers to further reading about LaTeX. This chapter only gives a brief overview of the files you are looking at.

1.1 Features

The rest of this document shows off a few features of the template files. Look at the source code to see which macros we used!

The template is divided into TeX files as follows:

1. `thesis.tex` is the main file.
2. `extrapackages.tex` holds extra package includes.
3. `layoutsetup.tex` defines the style used in this document.
4. `theoremsetup.tex` declares the theorem-like environments.
5. `macrosetup.tex` defines extra macros that you may find useful.
6. `introduction.tex` contains this text.
7. `sections.tex` is a quick demo of each sectioning level available.
8. `refs.bib` is an example bibliography file. You can use BibTeX to quote references. For example, read [**?**] if you can get a hold of it[4].

1.1.1 Extra package includes

The file `extrapackages.tex` lists some packages that usually come in handy. Simply have a look at the source code. We have added the following comments based on our experiences:

REC This package is recommended.

OPT This package is optional. It usually solves a specific problem in a clever way.

ADV This package is for the advanced user, but solves a problem frequent enough that we mention it. Consult the package's documentation.

As a small example, here is a reference to the Section *Features* typeset with the recommended *varioref* package:

> See Section 1.1 on the preceding page.

1.1.2 Layout setup

This defines the overall look of the document – for example, it changes the chapter and section heading appearance. We consider this a 'do not touch' area. Take a look at the excellent *Memoir* documentation before changing it.

In fact, take a look at the excellent *Memoir* documentation, full stop.

1.1.3 Theorem setup

This file defines a bunch of theorem-like environments.

Theorem 1.1 *An example theorem.*

Proof Proof text goes here. □

Note that the q.e.d. symbol moves to the correct place automatically if you end the proof with an `enumerate` or `displaymath`. You do not need to use `\qedhere` as with *amsthm*.

Theorem 1.2 (Some Famous Guy) *Another example theorem.*

Proof This proof

1. ends in an enumerate. □

Proposition 1.3 *Note that all theorem-like environments are by default numbered on the same counter.*

Proof This proof ends in a display like so:

$$f(x) = x^2. \qquad \square$$

1.1.4 Macro setup

For now the macro setup only shows how to define some basic macros, and how to use a neat feature of the *mathtools* package:

$$|a|, \quad \left|\frac{a}{b}\right|, \quad |\frac{a}{b}|.$$

Chapter 2

Writing scientific texts in English

This chapter was originally a separate document written by Reto Spöhel. It is reprinted here so that the template can serve as a quick guide to thesis writing, and to provide some more example material to give you a feeling for good typesetting.

2.1 Basic writing rules

The following rules need little further explanation; they are best understood by looking at the example in the booklet by Knuth et al., §2–§3.

Rule 2.1 Write texts, not chains of formulas.

More specifically, write full sentences that are logically interconnected by phrases like 'Therefore', 'However', 'On the other hand', etc. where appropriate.

Rule 2.2 Displayed formulas should be embedded in your text and punctuated with it.

In other words, your writing should not be divided into 'text parts' and 'formula parts'; instead the formulas should be tied together by your prose such that there is a natural flow to your writing.

2.2 Being nice to the reader

Try to write your text in such a way that a reader enjoys reading it. That's of course a lofty goal, but nevertheless one you should aspire to!

Rule 2.3 Be nice to the reader.

Give some intuition or easy example for definitions and theorems which might be hard to digest. Remind the reader of notations you introduced

many pages ago – chances are he has forgotten them. Illustrate your writing with diagrams and pictures where this helps the reader. Etc.

Rule 2.4 Organize your writing.

Think carefully about how you subdivide your thesis into chapters, sections, and possibly subsections. Give overviews at the beginning of your thesis and of each chapter, so the reader knows what to expect. In proofs, outline the main ideas before going into technical details. Give the reader the opportunity to 'catch up with you' by summing up your findings periodically.

Useful phrases: 'So far we have shown that . . . ', 'It remains to show that . . . ', 'Recall that we want to prove inequality (7), as this will allow us to deduce that . . . ', 'Thus we can conclude that . . . . Next, we would like to find out whether . . . ', etc.

Rule 2.5 Don't say the same thing twice without telling the reader that you are saying it twice.

Repetition of key ideas is important and helpful. However, if you present the same idea, definition or observation twice (in the same or different words) without telling the reader, he will be looking for something new where there is nothing new.

Useful phrases: 'Recall that [we have seen in Chapter 5 that] . . . ', 'As argued before / in the proof of Lemma 3, . . . ', 'As mentioned in the introduction, . . . ', 'In other words, . . . ', etc.

Rule 2.6 Don't make statements that you will justify later without telling the reader that you will justify them later.

This rule also applies when the justification is coming right in the next sentence! The reasoning should be clear: if you violate it, the reader will lose valuable time trying to figure out on his own what you were going to explain to him anyway.

Useful phrases: 'Next we argue that . . . ', 'As we shall see, . . . ', 'We will see in the next section that . . . , etc.

2.3 A few important grammar rules

Rule 2.7 There is (almost) *never* a comma before 'that'.

It's really that simple. Examples:

> We assume that . . .
> *Wir nehmen an, dass . . .*

> It follows that ...
> *Daraus folgt, dass ...*
>
> 'thrice' is a word that is seldom used.
> *'thrice' ist ein Wort, das selten verwendet wird.*

Exceptions to this rule are rare and usually pretty obvious. For example, you may end up with a comma before 'that' because 'i.e.' is spelled out as 'that is':

> For $p(n) = \log n/n$ we have ... However, if we choose p a little bit higher, that is $p(n) = (1+\varepsilon)\log n/n$ for some $\varepsilon > 0$, we obtain that...

Or you may get a comma before 'that' because there is some additional information inserted in the middle of your sentence:

> Thus we found a number, namely n_0, that satisfies equation (13).

If the additional information is left out, the sentence has no comma:

> Thus we found a number that satisfies equation (13).

(For 'that' as a relative pronoun, see also Rules 2.9 and 2.10 below.)

Rule 2.8 There is usually no comma before 'if'.

Example:

> A graph is not 3-colorable if it contains a 4-clique.
> *Ein Graph ist nicht 3-färbbar, wenn er eine 4-Clique enthält.*

However, if the 'if' clause comes first, it is usually separated from the main clause by a comma:

> If a graph contains a 4-clique, it is not 3-colorable .
> *Wenn ein Graph eine 4-Clique enthält, ist er nicht 3-färbbar.*

There are more exceptions to these rules than to Rule 2.7, which is why we are not discussing them here. Just keep in mind: don't put a comma before 'if' without good reason.

Rule 2.9 Non-defining relative clauses have commas.

Rule 2.10 Defining relative clauses have no commas.

In English, it is very important to distinguish between two types of relative clauses: defining and non-defining ones. This is a distinction you absolutely need to understand to write scientific texts, because mistakes in this area actually distort the meaning of your text!

It's probably easier to explain first what a *non-defining* relative clause is. A non-defining relative clauses simply gives additional information *that could also be left out* (or given in a separate sentence). For example, the sentence

> The WeirdSort algorithm, which was found by the famous mathematician John Doe, is theoretically best possible but difficult to implement in practice.

would be fully understandable if the relative clause were left out completely. It could also be rephrased as two separate sentences:

> The WeirdSort algorithm is theoretically best possible but difficult to implement in practice. [By the way,] WeirdSort was found by the famous mathematician John Doe.

This is what a non-defining relative clause is. *Non-defining relative clauses are always written with commas.* As a corollary we obtain that you cannot use 'that' in non-defining relative clauses (see Rule 2.7!). It would be wrong to write

> ~~The WeirdSort algorithm, that was found by the famous mathematician John Doe, is theoretically best possible but difficult to implement in practice.~~

A special case that warrants its own example is when 'which' is referring to the entire preceding sentence:

> Thus inequality (7) is true, which implies that the Riemann hypothesis holds.

As before, this is a non-defining relative sentence (it could be left out) and therefore needs a comma.

So let's discuss *defining* relative clauses next. A defining relative clause tells the reader *which specific item the main clause is talking about*. Leaving it out either changes the meaning of the sentence or renders it incomprehensible altogether. Consider the following example:

> The WeirdSort algorithm is difficult to implement in practice. In contrast, the algorithm that we suggest is very simple.

Here the relative clause 'that we suggest' cannot be left out – the remaining sentence would make no sense since the reader would not know which algorithm it is talking about. This is what a defining relative clause is. *Defining relative clauses are never written with commas.* Usually, you can use both 'that' and 'which' in defining relative clauses, although in many cases 'that' sounds better.

As a final example, consider the following sentence:

> For the elements in $\mathcal{B}$ which satisfy property (A), we know that equation (37) holds.

Table 21: Things you (usually) don't say

~~It holds (that) ...~~	We have ...	*Es gilt ...*
('Equation (5) holds.' is fine, though.)		
~~x fulfills property $\mathcal{P}$.~~	x satisfies property $\mathcal{P}$.	*x erfüllt Eigenschaft $\mathcal{P}$.*
~~in average~~	on average	*im Durchschnitt*
~~estimation~~	estimate	*Abschätzung*
~~composed number~~	composite number	*zusammengesetzte Zahl*
~~with the help of~~	using	*mit Hilfe von*
~~surely~~	clearly	*sicher, bestimmt*
~~monotonously increasing~~	monotonically incr.	*monoton steigend*
(Actually, in most cases 'increasing' is just fine.)		

This sentence does not make a statement about all elements in $\mathcal{B}$, only about those satisfying property (A). The relative clause is *defining*. (Thus we could also use 'that' in place of 'which'.)

In contrast, if we add a comma the sentence reads

> For the elements in $\mathcal{B}$, which satisfy property (A), we know that equation (37) holds.

Now the relative clause is *non-defining* – it just mentions in passing that all elements in $\mathcal{B}$ satisfy property (A). The main clause states that equation (37) holds for *all* elements in $\mathcal{B}$. See the difference?

2.4 Things you (usually) don't say in English – and what to say instead

Table 21 lists some common mistakes and alternatives. The entries should not be taken as gospel – they don't necessarily mean that a given word or formulation is wrong under all circumstances (obviously, this depends a lot on the context). However, in nine out of ten instances the suggested alternative is the better word to use.

Chapter 3

Typography

3.1 Punctuation

Rule 3.1 Use opening (‘) and closing (’) quotation marks correctly.

In LaTeX, the closing quotation mark is typed like a normal apostrophe, while the opening quotation mark is typed using the French *accent grave* on your keyboard (the *accent grave* is the one going down, as in *frère*).

Note that any punctuation that *semantically* follows quoted speech goes inside the quotes in American English, but outside in Britain. Also, Americans use double quotes first. Oppose

> “Using ‘lasers,’ we punch a hole in . . . the Ozone Layer,” Dr. Evil said.

to

> ‘Using “lasers”, we punch a hole in . . . the Ozone Layer’, Dr. Evil said.

Rule 3.2 Use hyphens (-), en-dashes (–) and em-dashes (—) correctly.

A hyphen is only used in words like ‘well-known’, ‘3-colorable’ etc., or to separate words that continue in the next line (which is known as hyphenation). It is entered as a single ASCII hyphen character (-).

To denote ranges of numbers, chapters, etc., use an en-dash (entered as two ASCII hyphens --) with no spaces on either side. For example, using Equations (1)–(3), we see. . .

As the equivalent of the German *Gedankenstrich*, use an en-dash with spaces on both sides – in the title of Section 2.4, it would be wrong to use a hyphen instead of the dash. (Some English authors use the even longer emdash (—)

instead, which is typed as three subsequent hyphens in LaTeX. This emdash is used without spaces around it—like so.)

3.2 Spacing

Rule 3.3 Do not add spacing manually.

You should never use the commands `\\` (except within tabulars and arrays), `\␣` (except to prevent a sentence-ending space after Dr. and such), `\vspace`, `\hspace`, etc. The choices programmed into LaTeX and this style should cover almost all cases. Doing it manually quickly leads to inconsistent spacing, which looks terrible. Note that this list of commands is by no means conclusive.

Rule 3.4 Judiciously insert spacing in maths where it helps.

This directly contradicts Rule 3.3, but in some cases TeX fails to correctly decide how much spacing is required. For example, consider

$$f(a,b) = f(a+b, a-b).$$

In such cases, inserting a thin math space `\,` greatly increases readability:

$$f(a,b) = f(a+b,\, a-b).$$

Along similar lines, there are variations of some symbols with different spacing. For example, Lagrange's Theorem states that $|G| = [G : H]|H|$, but the proof uses a bijection $f\colon aH \to bH$. (Note how the first colon is symmetrically spaced, but the second is not.)

Rule 3.5 Learn when to use `\␣` and `\@`.

Unless you use 'french spacing', the space at the end of a sentence is slightly larger than the normal interword space.

The rule used by TeX is that any space following a period, exclamation mark or question mark is sentence-ending, except for periods preceded by an upper-case letter. Inserting `\` before a space turns it into an interword space, and inserting `\@` before a period makes it sentence-ending. This means you should write

```
Prof.\ Dr.\ A. Steger is a member of CADMO\@.
If you want to write a thesis with her, you
should use this template.
```

which turns into

> Prof. Dr. A. Steger is a member of CADMO. If you want to write a thesis with her, you should use this template.

The effect becomes more dramatic in lines that are stretched slightly during justification:

> Prof. Dr. A. Steger is a member of CADMO. If you

Rule 3.6 Place a non-breaking space (`~`) right before references.

This is actually a slight simplification of the real rule, which should invoke common sense. Place non-breaking spaces where a line break would look 'funny' because it occurs right in the middle of a construction, especially between a reference type (Chapter) and its number.

3.3 Choice of 'fonts'

Professional typography distinguishes many font attributes, such as family, size, shape, and weight. The choice for sectional divisions and layout elements has been made, but you will still occasionally want to switch to something else to get the reader's attention. The most important rule is very simple.

Rule 3.7 When emphasising a short bit of text, use `\emph`.

In particular, *never* use bold text (`\textbf`). Italics (or Roman type if used within italics) avoids distracting the eye with the huge blobs of ink in the middle of the text that bold text so quickly introduces.

Occasionally you will need more notation, for example, a consistent typeface used to identify algorithms.

Rule 3.8 Vary one attribute at a time.

For example, for WeirdSort we only changed the shape to small caps. Changing two attributes, say, to bold small caps would be excessive (LaTeX does not even have this particular variation). The same holds for mathematical notation: the reader can easily distinguish g_n, $G(x)$, $\mathcal{G}$ and G.

Rule 3.9 Never underline or uppercase.

No exceptions to this one, unless you are writing your thesis on a typewriter. Manually. Uphill both ways. In a blizzard.

3.4 Displayed equations

Rule 3.10 Insert paragraph breaks *after* displays only where they belong. Never insert paragraph breaks *before* displays.

LaTeX translates sequences of more than one linebreak (i.e., what looks like an empty line in the source code) into a paragraph break in almost all contexts. This also happens before and after displays, where extra spacing is inserted to give a visual indication of the structure. Adding a blank line in these places may look nice in the sources, but compare the resulting display

$$a = b$$

to the following:

$$a = b$$

The first display is surrounded by blank lines, but the second is not. It is bad style to start a paragraph with a display (you should always tell the reader what the display means first), so the rule follows.

Rule 3.11 Never use `eqnarray`.

It is at the root of most ill-spaced multiline displays. The *amsmath* package provides better alternatives, such as the `align` family

$$\begin{aligned} f(x) &= \sin x, \\ g(x) &= \cos x, \end{aligned}$$

and `multline` which copes with excessively long equations:

$$\begin{multline} \mathrm{P}\bigl[X_{t_0} \in (z_0, z_0 + dz_0], \ldots, X_{t_n} \in (z_n, z_n + dz_n]\bigr] \\ = \nu(dz_0) K_{t_1}(z_0, dz_1) K_{t_2 - t_1}(z_1, dz_2) \cdots K_{t_n - t_{n-1}}(z_{n-1}, dz_n). \end{multline}$$

3.5 Floats

By default this style provides floating environments for tables and figures. The general structure should be as follows:

```
\begin{figure}
  \centering
  % content goes here
  \caption{A short caption}
  \label{some-short-label}
\end{figure}
```

Note that the label must follow the caption, otherwise the label will refer to the surrounding section instead. Also note that figures should be captioned at the bottom, and tables at the top.

The whole point of floats is that they, well, *float* to a place where they fit without interrupting the text body. This is a frequent source of confusion and changes; please leave it as is.

Rule 3.12 Do not restrict float movement to only 'here' (`h`).

If you are still tempted, you should avoid the float altogether and just show the figure or table inline, similar to a displayed equation.

Chapter 4

Introduction

In recent years, image-based camera localization has been a key task in the areas of augmented reality, virtual reality, robotics, autonomous driving, etc. There are many methods relying on RGB-D cameras which are quite robust and accurate. However, the RGB-D cameras are power consuming, which makes approaches based on mobile and wearable cameras more attractive. In this thesis, we focus on camera localization from RGB images and aim for real-time, efficient, and robust camera localization.

Traditionally, the PnP [2] algorithm is used to solve the camera localization problem by computing the 6D camera pose given the 2D and 3D corresponding coordinates of multiple points. The fundamental problems of traditional approaches relying on local image features are textureless environments and robustness against strong changes in illumination/occlusion/viewpoint/etc. between the localized image and the given 3D model. To address these limitations, recently, deep learning has been applied to predict the 2D and 3D coordinates of these control points [1] and has achieved superior results. However, these approaches are typically very time-consuming to train and evaluate. In this thesis, we want to combine the benefits of both worlds and develop a learned approach that is efficient and robust.

The problem of 6D object pose estimation from RGB images is related to the problem of camera localization. Recently, an efficient, single shot approach [3] for simultaneously detecting an object in an RGB image and predicting its 6D pose without requiring multiple stages has been proposed. This approach resulted in an improvement over the state-of-the-art in terms of accuracy and efficiency, and addressed the challenges on keypoint occlusion and multiple object pose estimation. In this project, we aim to adapt this approach for efficient image-based camera localization. To this end, we will define keypoints on the 3D room layout for indoor environments and predict the projections of these 3D keypoints. The 6D pose will then be computed using PnP [2] based on the correspondences between 2D predictions and 3D

reference keypoints.

The major challenges of the project include limited data for training the localization task, occlusion, and motion. To address these issues, one initial idea is to increase the number of keypoints without slowing down the method, which is a direction to go for higher accuracy and robustness. [1] provided a trainable RANSAC approach for larger set of control points and can be integrated with the current model. Cutting edge methods could also be learned and utilized to achieve a higher performance.

4.1 Motivation

In recent years, image-based camera localization has great and wide applications in multiple areas. Autonomous localization and navigation is necessary for a moving robot. Augment reality on images requires camera pose or localization. To view virtual environment, corresponding viewing angle needs to be computed.

Furthermore, unlike some other technics that require special devices, e.g. Lidar sensors, RGB-D cameras, etc, cameras are ubiquitous nowadays and people carry with their mobile phones that have cameras every day. Thus, we want to utilize only RGB images from 2D cameras to realize image-based camera localization.

As the current approaches are time-consuming [1] or can not be generalized to new scenes [4], we aim to come up with an efficient, real-time, and robust camera localization approach.

4.2 Background

4.2.1 Overview on localization tools

There are many localization tools nowadays, among which GPS is very commonly used outdoors, but it cannot be used indoors. Lidar, UWB, WiFi AP et al are effective indoor localization tools. However, they require special devices or data collections in advance. Compared to these tools, camera photos can provide higher discriminated features and more information, but in the same time require higher computation ability.

There are also many effective and robust methods relying on depth information acquired by RGB-D cameras currently. However, the RGB-D cameras are power consuming and not ubiquitous as 2D cameras. Thus, passive RGB images that are more commonly used and easy to be acquired by mobile devices and wearable cameras become more attractive. In this work we focus on RGB images that could be captured by 2D cameras, and do not rely

on depth information. Compared to other localization tools, image-based camera localization is the most flexible and low cost one.

4.2.2 Image-based camera localization

Image-based camera localization [5] is to compute camera poses under some world coordinate system from images or video captured by the cameras. Image-based camera localization can be classified into two categories according to that environments are prior or not: the one with known environments and the other one with unknown environments. Then one with known environments are usually the PnP problem studies, and the one with unknown environments consists of the methods with online and real-time environment mapping and the methods without online and real-time environment mapping. The former is commonly known as Simultaneous Localization and Mapping (SLAM) and the latter is the middle procedure of the commonly known structure from motion (SFM). In this thesis, we are not doing any mapping or reconstruction for unknown environments since we are aiming for real-time localization given only single image as input.

There are also some approaches using convolutional neural network to predict camera pose directly from the 2D images or to compute 6D pose in some other way without using PnP algorithm. [4] predicts the orientation and translation of a camera given only a single picture. However, this approach is solving camera relocalization problem, and it can only predict in the same scene that the training period learnt. While this approach is useful in many robotic applications such as navigation and Simultaneous Localization and Mapping (SLAM), it cannot be generalised to a camera localization problem in a new/unseen scene. SSD-6D [**?**] relies on SSD architecture to predict 2D bounding boxes and a rough orientation estimate. Then based on the size of the 2D bounding box, it estimates the depth of the object and lift the 2D detection to 6D. However, the refinement step of this approach increases the running time a lot that cannot make a real-time prediction feasible.

Recently, a real-time single shot 6D object pose estimation approach [3] has been proposed. 6D object pose estimation is related with camera pose estimation which let us see a possibility of realizing real-time camera localization task in a general scene. [3] is using a single shot deep convolutional neural network to predict the 2D projections of the object's 3D bounding boxes, and then use a PnP algorithm to compute 6D object pose. We adapt this approach for our camera localization task. Namely, we use a PnP algorithm to calculate the camera pose from the 2D coordinates predicted by the network and the corresponding 3D keypoints of some known 3D models.

4.2.3 PnP algorithm

As we are applying PnP algorithm to calculate the camera pose according to some corresponding 2D and 3D points. Here is a introduction to PnP algorithm.

Camera pose determinations from known 3D space points are called perspective-n-point problem, namely PnP problem. Let n be the number of used points. When $n \geq 6$, the problem is linear. When $n = 3, 4, 5$, the problem is nonlinear. And when $n < 3$, there is no solution. Although the P3P problem has been well solved, but there may exist multiple solutions. In this work, we set the number of keypoints per object as 9. Although there may sometimes be outliers due to occlusion or other reasons, we can still guarantee it is a linear problem at most of the cases.

When the PnP problem is linear, there are also a lot of works studying on efficient optimizations for the camera poses from small number of points. [2] provide an accurate O(n) solution to the PnP problem, called EPnP which is widely used today. In our approach, we also utilize EPnP and compare it with other PnP solutions.

4.2.4 6D object pose estimation

There are also lots of current works on 6D object pose estimation. Traditional approaches are mainly local image features extraction and matching. There are many fast keypoint and edge-based methods for textured objects, but not effective to weakly textured or untextured objects, or low resolution video streams.

Recently, deep learning methods are utilized in solving 6D object pose estimation problem and have achieved outstanding performance. BB8 [**?**] is a 6D object detection pipeline made of one CNN to coarsely segment the object and another to predict the 2D locations of the projections of the object's 3D bounding box given the segmentation. It then use PnP algorithm to compute 6D pose using the corresponding points. The method is effective but slow due to its multi-stage nature. SSD-6D [**?**] is another approach which relies on SSD architecture to predict 2D bounding boxes and a very rough estimate of the object's orientation in a single step. Then the object's depth is approximated from the size of the 2D bounding box in the image and the 2D detection is lifted to 6D. However, both BB8 and SSD-6D methods require a post-processing step to refine the result, which increase the running time linearly and make the methods not feasible for real-time tasks.

4.2.5 Constraints on image-based camera localization problem

The common problem for camera localization or object pose detection is that the datasets available are rather limited, compared to datasets for other tasks like classification or tagging. Specifically, the most common datasets for detection contain thousands to hundreds of thousands of images with dozens to hundreds of tags, while classification datasets have millions of images with tens or hundreds of thousands of categories. Labeling images for detection is far more expensive than labeling for classification or tagging where tags are often user-supplied for free. These result in object detection problem lack of enough datasets and labels.

In this work, we generate the 2D projections of object's 3D bounding boxes for multiple objects as labels for indoor environments. We take Scannet dataset as the realistic dataset and SunCG together with six tools for synthetic dataset. The labels generated can also be used in future works on related topics.

4.3 Focus of this work

As clarified in the previous background section, we are aiming for a real-time and robust image-based camera localization approach that can predict the camera pose in a new scene. The new scene could be in similar environment as the training scenes, e.g. indoor environment with furnitures, but they should not be the same scenes. The approach should also be able to generalise to various object with no need of precise and detailed texture.

We adapt the recent real-time single shot 6D object pose prediction approach [3] to our camera localization approach. We also generate new labels for indoor environment dataset. Basically, the approach uses a single shot deep CNN architecture to find the corresponding 2D image coordinates of the 3D ground keypoints for some known 3D models in the environments and then applies a PnP algorithm to calculate the camera pose. We choose the 3D keypoints as the 8 corners of the object's 3D bounding box plus the center point of the object. The network is single shot and end-to-end trainable, and the model is also accurate without requiring any post-processing, so it is faster than the other current existing methods which are multi-stage or require post-processing for pose refinement.

Chapter 5

Related works

In this chapter we review the existing works on camera localization from classical feature and template matching methods to newer end-to-end trainable CNN-based methods. Since we use 6D object pose estimation to realize our camera localization, object detection and 6D pose estimation related works are also reviewed.

5.1 Classical methods

Traditional RGB object instance recognition and pose estimation works are mainly local feature extraction and mapping.

5.2 RGB-D methods

5.3 CNN-based methods

5.3.1 Camera relocalization

5.3.2 Object detection

R-CNN and further methods propose various locations and regions

SSD

YOLO YOLO is a single shot object detection approach and is proposed in the same time as SSD. Most of the works for object detection before YOLO have a first step to propose possible regions of the object, and then do the classification and refinement. The multiple stages make the model hard to optimize since each component must be trained separately. The post-processing step also makes the training and prediction rather slow.

YOLO change the network from doing classification to regression for object detection. Instead of doing classification on a proposed region, YOLO can predict the location of the bounding boxes of the object directly. Given a full image, YOLO use a single network to predict all bounding boxes across all classes for the image simultaneously in one evaluation. Thus, YOLO realizes a single shot, end-to-end trainable network for object detection task.

Although YOLO cannot achieve the state-of-the-art accuracy but it is extremely fast and capable for real-time object detection. Compared with other real-time systems, it has more than twice of the mean precision. Although YOLO has lower recall compared to region proposal-based methods, it has less than half false positives on background errors compared to Fast R-CNN.

There are some constraints of YOLO as well. The first one is that there is a limit on the number of nearby objects that the model can predict. Since YOLO partition the image into multiple grid, and each grid cell can only predict one class, although it can predict B bounding boxes per grid, there is still only one object predicted as output. If two objects are very close and have their center in the same grid, only one of them can be predicted. The second one is that YOLO does not achieve the state-of-art accuracy. There is significant number of localization errors and relatively low recall of YOLO compared to region proposal-based methods.

YOLO v2 YOLO v2 has done some improvements to the YOLO detection method. Inspired by Faster R-CNN, which use the region proposal network(RPN) to predict offsets and confidences for anchor boxes, YOLO v2 removed the fully connected layer at the end of the network, and instead use anchor boxes to predict bounding boxes. YOLO v2 decouples the class prediction mechanism from the spatial location and instead predict class for every anchor box. In this way, the limitation of only one class is predicted per grid cell is released, and now we can better predict nearby objects.

5.3.3 6D object pose detection

BB8 BB8 [**?**] is a 6D object detection approach which consists of one CNN to realize a coarse segmentation and another to predict the 2D locations of the projections of the object's 3D bounding box, which are then used by a PnP algorithm to compute the 6D object pose. The segmentation stage generates a 2D segmentation mask for presenting a cropped image to the second network. There is also an optional additional step that refines the predicted poses. The method is slow due to its multi-stage nature.

SSD-6D The SSD-6D [**?**] approach relies on the SSD architecture to predict object's 2D bounding boxes and a pool of the most likely 6D poses for that instance. It then predicts the approximated depth of the object from the size

of the 2D bounding box in the image and lift the output from 2D to 6D object pose. In the final step, the approach refine each pose in every pool and select the best after verification. As the method require a refinement step to get a good accuracy, the running time is increased linearly with the number of objects being detected.

Chapter 6

Dataset

Dummy text.

6.1 Dataset types

Dummy text.

6.1.1 Synthetic data

Dummy text.

6.1.2 Realistic data

Dummy text.

Scannet

Example Paragraph Dummy text.

Example Subparagraph Dummy text.

Appendix A

Dummy Appendix

You can defer lengthy calculations that would otherwise only interrupt the flow of your thesis to an appendix.

Bibliography

[1] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. Dsac-differentiable ransac for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6684–6692, 2017.

[2] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[3] Bugra Tekin, Sudipta N Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 292–301, 2018.

[4] Jian Wu, Liwei Ma, and Xiaolin Hu. Delving deeper into convolutional neural networks for camera relocalization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651. IEEE, 2017.

[5] Yihong Wu, Fulin Tang, and Heping Li. Image-based camera localization: an overview. *Visual Computing for Industry, Biomedicine, and Art*, 1(1):1–13, 2018.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.