Category-level Object Models to Enhance Monocular SLAM

Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science
in
Electronics And Communication Engineering
by Research

by

RISHABH KHAWAD
201331187
rishabh.khawad@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
JULY 2018
It is certified that the work contained in this thesis, titled “Category-level Object Models to Enhance Monocular SLAM” by RISHABH KHAWAD, has been carried out under my supervision and is not submitted elsewhere for a degree.

Date

Adviser: Prof. K. Madhava Krishna
To my Mother and Brother.
Acknowledgments

I had a great time working with the Robotics Research Center, IIIT Hyderabad. First and foremost, I would like to thank my guide Prof. K. Madhava Krishna for believing in me and giving me an opportunity to work at his lab and for his guidance, patience and support throughout my thesis and other lab work. His push made it possible.

I would like to extend my thanks to Jatavallabha Krishna Murthy for being my guru. Anytime anywhere, me having any problem while working, he was there with the solution. If not for him, my thesis would not have been possible. So, thank you very much for your constant support, trust and the calmness. I would also like to thank Parv Keshavlal Parkhiya, one of the smartest mind, for being there. His way of handling problems and putting it out encouraged me a lot and gave the required confidence. I appreciate your perseverance, all the efforts and thanks for bearing me.

A special thanks to Roopal Nahar for helping me out with the most important and toughest table of the thesis. Thanks for the emotional support I needed and taking care of me. You made, working for the thesis in lab much more fun and interesting with all the randomness and fights. It would have definitely not been possible without you.

I am also grateful to Utsav Shah, Nazrul Haque, Vignesh Prasad, Gourav Kumar, Sudeep Fadadu, Junaid, Sarthak Upadhyay. The first ones to guide me in the lab projects. And finally thanks to all other lab mates for providing constant support throughout the course of thesis. Last but not the least, I express my profound gratitude to my family and to all my friends (long list to put up) for their patience and undivided support in overcoming numerous obstacles I have been facing through my research.

To my family and friends

“For your support, your patience, your faith,
Because you were there and you understood.”
Abstract

The thesis presents a novel framework for monocular object-based simultaneous localization and mapping (SLAM) in quasi-static scenes.

This presents a new paradigm for real-time object-oriented SLAM with a monocular camera. Contrary to previous approaches, that rely on object-level models, we construct category-level models from CAD collections which are now widely available. To alleviate the need for huge amounts of labeled data, we develop a rendering pipeline that enables synthesis of large data sets from a limited amount of manually labeled data. Using data thus synthesized, we learn category-level models for object deformations in 3D, as well as discriminative object features in 2D. These category models are instance-independent and aid in the design of object landmark observations that can be incorporated into a generic monocular SLAM framework. Where typical object-SLAM approaches usually solve only for object and camera poses, we also estimate object shape on-the-fly, allowing for a wide range of objects from the category to be present in the scene. Moreover, since our 2D object features are learned discriminatively, the proposed object-SLAM system succeeds in several scenarios where sparse feature-based monocular SLAM fails due to insufficient features or parallax. Also, the proposed category-models help in object instance retrieval, useful for Augmented Reality (AR) applications. We evaluate the proposed framework on multiple challenging real-world scenes and show — to the best of our knowledge — first results of an instance-independent monocular object-SLAM system and the benefits it enjoys over feature-based SLAM methods.
# Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Monocular Object SLAM</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Related Work</td>
<td>4</td>
</tr>
<tr>
<td>1.2.1 Object-SLAM</td>
<td>4</td>
</tr>
<tr>
<td>1.2.2 Object-Category Models</td>
<td>5</td>
</tr>
<tr>
<td>1.2.3 Keypoint Localization Using CNNs</td>
<td>5</td>
</tr>
<tr>
<td>1.2.4 Render Pipelines for Data Synthesis</td>
<td>5</td>
</tr>
<tr>
<td>2 Category-Level Object Models</td>
<td>8</td>
</tr>
<tr>
<td>2.1 Category-Level Models</td>
<td>8</td>
</tr>
<tr>
<td>2.1.1 Semantic Keypoints</td>
<td>9</td>
</tr>
<tr>
<td>2.1.2 Shape Priors</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Applications to Augmented Reality (AR): Instance Retrieval</td>
<td>12</td>
</tr>
<tr>
<td>3 Keypoint Localization using Convolutional Neural Network Architectures</td>
<td>14</td>
</tr>
<tr>
<td>3.1 Discriminative Feature Extraction</td>
<td>14</td>
</tr>
<tr>
<td>3.2 CNNs for Keypoint Localization</td>
<td>14</td>
</tr>
<tr>
<td>3.3 Stacked Hourglass Network</td>
<td>15</td>
</tr>
<tr>
<td>3.3.1 Original Architecture</td>
<td>15</td>
</tr>
<tr>
<td>3.3.2 Our Enhancements</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Dataset Preparation for Training</td>
<td>16</td>
</tr>
<tr>
<td>3.4.1 PASCAL 3D+ Real Set</td>
<td>16</td>
</tr>
<tr>
<td>3.4.2 Render Pipeline - Synthetic Set</td>
<td>17</td>
</tr>
<tr>
<td>3.5 Training of the Network</td>
<td>19</td>
</tr>
<tr>
<td>3.6 Results</td>
<td>20</td>
</tr>
<tr>
<td>3.7 Generalization to Other Object Categories</td>
<td>22</td>
</tr>
<tr>
<td>4 Monocular Object-SLAM</td>
<td>24</td>
</tr>
<tr>
<td>4.1 Object Measurement Factors Using Category-Models</td>
<td>24</td>
</tr>
<tr>
<td>4.1.1 3D Pose-SLAM</td>
<td>25</td>
</tr>
<tr>
<td>4.1.2 Object Observation Factors</td>
<td>25</td>
</tr>
<tr>
<td>4.1.3 Object-SLAM with Category-Level Models (Proposed Approach)</td>
<td>25</td>
</tr>
<tr>
<td>4.2 Implementation Details</td>
<td>26</td>
</tr>
<tr>
<td>4.2.1 Object Detection and Data Association</td>
<td>26</td>
</tr>
<tr>
<td>4.2.2 Object Initialization in 3D</td>
<td>27</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.3 Results</td>
<td>27</td>
</tr>
<tr>
<td>4.3.1 Object-SLAM Evaluation</td>
<td>27</td>
</tr>
<tr>
<td>4.3.2 Dataset</td>
<td>27</td>
</tr>
<tr>
<td>4.3.3 Discussion</td>
<td>29</td>
</tr>
<tr>
<td>5 Extensions: Handling severely occluded and truncated objects</td>
<td>32</td>
</tr>
<tr>
<td>5.1 Handling Occlusion</td>
<td>33</td>
</tr>
<tr>
<td>5.2 Handling Truncation</td>
<td>34</td>
</tr>
<tr>
<td>5.3 Results</td>
<td>34</td>
</tr>
<tr>
<td>6 Conclusions</td>
<td>37</td>
</tr>
<tr>
<td>Bibliography</td>
<td>39</td>
</tr>
</tbody>
</table>
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Monocular SLAM visualization</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Sample output of the proposed object-SLAM system. Given RGB images from a monocular camera (here from a quadrotor), we estimate the trajectory of the camera, as well as the poses and shapes of various objects in the scene. The proposed category-specific models can be used either in an incremental setting for online SLAM, or in a batch setting for offline factor-graph optimization.</td>
<td>3</td>
</tr>
<tr>
<td>1.3</td>
<td>Complete pipeline for learning and incorporating category-level models for object-SLAM</td>
<td>7</td>
</tr>
<tr>
<td>2.1</td>
<td>Visualization of the learnt linear-subspace shape models. Using the learnt deformation basis, we can deform the mean shape for the category in a more meaningful manner, i.e., such that deformations still produce meaningful chair shapes, and not an arbitrary collection of 3D points.</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>A simple illustration of how the learnt shape models can be used for estimating the actual shape of an instance from the object category. In this experiment, a mean wireframe was allowed to deform only using directions determined by the deformation basis such that its 3D key point locations matched with those of the actual model. The predicted model ends up with very little error (of the order of 3 cm), which demonstrates the efficacy of the proposed approach. (Scale in cm)</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>All 10 keypoints of chair category used in our shape presentation.</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>Mean wireframe of a chair. These arrows in the image correspond to the direction of almost all the possible deformations for this object category chair. And accordingly, we have distinct instances of the category shown in Fig. 2.1.</td>
<td>12</td>
</tr>
<tr>
<td>2.5</td>
<td>Qualitative results for instance retrieval. Query images and their retrieved CAD models are shown.</td>
<td>13</td>
</tr>
<tr>
<td>2.6</td>
<td>Rendering objects estimated in a run along with the drone and its trajectory. Retrieval of the scene captured.</td>
<td>13</td>
</tr>
<tr>
<td>3.1</td>
<td>An illustration of a single “hourglass” module. Each box in the architecture corresponds to Residual Module shown in the figure. The number of features is consistent across the whole hourglass. Diagram is reproduced from [20] with the author’s permission.</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Proposed network architecture. The yellow and red blocks show the keypoint likelihoods and their disparities separately. Both of these blocks are Collectively trained allowing the network to detect the pairwise relations among keypoints and helps in preventing the overfitting. Adopted from [19].</td>
<td>16</td>
</tr>
<tr>
<td>3.3</td>
<td>Real Set from PASCAL3D.</td>
<td>16</td>
</tr>
</tbody>
</table>
3.4 Visualization of the Render Pipeline, same has been done for the chair class. Diagram reproduced from [29] with permission from the authors. .......................................................... 17
3.5 Synthetic Set Generation ....................................................................................... 17
3.6 Local frame of Object and Camera ..................................................................... 19
3.7 Applied transformations for connecting the object’s frame to the camera’s frame. .... 19
3.8 Insight of the Blender Framework (3D Visualization of the world in the blender framework). Camera position and other sample parameters - Elevation (Ele) and Azimuth (Azmth) are localized with respect to the Object placed at the center(0,0,0) of the world. 20
3.9 Progress of training the hourglass CNN using the proposed render pipeline. Loss per epoch when training on synthetic data followed by finetuning on a mix of real and synthetic images is illustrated here. ................................................................. 21
3.10 Qualitative Results showing the 2D keypoint localization performance of the proposed architecture [20]. Keypoints with more than 0.7 confidence scores output by the CNN are shown per instance. Discriminative features are extracted persistently across all poses. The last row shows some failed cases. .................................................................. 22
3.11 Bar graph for confidence variation of each keypoint. Along with the order of keypoints. 23
3.12 Some successful cases for keypoint localization on Laptops .................................. 23
4.1 Initialization of 3D object from 2D monocular image ............................................. 27
4.2 Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM with/without Object Loop Closure detection (OLC) ................................................................. 29
4.3 Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM with/without Object Loop Closure detection (OLC) ................................................................. 30
4.4 Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM with/without Object Loop Closure detection (OLC) ................................................................. 30
4.5 Rendering objects estimated in a run where the robot rotates in place. ORB-SLAM [17] fails to initialize due to insufficient parallax. ................................................................. 31
4.6 Qualitative results on another object category (laptops). ........................................ 31
5.1 Visualization of the Constraints in the scene ......................................................... 32
5.2 Keypoint Localization for the constrained cases .................................................. 33
5.3 In these instances, all the keypoints are localized, even if not visible. A predicted keypoint for that part is annotated ................................................................. 34
5.4 Creation of the truncated instances ..................................................................... 34
5.5 Row 1: Results of the keypoint localization for the truncated object instances. Row 2: Results of the Keypoint localization for the occluded object instances. .......... 35
5.6 Qualitative Results for some Instances from the sequence taken ............................. 35
5.7 Bar graph for confidence variation of each keypoint for both the models ................. 36
5.8 Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM without Object Loop Closure detection (OLC) for our new sequence .................. 36
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Evaluation of the proposed keypoint localization network architecture</td>
<td>20</td>
</tr>
<tr>
<td>3.2</td>
<td>Keypoint localization accuracy (2D) for the stacked hourglass network trained using the proposed render pipeline</td>
<td>21</td>
</tr>
<tr>
<td>4.1</td>
<td>Quantitative analysis of several variants of the proposed object-SLAM approach. Here, the best and worst object localization errors refer to the errors in the most accurately localized and the most erroneously localized object respectively. We also show the average object localization error and accumulated endpoint drifts, wherever applicable.</td>
<td>28</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Simultaneous localization and mapping (SLAM) is a fundamental problem in robotics which involves estimating simultaneously a robot’s configuration, as well as that of the environment. It constitutes two parts: **mapping**, building a map of the environment which the robot is in, and **localization**, navigating this environment using the map while keeping track of the robot’s relative position and orientation. [24, 3, 4]. This knowledge is crucial for performing tasks that call for a higher level of autonomy such as path planning, manipulation etc. Fundamentally SLAM has been a chicken-and-egg problem, we need to have a precise map to solve the localization problem, and we need precise localization to solve the mapping problem. Fig. 1.1 demonstrates dense visual monocular SLAM system along with the scene recreation. SLAM algorithms are tailored to the available resources, hence not aimed at perfection, but at operational compliance.

Figure 1.1: Monocular SLAM visualization

SLAM finds various real-world applications such as autonomous navigation, visual inspection, mapping, and surveillance. Monocular cameras have evolved as popular choices for SLAM, especially on platforms such as hand-held devices and Micro Aerial Vehicles (MAVs). Micro Aerial Vehicles (MAVs)
have become an intriguing platform for many contemporary applications. They have also become standard test bed for many vision and navigation algorithms. I have been working with the monocular drones for a long time, solving the problems such as autonomous navigation, geo-tagging for them [36, 37]. While working, particularly with the monocular vision, we faced problems like scale-factor ambiguity, localization of the bots and hence the motivation for the monocular SLAM. One of the main reasons that pure monocular SLAM is being used and researched is because of the hardware setup required to implement it is easier and simpler. This means that it implies cost cutting in terms of money and physically smaller in size and area than other systems, for example, stereo SLAM.

**Problem Proposed**

In this thesis, we present a new paradigm for real-time object-oriented SLAM with a monocular camera. Our SLAM pipeline rely on the category-level rather than the object-level [26]. Using the keypoint information and the shape priors, we construct the category-specific model of an object. These instance-independent category models are integrated into a generic monocular SLAM framework using the factor graph [9] to formulate the whole pipeline.

### 1.1 Monocular Object SLAM

Most state-of-the-art monocular SLAM systems [17] operate on geometric primitives such as points, lines, and planar patches. Others operate directly on images, without the need for expensive feature extraction steps [10]. However, both these sets of approaches lack the ability to provide a rich semantic description of the scene. Recognizing and keeping track of objects in a scene will enable a robot to build meaningful maps and scene descriptions. Object-SLAM is a relatively new paradigm [32, 26, 16] towards achieving this goal. Summarized in one sentence, object-SLAM attempts to augment SLAM with object information so that robot localization, object location estimation (in some cases, object pose estimation too), and mapping are achieved in a unified framework.

There are two dominant paradigms in object-SLAM research, depending on the way objects are characterized in the SLAM framework. In the first paradigm [26, 7], object-level (instance-specific) models are assumed to be available beforehand. However, the very nature of monocular SLAM with scale ambiguity coupled with the loss in information due to projection onto the image plane renders this paradigm infeasible for monocular object-SLAM systems. The second paradigm [31, 8], assumes a generic model, regardless of the object category. For instance, [8] models all objects as ellipsoids, and [12, 16] model all objects as cuboids. Both these approaches suffer a few disadvantages. Relying on object-level models will result in the need to have precise object models for all instances of an object category. On the other hand, generic models do not give much information about an object beyond the object category label. In many applications, such as manipulation for instance, it is advantageous to know the object pose.
In this thesis, we propose a new paradigm for monocular object-SLAM, that combines the best-of-both-worlds. To enjoy the expressive power of instance-specific models yet retain the simplicity of generic models, we construct category-specific models, i.e., the object category is modeled as a whole. We employ the widely used linear subspace model \cite{18, 40, 33} to characterize an object category and define object observations as factors in a SLAM factor graph \cite{14, 25}. In our object-SLAM formulation, we do not assume any knowledge about the instance (interchangeably referred to as shape) of the object. Rather, we explicitly solve for the object shape in a joint formulation. The object-SLAM backend estimates robot trajectory and map, as well as poses and shapes of all objects in the scene.

Naturally, one would expect that a lot of data will be needed to learn category-specific models that generalize well across object instances, and rightly so. Datasets such as ShapeNet, SceneNet, ObjectNet have made available CAD collections of various object categories. We exploit the ready availability of such CAD collections to construct our category models. These category models capture the deformation modes of objects in 3D. Correspondingly, we leverage recent successes of Convolutional Neural Networks (CNNs) in keypoint localization \cite{18, 19, 21, 20} to train 2D object feature extractors. To alleviate the need for large amounts of manually annotated training data, we design a render pipeline, along the lines of RenderForCNN \cite{29}, to synthesize enormous amounts of training data for category model learning. The presented render pipeline takes in a small volume of manually annotated data and synthesizes a large dataset that can be used to efficiently train a 2D object feature extraction network.
We show that feature detectors learned from the render pipeline are more precise compared to ones that are learned over real data alone, which corroborates claims in [29].

We evaluate our object-SLAM system on multiple challenging real-world sequences and present — to the best of our knowledge — first steps towards instance-independence in monocular object-SLAM. Since we use discriminative 2D features on objects, our system is robust to conditions such as strong rotation, where monocular SLAM approaches usually face catastrophic failure. We present both incremental and batch versions of our object-SLAM pipeline and demonstrate its advantages qualitatively and quantitatively over feature-based visual SLAM approaches [17]. Finally, we show that, using our category-level models, one can perform object instance retrieval and this can be used in many Augmented Reality (AR) applications for overlaying object models in a scene. Fig. 1.2 illustrates an output from our pipeline. Objects are consistently embedded onto the robot’s trajectory and their 3D models are rendered.

1.2 Related Work

Nearly all state-of-the-art SLAM systems [17,10,11] rely on pose-graph (or other factor graph) optimization [15]. In this section we review related work on object-SLAM, and outline certain limitations in them that form motivating factors for the proposed approach.

1.2.1 Object-SLAM

With recent advances and subsequent stabilization of SLAM systems, the community has been devoting attention to incorporating objects into the SLAM framework. To this end, some recent approaches for object-oriented SLAM have been proposed [26,16,12,8,32,31].

Most of these works rely on depth information from RGB-D or stereo sensors [26,7,16,31]. In [26,7], assume an instance-level model of the objects are known a priori. In [26], a real-time 3D object detection algorithm is applied on an RGB-D image stream and these objects are fused along with odometry information in a pose graph optimization scheme. Similarly in [7], a framework for multi-robot object-SLAM is proposed. Again, each robot is equipped with an RGB-D sensor and object models are available a priori.

There is another paradigm in which no instance-level models are available a priori. In [16], association and object poses are solved for jointly, in a factor graph framework, using data from RGB-D cameras. Among monocular object-SLAM/SfM approaches, [8,12] fall under this paradigm. In such approaches, objects are modeled as bounding boxes [12,31] or as ellipsoids [8].

Our method hence falls under a third paradigm, where we assume category-models, and not instance-level models.
1.2.2 Object-Category Models

Over the last few years, object-category models have been applied to several problems in monocular vision. In [33, 18, 40], category-level models are employed to obtain object reconstructions from single images. These approaches demonstrate that the loss of information in the monocular imaging process can be compensated for by incorporating priors on shapes of objects belonging to a particular category.

We use these category models and exploit them to design object observation factors that can be easily incorporated into monocular SLAM, and also generalize across several instances from the category, without the need for modeling all possible instances from the category.

1.2.3 Keypoint Localization Using CNNs

Convolutional Neural Networks (CNNs) have been the driving reason behind recent advances in object detection [23, 13] and object keypoint localization [20, 34, 21, 19]. When run on a GPU, these CNNs are capable of processing image frames with latencies of about 100-300 milliseconds and form important components of our pipeline.

1.2.4 Render Pipelines for Data Synthesis

With the advent of CAD model collections such as [5], 3D data is now in abundance. In [29], image synthesis using a rendering engine is proposed as an alternative to training on real images annotated manually. Models trained for the task of object viewpoint prediction on rendered data (and subsequently finetuned on a smaller dataset comprising real data) are shown to outperform models that have been trained on (larger) real datasets alone. Our experiments corroborate this fact for the task of object keypoint prediction as well.

We build on several of the components described here, however we craft the outputs to create object factors that can be augmented to factor graphs [14] constructed using monocular SLAM approaches. The entire pipeline is summarized in Fig. 1.3 and is explained in the subsequent sections.

Organization

The remaining of this thesis is organized as follows. In chapter 2, we extensively survey the existing approaches and comment on the state-of-the-art of the Object-SLAM, mentioning the related work to the keypoint localization, render pipeline for data synthesis. We describe our Construction of category specific models in chapter 3. We define the shape priors and present methods for learning and formulating them. In the same chapter we discussed about Instance Retrieval and its influence in augmented reality. Followed by, in chapter 4 we present Convolutional Neural Network (CNN) architectures for the task of object keypoint localization in images and the whole process of data generation using the render pipeline for annotated synthetic set. Chapter 3 and chapter 4 are the buildups for the chapter 5.
We present our Object SLAM with category-level models (proposed approach) pipeline and overall framework with all the experiments conducted in chapter 5. In chapter 6, we addressed some visual constraints of scene and laid emphasis on recovering the wireframe even when 2D keypoint localization incurs noise. We proposed pipeline for future work in chapter 6.
Figure 1.3: Complete pipeline for learning and incorporating category-level models for object-SLAM
Chapter 2

Category-Level Object Models

In this chapter, we describe the category-level models that we employ in our object-SLAM system. Our key insight is that shapes of objects from the same category are not arbitrary. Instead, they all follow a set pattern that can be learned by examining several instances. We adopt the widely used linear subspace model \[40, 33, 18, 19\] and represent objects as 3D wireframes.

2.1 Category-Level Models

In the proposed approach, we lay an emphasis on the use of category-level models as opposed to instance-level models for objects. To construct a category level model, each object is first characterized as a set of 3D locations of salient points that are common across all instances of the category, explained in section 2.1.1.

According to the linear subspace model \[18\], the space of actual shapes of a category are confined to a much lower dimensional subspace of \(\mathbb{R}^{3K}\). This is easy to see, since there are several dependencies that exist among the \(K\) keypoints, such as subsets of the keypoints being planar, symmetric, etc. Any object can then be expressed as a mean shape that can be deformed along linearly independent directions (basis vectors) using shape parameters (coefficients of the linear combination).

Mathematically, if \(\bar{S}\) is the mean shape of the category, and \(V_1s\) are a deformation basis obtained from PCA over a collection of aligned ordered 3D CAD models (such as \(5\)),

\[
S = \bar{S} + \sum_{b=1}^{B} \lambda_b V_b = \bar{S} + V\Lambda
\]

(2.1)

where \(B\) is the number of basis vectors (the top-\(B\) eigenvectors after PCA). Here, \(V\) represents the learnt \(3K \times B\) deformation modes. \(\Lambda\) is a \(K\)-vector containing the deformation coefficients. Varying the deformation coefficients produces various shapes from the learnt shape subspace, as shown in Fig. 1.3 (panel: Category Level Shape Priors). Fig. 2.2 demonstrates that we can simply deform along the learnt basis shapes to estimate shapes of objects of the category. This principled method of deforming
along a learnt basis space reduces the ambiguities in recovered shape, which makes this an interesting measurement model for object-based monocular SLAM.

Further, in the chapter, I have discussed the Semantic Keypoints, definition, formulation and learning of the Shape Priors, giving the better insight of constructing a category-specific model.

Figure 2.1: Visualization of the learnt linear-subspace shape models. Using the learnt deformation basis, we can deform the mean shape for the category in a more meaningful manner, i.e., such that deformations still produce meaningful chair shapes, and not an arbitrary collection of 3D points

2.1.1 Semantic Keypoints

For formulating a shape, we could use all the points of the object represented as the dense point cloud comprising of the 3D locations of the points but howsoever it seems to be unsuitable and of no use considering the real-time scenarios. Instead we use some salient points which are consistent across all the instances.
Figure 2.2: A simple illustration of how the learnt shape models can be used for estimating the actual shape of an instance from the object category. In this experiment, a mean wireframe was allowed to deform only using directions determined by the deformation basis such that its 3D key point locations matched with those of the actual model. The predicted model ends up with very little error (of the order of 3 cm), which demonstrates the efficacy of the proposed approach. (Scale in cm)

For example, such salient points for the chair category could be legs of the chair, corners of the chair backrest, and so on. We have a set of 10 keypoints that characterize the shape of a chair. These points are common across all instances of the category. We denote by $S$ the $3K$-vector comprising of an ordered collection of 3D locations of these $K$ salient points, which we refer to as keypoints for that object category. Keypoints for the chair category which we use throughout this work, are shown in Fig. 2.3.

Figure 2.3: All 10 keypoints of chair category used in our shape presentation.
2.1.2 Shape Priors

Forming a 3D shape and generating the pose of an object in a monocular framework (from a single image) is tough given that there is no additional information. We adopted the \([40, 33, 18, 19]\) framework and exploited linear subspace model to tackle this ill-posedness. Modeling the prior information about the object shapes and consequently their 2D projection. This domain knowledge about the 3D shape of an object category is what we call as the Shape Priors.

Formulation

A shape prior is the mathematical model that defines the shape of a given object as a whole. We cogitate that the space of the valid shapes of a particular object is limited to a low-dimensional subspace of the space of all the possible shape of the object. As we speak, shape of each chair is unique, even though it resembles with the shape of other instances of the same category but is very different from the other object category.

Definition

As explained before, basically, we characterize the shape of any instance as the requested accumulation of 3D locations of the keypoints for that particular instance. These keypoints are object specific, like in the case of chairs we have 10 such distinct keypoints.

Mean Shape and deformation basis vector constitutes a Shape prior. The mean shape corresponds to the average of all shapes of a particular class. The deformation basis refers to a set of linearly independent basis vectors which when combined with the mean shape can result in the formation of any specific instance from that class. In Eqn. 2.2, we define the linear subspace model, already explained above. The mean (average) shape deforms along the dimensions specified by the basis vectors to form the shape of a particular instance. Fig. 2.4 demonstrates the different directions of varied deformations.

\[
S = \bar{S} + VA
\]  

Learning Shape Priors

We followed the usual way for the prior leaning which involves the use of large dataset of 3D CAD models annotated with the keypoint information. This method provides us with very accurate 3D descriptors of shape. We selected around 250 CAD models of chair category trying to cover the major instances of the category, which was followed by the 3D annotations of it. It was a time consuming process as it involved annotation of several 2D views of the object followed by propagation of those 2D labels to 3D, since direct 3D labeling is error prone. We did the PCA of these annotated CAD models generating the mean wireframe of the object.
We followed the same pipeline of constructing a category-specific model for another object category laptops. With the less number of distinct instances, it was easier in the case of laptops.

2.2 Applications to Augmented Reality (AR): Instance Retrieval

Instance retrieval (IR) is the problem of retrieving specific instances of a particular object from the given query 2D image scene. Visual instance retrieval is amongst widely studied applications of computer vision, and the research progress has been reported based on several benchmark datasets. Instance retrieval systems retrieve similar 3D objects based on a given query object and regenerate the scene. The developments in techniques for digitizing, modeling and visualizing 3D shapes, in recent times, has led to a boost in the number of available 3D CAD models on the Internet and in the other domain-specific databases [5].

The proposed category-level model certainly achieves instance independence, but instance retrieval itself may be of use in several scenarios, such as Augmented Reality (AR). With the proposed interaction paradigm, (a) 3D models which seem to be comparable from a specific view, or (b) have a high general comparability can be effortlessly found. The shape parameters estimated by our approach can be used to guide an instance search over the keypoint annotated CAD collection. Given a query image (2D), we run a K-Nearest Neighbors search to retrieve the closest instance possible from the CAD collection. In Fig. 2.5, we present results from running a 5-nearest neighbor search and manually choosing the closest instance. Similarly,

\[ \text{nearest}\_\text{model} = \arg\min_{i \in 1...N} \{ \text{Distance}_i \} \]  
(2.3)

where N is the total number of models in our database and

\[ \text{Distance}_i = ||X - Y_i||_2 \]  
(2.4)
Equation 2.4 calculates the distance cost for the given query image (2D) with all the CAD models in our database. In this equation, $X$ is the data of the computed wire frame of the object, $Y$ is the data of the annotated 3D query CAD Models and $i$ is the index of the CAD model for which the distance is computed. We save the top 5-least distances and manually allocate the closest 3D CAD model to the input object and thus trying to retain the reality to the scene retrieved.

Figure 2.5: Qualitative results for instance retrieval. Query images and their retrieved CAD models are shown.

These instances can then be used to render the objects in the scene as well as the robot trajectory. One such rendering for a particular Sequence is shown in Fig 2.6. In this Figure we have shown the output trajectory of our Object SLAM pipeline and the rendered objects recreating the whole scene, using the Gazebo Framework [35].

Figure 2.6: Rendering objects estimated in a run along with the drone and its trajectory. Retrieval of the scene captured.
Chapter 3

Keypoint Localization using Convolutional Neural Network Architectures

3.1 Discriminative Feature Extraction

Since our category-level models rely on keypoints for that category, we need reliable discriminative feature extractors that localize the corresponding keypoints in 2D images. To alleviate this problem, we propose a Convolutional Neural Network (CNN) architectures for the task of localizing semantic keypoints in images. For the same, we adopt the *stacked hourglass* model introduced in [20] and subsequently modified in [19]. The outputs of these networks serve as an initial 2D guess which are subsequently lifted to 3D using methods proposed in Chapter. 2 and 4.

3.2 CNNs for Keypoint Localization

The conventional pipeline for the keypoint localization, has been enhanced to the extent by the convolutional neural networks (ConvNets) [34, 30, 22], a main driver behind the successful rise in performance across many computer vision tasks. The present-day successes in the shape estimation using a single image can be associated to the availability of deep keypoint localization CNN architectures. Some of the earlier approaches for keypoint localization has been presented in [34], establishing a relation between the Viewpoint and Keypoint of the target object. Estimated Keypoint from two distinct scales are composed in conjunction with a viewpoint prior to produce keypoint likelihoods throughout the image. However, the response maps from the CNN were exceedingly multi-modal. As a result, accuracy suffered.

In [30, 22, 18], finetuning subnetworks were proposed to endure the accuracy and refine the estimates from a coarse-grained regressor. In [6], intermediate shape concepts are provided to better supervise the learning process. In the recent times, *stacked hourglass networks* [20] have been proposed for the task of keypoint localization for human pose estimation. This network architecture is, by development, multiscale and have an iterative refinement nature. We pick this as our base network architecture and impose spatial limitations among keypoints, for improvement and increased accuracy. In the next section, I have briefly explained the architecture of the *stacked hourglass network*. 
3.3 Stacked Hourglass Network

Figure 3.1: An illustration of a single "hourglass" module. Each box in the architecture corresponds to Residual Module shown in the figure. The number of features is consistent across the whole hourglass. Diagram is reproduced from [20] with the author’s permission.

3.3.1 Original Architecture

Stacked Hourglass Network is a novel convolutional network architecture for the task of Keypoint localization in RGB images [20]. In the stacked hourglass network, features are processed across all scales maintaining the fixed spatial dimensions (height, width) across the network and consequently capturing the best spatial relationship associated with the object. This consistent scaling makes it different from the existing architectures for keypoint localization [34, 22, 30].

The input to the network is a $3 \times 64 \times 64$ image of the resized, cropped bounding box containing a chair. The most essential attribute of the network is what we call an hourglass [20] explained in Fig. 3.1, which consists of an encoder and a decoder block. The topological structure of the hourglass is symmetric, so for every down going layer present we have a corresponding layer going up. To make up for the loss of information due to pooling in the encoder block, a set of skip connections forward data (via a series of convolutions) to the corresponding decoder block. At the output resolution of each hourglass, 2 consecutive rounds of $1 \times 1$ convolutions are applied to produce the final network predictions, set of keypoint likelihood maps (one map per keypoint) over the entire image. These hourglass modules are stacked multiply on top of each other, iteratively refining the keypoint likelihoods. The resulting output serves directly as the input for the following hourglass module which generates another set of predictions. The key to this stacked approach is the prediction of intermediate heatmaps upon which the loss is applied. An intermediate loss function is applied to the network output at the end of each hourglass.
This intermediate supervision has performed better than scenarios where loss has been applied only at the end of the network [20, 6].

### 3.3.2 Our Enhancements

We adopted the modified version of stacked hourglass network [19], in which CRF-Style loss function is used, applied to all the predictions of each hourglass.

![Proposed network architecture. The yellow and red blocks show the keypoint likelihoods and their disparities separately. Both of these blocks are Collectively trained allowing the network to detect the pairwise relations among keypoints and helps in preventing the overfitting. Adopted from [19].](image)

#### 3.4 Dataset Preparation for Training

One of the hardest problems to solve in deep learning has nothing to do with neural nets: its the problem of getting the right data in the right format. To train the keypoint localization network, we prepared both real chair set and synthetic chair set.

#### 3.4.1 PASCAL 3D+ Real Set

![Real Set from PASCAL3D](image)

For the real set, we use keypoint-annotated data for the chair class of the PASCAL3D [38] dataset. It has around 2300 images having a chair as one of the objects present which then, was segmented out
and scaled to the size of $64 \times 64$. Random horizontal flips, crops, jittering and color space augmentation were employed to synthesize newer samples, multiplying the whole annotated-set to around 4,30,000.

### 3.4.2 Render Pipeline - Synthetic Set

![Diagram of Render Pipeline](image)

Figure 3.4: Visualization of the Render Pipeline, same has been done for the *chair* class. Diagram reproduced from [29] with permission from the authors.

3D CAD models have the potential in generating a large number of rendered images of varied viewpoints, which can be very well exploited by deep CNN training achieving a high validation accuracy. Inspired by RenderForCNN [29], we implement our customized render pipeline for generating huge amounts, approximately a million of synthetic keypoint annotated chair images using a small set of 3D annotated keypoints. We briefly summarize the steps in our render pipeline, and how we exploit its advantages for learning 3D category-models as well as discriminative 2D feature extractors. In Fig. 3.8, we gave an insight of the blender framework and reference for the co-ordinate axis.

![Diagram of Synthetic Set Generation](image)

Figure 3.5: Synthetic Set Generation
Render Pipeline for 3D Category-level Models

First, we choose a collection of CAD models of chairs, comprising of about 250 chairs sampled from the ShapeNet [5] repository. For each chair, we synthesize a few (typically 8) 2D images with predetermined viewpoints (azimuth, elevation, and camera-tilt angles). Keypoints in these images are then annotated (in 2D) manually, and then triangulated to 3D to obtain 3D keypoint locations on the CAD model.

Since the models are already assumed to be aligned, performing a Principal Component Analysis (PCA) over the (mean-subtracted) 3D keypoint locations results in the deformation basis (eigenvector obtained from PCA). This constitutes the category-level model learning phase. An overview of the pipeline developed is illustrated in Fig. 3.5.

Render Pipeline for 2D Feature Extraction

For training 2D feature extractors, we use the 3D keypoint annotated CAD models to synthesize 2D images with Blender [2] as the rendering engine. Apart from using the RenderForCNN [29] framework to randomly sample view parameters and generate 2D images with overlaid backgrounds, we also perform a projection of the 3D keypoints to 2D using the same view parameters to obtain 2D keypoint annotations, explained in the equation below, for training the stacked hourglass architecture [20, 19, 21] for keypoint localization.

\[
x = K \times [R|t] \times X
\]  

(3.1)

In the above equation, x is the keypoints projected from 3D model to 2D image, K is the camera matrix which we get from the intrinsics of the camera used in the blender framework. R is defined as the rotation matrix which is computed as the combination of azimuth and elevation (random sample view parameters) and t is the translation received from the rho, another sample view parameter, defined as the camera-object distance. X is the set of 3D keypoints of the 3D CAD model.

Calculation of the R, Rotation Matrix from the above Eqn. 4.7 is a bit tricky to handle and is not done in the usual way. The annotations of the 3D CAD models are in an object frame and we project every annotation of the keypoint to 2D, that is in the camera frame. So we compute the rotation matrix frame wise frame and then integrate the euler properties of the camera, the combination of azimuth and elevation, in it.

Fig. 3.6 shows the local frame of camera and object inside the blender framework. And it can be seen that there is a difference in the co-ordinate axis of both the frames. The transformations illustrated in Fig. 3.7 shift the focus from object’s frame to the camera’s frame. The output of these transitions are then integrated with the random sample view parameters, the azimuth and the elevation to generate the rotation matrix used for the projection of 3D annotated points (X) to 2D (x).
Advantages of the Proposed Render Pipeline

- Only 2D annotation is required. Annotation on 3D CAD models requires expensive labeling effort. We, however, perform annotations on a small set of rendered 2D images and triangulate them to 3D.

- Keypoint annotations will be available even for occluded parts. This is one major drawback of keypoint localization architectures trained over real data. Since there are no reliable estimates of keypoint locations for occluded parts, these labels are not present in the ground-truth in datasets such as PASCAL3D+ [38]. We demonstrate that our models perform much better due to the availability of occluded keypoint locations as well.

- Using only small volumes of labeled data, we can generate millions of training examples for the keypoint localization CNN.

3.5 Training of the Network

For the task of training CNNs to detect keypoints for the chair category, we have half a million annotated real chair set and we use our render pipeline to synthesize a little over a million keypoint annotated images of various models of chairs that is the synthetic set. Out of the whole dataset, 80% is training set, 15% is Validation Set and 5% is Testing Set. We train the stacked hourglass [20, 19] architecture on this synthesized dataset. We then finetune the CNN on a smaller dataset that comprises of real chair images from the PASCAL3D+ [38] dataset.
The network is trained using the popular Torch7 [48] framework and for optimization, we use rmsprop [49] with a learning rate of $2.5 \times 10^{-4}$. Initial training, as mentioned, is done for 1 million dataset, which takes about 3 days on 2, 12 GB NVIDIA TitanX GPU. As soon as the validation accuracy plateaus, we drop the learning rate by the factor of 10. Batch Normalization is also used to improve the training and further enhance the accuracy.

### 3.6 Results

Here, we evaluate the accuracy of our 2D Keypoint localization network over the chair class of the PASCAL3D [38] dataset and the synthetic dataset generated using the render [29] pipeline. Evaluation is done using the standard Percentage of Correct Keypoints (PCK) metrics used in [39, 34, 20] which reports the percentage of the detections that fall within a normalised distance of very tight threshold of 2px.

<table>
<thead>
<tr>
<th>Approach</th>
<th>PCK (%) $(\alpha = 0.1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zia et al [6]</td>
<td>63.9</td>
</tr>
<tr>
<td>Tulsiani et al [34]</td>
<td>65.0</td>
</tr>
<tr>
<td><strong>Ours (Stacked Hourglass)</strong></td>
<td><strong>87.46</strong></td>
</tr>
</tbody>
</table>

Table 3.1: Evaluation of the proposed keypoint localization network architecture.
Table 3.1 shows the keypoint localization accuracy obtained by the proposed network architecture and its comparison with the approaches [6, 34]. The testing models from which we are computing the PCK of all the approaches above are trained on the PASCAL3D [38] dataset, for maintaining the consistency. The results indicate a significant performance boost in the task of keypoint localization, which also helps in boosting the formulation of the 3D frame, further helping in the Object SLAM pipeline.

Keypoint localization accuracies for several configurations of the hourglass network are shown in Table 3.2. We see that CNNs trained on synthetic data alone fail to generalize to real data from our sequences. Similarly, training on real data (from the PASCAL3D+ dataset) alone performs fairly well on test samples from the same dataset, but fails to generalize to other kinds of data, such as our sequence. However, finetuning the CNN on a combination of both real and synthetic data leads to better generalization. Fig. 3.10 shows the decrease of loss with time when the best performing approach from Table 3.2 is trained using our render pipeline.

Table 3.2: Keypoint localization accuracy (2D) for the stacked hourglass network trained using the proposed render pipeline

<table>
<thead>
<tr>
<th>Test Accuracy</th>
<th>Synthetic only</th>
<th>Real only</th>
<th>Synthetic + Real</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic data</td>
<td>90.67%</td>
<td>85.90%</td>
<td>94.51%</td>
</tr>
<tr>
<td>Pascal3D+</td>
<td>75.99%</td>
<td>87.46%</td>
<td>93.4%</td>
</tr>
<tr>
<td>Sequence 1 (Ours)</td>
<td>21.11%</td>
<td>66.18%</td>
<td>95.93%</td>
</tr>
</tbody>
</table>

Figure 3.9: Progress of training the hourglass CNN using the proposed render pipeline. Loss per epoch when training on synthetic data followed by finetuning on a mix of real and synthetic images is illustrated here.
Figure 3.10: Qualitative Results showing the 2D keypoint localization performance of the proposed architecture [20]. Keypoints with more than 0.7 confidence scores output by the CNN are shown per instance. Discriminative features are extracted persistently across all poses. The last row shows some failed cases.

**Confidence Score vs. Keypoint Index**

The proposed CNN architecture for keypoint localization also gives a confidence score for each keypoint depending upon the probability of its visibility nature in the input image of the object. We tested the trained model on 2000 images, which includes both real and synthetic images. We did the analysis of keypoint visibility and took the average confidence score of each keypoint for all the testing images. The experiment indicates that the network has learnt the notion of visibility of keypoints along with the localization. Figure 3.11 shows the variation of confidence score for all the keypoints. Turns out, that its a bit high for the 9th and 10th index, which are easily visible points for both front and back facing chairs.

### 3.7 Generalization to Other Object Categories

We formulated the same pipeline of deep keypoint localization network to train for the Laptop category. With a fixed number of keypoints and normalized 3D wire frame, just like the chair category, Laptop is also suited for our Object SLAM frame work. The major issue with this category is, the availability of the real annotated dataset. To my knowledge, there is no keypoint-annotated real dataset for laptops. So we collected the real images of laptops from our laboratory and other sources, following is
Figure 3.11: Bar graph for confidence variation of each keypoint. Along with the order of keypoints. The manual annotation of them. And also due to the availability of the 3D CAD Models of laptops, we were able to use the render [29] pipeline to generate the synthetic set. Fig. 3.12 shows a few qualitative results of keypoint localization on Laptops.

Figure 3.12: Some successful cases for keypoint localization on Laptops.
Chapter 4

Monocular Object-SLAM

In this chapter, we describe how the category-models thus learned can be incorporated into a monocular SLAM backend.

4.1 Object Measurement Factors Using Category-Models

We first introduce the notation we use throughout the section. $T_{ij} \in SE(3)$ denotes a rigid-body transform that takes a 3D point expressed in the camera frame at time $i$ and expresses it with respect to the camera frame at time $j$. Hence, $T_{ij} = T_j T_i^{-1}$ ($T_\ast$ denotes a transformation matrix that transforms points from frame $\ast$ to the world frame $W$). Each such matrix $T_{ij}$ is a $4 \times 4$ matrix and takes the following form.

$$T_{ij} = \begin{bmatrix} R_{ij} \\ t_{ij} \\ 0 \\ 1 \end{bmatrix} \text{ where } R_{ij} \in SO(3), t_{ij} \in \mathbb{R}^3$$

Each camera frame $i$ is related to camera frame $j$ in the following manner. If $^{w}X$ denotes the 3D coordinates of a world point $^{w}X$ in the $i$th camera frame, $^{j}X$ is given by $^{j}X = T_{ij}^{1}X$. We use $\pi : \mathbb{R}^4 \mapsto \mathbb{R}^2$ that projects 3D homogeneous coordinates to 2D Euclidean coordinates. We subsume the camera intrinsics $f_x, f_y, c_x, c_y$ into the function $\pi$.

$$\pi \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f_x X}{Z} + c_x \\ \frac{f_y Y}{Z} + c_y \end{bmatrix}$$

We use $Log(.)$ to denote the logarithm map that maps an element of the group $SE(3)$ to a corresponding element in the exponential coordinates of its tangent space $se(3)$. 

24
4.1.1 3D Pose-SLAM

We formally define the 3D pose-SLAM problem as follows. Given a set $\mathcal{Z}$ of relative pose measurements $\hat{T}_{ij}$ among robot poses $i,j$, where $i,j \in \{1..N\}$, estimate $T_i$ for all $i \in \{1..N\}$ such that the log-likelihood of the observations (relative pose measurements) is maximized. This reduces to the problem of minimizing the observation errors (minimizing the negative log likelihood). We assume that each relative-pose measurement $\hat{T}_{ij}$ has an associated uncertainty $\Sigma_{ij}$.

$$\min_{T_i, i \in \{1..N\}} \mathcal{E}_{pose} = \sum_{\hat{T}_{ij} \in \mathcal{Z}} \| \log(\hat{T}_{ij}^{-1} T_j T_i^{-1}) \| \Sigma_{ij}$$ (4.3)

One popular approach of minimizing the error in (4.3) is the use of factor graphs [14, 9]. We adopt this approach since it naturally extends to an incremental optimization framework that enables online processing.

4.1.2 Object Observation Factors

Using the linear subspace model illustrated in (2.1), we now design object observation factors that are suitable for instance-independent monocular object-SLAM. Given 2D locations of object keypoints $s$, we use the pose and shape adjustment scheme proposed in [18] to lift the 2D keypoints to 3D, thereby estimating the shape and pose of the object from just a single image.

We assume that, from the $i^{th}$ pose, the robot observes $M$ objects ($M$ could also be zero). The set of object observations in frame $i$ is denoted $O_i$, and each observation in the set is indexed as $\hat{T}^O_{mi}$. We denote the number of keypoints of the object category by $K$. The $k^{th}$ keypoint of the $m^{th}$ object observed are denoted by $s^m_k$. The pose and shape of each object can then be computed by minimizing the following keypoint reprojection error (see [18] for details).

$$\min_{\hat{T}^O_{mi}, \Lambda_m} \left( \sum_{m=1}^{M} \sum_{k=1}^{K} \| \pi(K \hat{T}^O_{mi}(S + V \Lambda_m)) - s^m_k \|_2^2 \right) + \rho(\Lambda_m)$$ (4.4)

In the above equation, $\rho(\Lambda_m)$ denotes an appropriate regularizers (such as an $L2$ norm, for instance) to prevent shape parameter ($\Lambda$) estimates from deviating too much from the category-model.

We estimate each object’s shape ($\Lambda$) and pose ($\hat{T}^O_{mi}$) in each frame $i$ by alternating minimization of the error term in (4.4), with respect to the pose and shape parameters. If the same object has been associated successfully across multiple frames, we also exploit temporal consistency [19] for more precise estimates.

4.1.3 Object-SLAM with Category-Level Models (Proposed Approach)

The object pose observations arising from (4.4) now form additional factors in the SLAM factor graph. If $\hat{T}^O_{mi}$ denotes the pose of object $m$ with respect to the $i^{th}$ camera frame, for each object node in the factor graph, the following pose error is to be minimized.
\[
\min_{T_i,\tilde{T}_m} \mathcal{E}_{obj} = \sum_{i=1}^{N} \sum_{m=1}^{M} \| \log( (T^O_{1m})^{-1} T_i^{-1} T^O_{\phi(m)} ) \| (4.5)
\]

Here, \( T^O_{\phi(m)} \) denotes the pose of object \( O_{\phi(m)} \) with respect to the global coordinate frame, where \( \phi(m) \) denotes the data association function which assigns a globally unique identifier to each distinct object observed so far. The data association pipeline is briefly outlined in Section 4.2.

Finally, the object-SLAM error that jointly estimates robot poses as well as object poses from relative pose observations is given by

\[
\mathcal{E} = \mathcal{E}_{pose} + \mathcal{E}_{obj} \quad (4.6)
\]

### 4.2 Implementation Details

We use the publicly available GTSAM \[^{[9]}\] framework for constructing and optimizing the proposed factor graph model. Robot pose observations are obtained from a visual odometry/SLAM approach. We use odometry information published by ORB-SLAM \[^{[17]}\] for this purpose. However, monocular SLAM inherently suffers from scale ambiguity, i.e., the absolute scale of reconstruction cannot usually be recovered. To recover actual scale, we use information from a range sensor to estimate the height of the drone (camera) above the ground. Thereby, using objects that lie on the ground plane (chairs, for instance), we can compute the absolute scale factor by backprojection via the ground plane \[^{[18, 28]}\]. This height incorporated into ORB-SLAM estimates to scale the trajectory estimates and the map so that everything ends up being consistent. Moreover, ShapeNet \[^{[5]}\] CAD models for the categories we use are available in metric scale. Hence, a rough prior on object dimensions is available and can be used to initialize the solution for faster convergence \[^{[18]}\].

#### 4.2.1 Object Detection and Data Association

We use the YOLO9000 \[^{[23]}\] object detector to detect objects of interest and perform non-maximum suppression. Each bounding box detection is then passed to the keypoint localization network. We could have used more accurate detectors like \[^{[13]}\] but it would cost us at run time which is not in the case of YOLO9000.

Between successive frames, objects are tracked using a greedy tracker based on the Hungarian algorithm. For long-term tracking and for Object Loop Closure detection (OLC), we use the estimated shape and pose parameters as costs in the Hungarian algorithm. This has shown to be an extremely simple, yet effective data association method in \[^{[27]}\].

The object shape and pose are optimized using Ceres solver \[^{[1]}\] and are cast as observations into the factor graph. The factor graph can be optimized in incremental or batch mode, and with or without object loop closures.
4.2.2 Object Initialization in 3D

![Diagram of 3D object initialization](image)

Figure 4.1: Initialization of 3D object from 2D monocular image.

\[ X = \frac{h.K^{-1}.x}{n^T.K^{-1}.x} \]  

(4.7)

Fig. 4.1 demonstrates the 3D initialization of the object, calculation of the center co-ordinate of the wireframe and the estimated distance of the object, in a 3D world. Eqn. 4.7 calculates the same. In this equation, \( h \) is the height of the drone from the ground estimated by the on-board sensors. \( K \) is the camera matrix, \( n \) is normal to the ground plane, \( x \) is the location of the point in a 2D image. And \( X \) is the same point projected in 3D. Using this information and the offsets, center 3D co-ordinate of the wireframe is calculated. (Note: Object is on the ground and YZ plane of the co-ordinate axis of drone is parallel to the ground plane.)

4.3 Results

In this section, we present experimental results upon evaluating various modes of operation of the proposed object-SLAM approach and the render pipeline on multiple real-world sequences.

4.3.1 Object-SLAM Evaluation

By explicitly modeling objects in a SLAM framework, we achieve substantially lower object localization errors over feature-based SLAM [17]. Also, we highlight the use of discriminatively learned features that enable us to obtain trajectory and object pose estimates even in scenarios where monocular SLAM approaches fail, such as strong rotational motion.

4.3.2 Dataset

To demonstrate object SLAM in a real-world setting, we collected a dataset comprising of 4 monocular video sequences from robots operating in office and laboratory environments. These sequences
were collected from a micro aerial vehicle (MAV) flying at a constant height above the floor. For one of the sequences, fiducial markers were placed in the environment and were used to estimate ground truth camera pose and object locations. For two other sequences, no markers were placed in the environment, to deny any undue advantage enjoyed by monocular SLAM. In these sequences, all object locations were measured a priori and this is used in evaluating the object localization accuracy. There was a rotation-only run, where the MAV rotates more-or-less in-place. In all these runs, we evaluate object localization precision for ORB-SLAM [17] and for various flavors of the proposed object-SLAM approach.

A brief description of the dataset is as follows.

- **Sequence 1**: An elongated loop with two parallel sides exhibiting dominant straight motion (7.6 meters) (fiducial markers were used for acquiring ground-truth).
- **Sequence 2**: Smaller run with smoother turns (no fiducial markers).
- **Sequence 3**: 360° Rotation in place without any translation from origin.
- **Sequence 4**: 2 meters forward, sharp 90° rotation towards left, 4.2 meters forward

In all these sequences we consider chairs as the objects of interest. For a fair evaluation, and to demonstrate the efficacy of the proposed object-SLAM system, we ensure that most of the frames in the sequence contain various chair instances.

Table 4.1: Quantitative analysis of several variants of the proposed object-SLAM approach. Here, the best and worst object localization errors refer to the errors in the most accurately localized and the most erroneously localized object respectively. We also show the average object localization error and accumulated endpoint drifts, wherever applicable.
4.3.3 Discussion

On the collected dataset, we evaluate the several variants of the proposed object-SLAM approach, viz. batch mode (the factor graph is constructed for the entire scene and is optimized in one go), incremental mode (observations are processed as they arrive). We also evaluate object-SLAM with and without object-based loop closures. Table 4.1 summarizes the results of these experiments.

For each of the approaches, we evaluate object localization error (in meters), and also the accumulated drift in the trajectory in the X and Z directions. The accumulated drift is computed only in cases where the robot returns to the starting position at the end of the run. Significant drifts in the Z-direction occur in sequence 1 for ORB-SLAM [17], but these are corrected by the object-SLAM system, regardless of the presence or absence of object loop closures. In sequence 4, there is a strong rotation (a perpendicular turn), where ORB-SLAM performs poorly.

Our discriminative features enjoy an advantage over tracked features [17] in that they need to be visible in just a single image to enable object measurements. We demonstrate this in sequence 3 (cf. Fig. 4.5). In this sequence the robot purely rotates about its origin. While ORB-SLAM [17] fails to initialize for this sequence, the object-SLAM proposed here is able to accurately localize all chairs in the scene. The circle traced out by our object-SLAM systems has a radius roughly equal to that of the drone used in the run.

The incremental version performs competitively, and in cases, better, than the batch-mode object-SLAM. Loop closures based on object observations further boost the accuracy of the proposed approach. For batch mode, we perform a chordal initialization before optimizing the factor graph.

Fig 4.2, 4.3, 4.4 show plots of the trajectories estimated by ORB-SLAM [17], as well as from object-SLAM. We also show object location estimates overlaid on ground-truth object locations for the sequence.
Figure 4.3: Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM with/without **Object Loop Closure detection (OLC)**

Figure 4.4: Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM with/without **Object Loop Closure detection (OLC)**

Sequence. Sequences 1 and 2 are plotted with and without incorporating object loop closure constraints, to show that object loop closure further reduces endpoint drift.
Figure 4.5: Rendering objects estimated in a run where the robot rotates in place. ORB-SLAM [17] fails to initialize due to insufficient parallax.

Figure 4.6: Qualitative results on another object category (laptops).
Chapter 5

Extensions: Handling severely occluded and truncated objects

Our approach of category-specific Object SLAM, involves object detection and keypoint localization of the object retrieved from the scene as pre-processing steps. There are several technical and visual challenges to this. Here, in this chapter, we have discussed and proposed to tackle, the constraints of the object visibility in the scene. If any object part is obstructed, it makes very difficult for the keypoint localization network to consistently predict the point’s region for that portion of the object with a considerable confidence. Consequently, it leads to the poor construction of the category model and hence the performance of shape and pose adjuster deteriorate. In the case when the visibility of the object in the scene is compromised, it can be difficult to assess the performance of our pipeline.

Predominantly, visual constraints, exhibited by objects are, (1) Occlusion: An object marked as occluded indicates that a significant portion of the object is blocked within the bounding box. The portion, either is blocked by another object that is mutual-occlusion or self-occlusion, in which one part of the object is blocking the other part, as can be seen in Figure. 5.1. In this case the part is not visible but its position is apparent given the context of the image. and (2) Truncation: an object marked as truncated indicates that the bounding box specified for the object does not correspond to the full extent of the object, example shown in the Figure 5.1 below. In this one we don’t have any prior information about the position of the part in the image. These visual constraints are subjective to how the object detector, like YOLO9000 in our case, works for a particular instance.

In the Fig. 5.1 we have demonstrated some of the instances of the object explaining the constraints. Case A, the base case, shows the object with all the keypoints visible, easy to handle. In case B, the
object is truncated, all 4 legs are out of the bounding box. Case C, D and E show the types of object occlusions. In Case C, self-occlusion of the object can be seen, the right-back leg is occluded by the left-front leg. In case D, our object is occluded by another object, explaining the situation of mutual occlusion and finally in Case E scenario, we have multiple objects in the image. Figure 5.2 shows the failure of our keypoint localization network for all these cases apart from the base case that is Case A where all the keypoints are in the sight. Consequently these all instances lead to the formation of distorted wire frame, hence these instances are useless for our SLAM framework.

![Figure 5.2: Keypoint Localization for the constrained cases.](image)

From the results above, we inferred that the improvement in the keypoint localization network for such instances will do the trick and hence we exploited the nature of our network architecture and modified our hourglass network [20] accordingly. As explained in chapter 3, in the stacked hourglass network, the features are processed across all scales and consolidated to capture the various spatial relationship associated with the object and so it can also handle such cases provided, the training dataset includes such instances. We tackled this problem by altering the training dataset for both, truncation and occlusion cases. If there is no ground truth annotation provided for a keypoint it is impossible to assess the quality of the prediction made by the system.

5.1 Handling Occlusion

Initially, in the real set from PASCAL 3D+ [38] which we trained our network on, includes only those keypoints which has its visibility check as “1”. Adopting [40], we pre-processed our training dataset and provided all the information regarding the location of each keypoint. We manually annotated all the keypoints, even if its visibility check is “0”. Occlusion is clearly a significant challenge, but the network still makes strong estimates in most cases. In many examples, the network prediction and ground-truth annotation may not agree while both residing in valid locations, and the ambiguity of the image means there is no way to determine which one is truly correct.
5.2 Handling Truncation

We included the instances, involving the truncation of the object with the information about the location of the keypoint which got truncated, in our training dataset. We cropped the images and padded it from all around as shown in the Figure below. We did this for over a million set.

Figure 5.4: Creation of the truncated instances

5.3 Results

Usually our discriminative features enjoy an advantage over tracked features [17] in that they need to be visible in just a single image to enable object measurements but in the cases we are dealing with, their visibility is compromised. Still, the changes we made in our training pipeline for handling these constraints, somehow generated the promising outcome. Here, we present experimental result of our proposed object SLAM approach in a real-world setting as done in the Chapter.

Discussion

To start with, in Fig. 5.5 we have shown some of the best Qualitative Results of 2D keypoint localization on real set along with some failed cases. Now, for the whole pipeline of our proposed Object SLAM, we collected a monocular video sequence, from a micro aerial vehicle (MAV), containing all
the visibility constraints mentioned above. Cases discussed above in Fig. 5.1 are some of the instances from the video sequence. Our sequence contains 7 objects from the category. We followed the same pipeline for this run as done for all the other sequences mentioned in Section 5.2 of Chapter. Apart from the keypoint localization part. After passing the whole sequence through the YOLO9000 object detector and ID-ing all the chairs using the tracking technique used, we cropped the detected objects and then padded it so as to prepare the input set for the trained network for truncation and occlusion cases. We also present some of the best qualitative results for some individual cases retrieved from the sequence in Fig. 5.6.

Fig. 5.7 demonstrates the quantitative analysis of the detected keypoints under these visibly challenged instances. A bar graph showing the comparison of the confidence score for each keypoint of our new sequence. It has showed significantly better results for our modified model as compared to the previous model. We did an average analysis for both the truncation and occlusion instances.

After the analysis of output from Fig. 5.5 and Fig. 5.6, we can infer that all the instances of the visible constraints are looked into but the instances having self-occlusion and truncation are handled very well by the network. The last frame from Fig. 5.6 where an object is getting truncated, initially
due to the wrong localization of the truncated keypoints, the fitting wireframe lifts from the ground, consequently lifting the trajectory of the proposed SLAM but as we can see that is not the case here. The frames involving multiple objects - mutual-occlusion have still some issues and can be improved.

Fig. 5.7 shows a plot of the trajectory estimated by ORB-SLAM [17], as well as from object-SLAM. We also show object location estimates overlaid on ground-truth object locations for the sequence. The performance is not as good as that of the other sequences but is able to handle the visual constraints.

Fig. 5.8: Estimated Trajectories and Object Locations by ORB-SLAM and Object-SLAM without Object Loop Closure detection (OLC) for our new sequence.
Chapter 6

Conclusions

In this thesis, we initially described the category-level models that we employ in our object-SLAM system and construction of those models. Exploiting the idea that the shapes of objects from the same category are not arbitrary, we displayed an approach for assessing the 3D shape and pose of an object from a solitary image using the shape priors. We solved for both the shape and pose optimisation separately and conducted a lot of experiments demonstrating a very good performance.

All things considered, we feel that the bottleneck to accomplishing close immaculate performance is uncertain keypoint localization. Hence we adopted the CRF-style stacked hourglass architecture and modified it for our object category. We then made use of the render pipeline and reformatted it to generate the annotated 2D synthetic images from the annotated 3D CAD models. We achieved state-of-the-art accuracy for the keypoint localization of our object category over the PASCAL 3D+ real set and the generated synthetic set. This provided a better initialisation.

Finally, we have presented an approach for monocular object-SLAM that relies on category-specific models, rather than relying on instance-specific models or generic models. This is a new paradigm in the nascent field of object-SLAM research. Also, the proposed category-models help in object instance retrieval, useful for Augmented Reality (AR) applications. We evaluate the proposed framework on multiple challenging real-world scenes and show — to the best of our knowledge — first results of an instance-independent monocular object-SLAM system and the benefits it enjoys over feature-based SLAM methods. We made some changes in the formulated pipeline and tried to remove the scene visibility constraints from the sequences such as occlusion and truncation of the objects.

The proposed category-specific models can be adopted to many other such rigid objects, as long as data for the corresponding category is available in the form of aligned CAD models. And this again is counted as one of the shortcomings of the proposed approach. Our keypoint localization network is subjective to the object category. So for different object category, having different keypoints and the varied number of keypoints, involves repeating the data pre-processing and the training, which is a time taking process. Another shortcoming is to deal with the mutual-occlusion in an instance.

Future work could focus on further reducing the extent of supervision required to learn such category-specific models and to modify the pipeline for the visual constraints.
Related Publications


Other Publications


Bibliography


