Systematic Generation of Datasets and Benchmarks for Modern Computer Vision

by

Sri Raghu Malireddi
B.Tech., Indian Institute of Technology Gandhinagar, 2016

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Computer Science
University of Victoria

Systematic Generation of Datasets and Benchmarks for Modern Computer Vision

by

Sri Raghu Malireddi
B.Tech., Indian Institute of Technology Gandhinagar, 2016

Supervisory Committee

Dr. Kwang Moo Yi, Supervisor
(Department of Computer Science)

Dr. George Tzanetakis, Departmental Member
(Department of Computer Science)

Supervisory Committee

Dr. Kwang Moo Yi, Supervisor
(Department of Computer Science)

Dr. George Tzanetakis, Departmental Member
(Department of Computer Science)

ABSTRACT

Deep Learning is dominant in the field of computer vision, thanks to its high performance. This high performance is driven by large annotated datasets and proper evaluation benchmarks. However, two important areas in computer vision, depth-based hand segmentation, and local features, respectively lack a large well-annotated dataset and a benchmark protocol that properly demonstrates its practical performance. Therefore, in this thesis, we focus on these two problems. For hand segmentation, we create a novel systematic way to easily create automatic semantic segmentation annotations for large datasets. We achieved this with the help of traditional computer vision techniques and minimal hardware setup of one RGB-D camera and two distinctly colored skin-tight gloves. Our method allows easy creation of large-scale datasets with high annotation quality. For local features, we create a new modern benchmark, that reveals their different aspects. Specifically wide-baseline stereo matching and Multi-View Stereo (MVS), of keypoints in a more practical setup, namely Structure-from-Motion (SfM). We believe that through our new benchmark, we will be able to spur research on learned local features to a more practical direction. In this respect, the benchmark developed for the thesis will be used to host a challenge on local features.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisor Prof. Dr. Kwang Moo Yi for the continuous support of my M.Sc study and related research, for his patience, motivation and immense knowledge. His guidance helped me in all the time of research and writing this thesis. I could not have imagined having a better advisor and mentor for my M.Sc study.

Besides my advisor, I would like to thank the rest of the thesis committee: Prof. Dr. George Tzanetakis, and Prof. Dr. T. Aaron Gulliver, for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

My sincere thanks also goes to Dr. Andrea Tagliasacchi, Dr. Eduard Trulls who provided me an opportunity to work with them during my M.Sc studies with access to laboratory and research facilities. I would also like to thank Microsoft for providing me an opportunity to work as an intern in the field of Computer Vision where I got hands-on experience to push computer vision research algorithms to production.

I thank my fellow labmates for the stimulating discussions, and for all the fun we have had in the last two years. Also, I thank my friends in the University of Victoria. In particular, I am grateful to Mohan Noolu and Abhishake Kumar Bojja for filling my life outside lab and made me feel like home away from home.

Last but not the least, I would like to thank my family: my parents and my brother for supporting me spiritually throughout writing this thesis and my life in general.

DEDICATION

This thesis is dedicated to all my well-wishers - parents, friends and my mentor.

Chapter 1

Introduction

Recent advances in modern computer vision is based highly on datasets [95] and good evaluation benchmarks [24, 47, 79]. This is especially true in the era of deep learning, where most state-of-the-art methods in computer vision rely on deep learning [48, 53, 72, 18]. Deep learning is a fragment of a broader family of machine learning for learning data representations. Some deep learning architectures such as deep neural networks, recurrent neural networks, deep belief networks have been applied to the field of computer vision [48, 32, 50]. For successful deep learning in modern computer vision, it is a common practice to have: (i) high computational power to train neural networks; (ii) complex and large deep network architectures; and (iii) labeled data to train the neural networks [95]. With advances in modern GPU hardware, cloud compute and open-source deep learning libraries such as TensorFlow [2], Caffe [42], and PyTorch [68], training deep neural networks has become easy. One of the challenges for training deep neural networks is now the creation of large datasets with high qualitiy annotations, as well as having a proper benchmark setup.

In this thesis, we focus on mainly two application areas of computer vision. In the first part of the thesis, we will focus on depth-based hand segmentation, and propose a systematic way of generating large high-quality datasets without much human labeling effort. In the second part, we will focus on benchmarking of local features, where we propose a benchmark framework that is well-tied with how local features are used in practice.

1.1 Dataset for Hand Segmentation

In everyday life, we interact with our surrounding environment utilizing our hands as analog controllers. To have a fully immersive experience in virtual environments, there is a need to transfer this natural way of hand interaction. Hence, the development of robust hand tracking technology becomes an essential requirement for the success of immersive AR/VR experiences. Thanks to depth cameras, substantial progress towards this goal has been made, where state of the art constitutes a mixture of generative and discriminative methods to fulfill the objectives of efficiently and accurately tracking various poses of hands and efficient re-initialization in cases of a tracking failure. Most real-time tracking algorithms depend on the pre-processing step of identifying the location of hands in the image. In the case of generative trackers, we need to identify the subset of a point cloud to which a known digital model has to be aligned. Discriminative trackers assume the input to the regressor to be a rectangular region of fixed size, with the hand roughly centered. Rather than addressing the generic problem of hand tracking, we focus our attention to the issue of hand segmentation, as a robust solution to this first step is essential to enable accurate hand tracking.

Some heuristic solutions have been proposed to simplify the task of hand segmentation [66, 56, 64, 96, 87]. While these approaches are sufficiently suited for small-scale lab experiments, they don't possess the robustness required for any consumer-level solution. A consumer-level solution requires a robust hand tracker working under the full diversity of interactions in real-world scenes. Any violation of the underlying assumptions for hand localization will result in an immediate tracking failure. One could train a hand segmentation model from a dataset of color/depth images with hand annotations as labels. The lack of quality and limited size of currently available datasets result in regressors that generally overfit to the training data. Overfitting of models leads to poor generalization to unseen scenarios. Further, in contrast to marker-based hand tracking, the limited size of available hand segmentation datasets had primarily led to the insufficient attention for applying modern deep learning solutions to the problem of real-time hand segmentation. Hence, a central challenge is to generate a sufficiently large dataset equipped with high-quality ground truth annotations.

Our primary contribution is our method to systematically create a depth-based hand segmentation dataset, as well as the created dataset itself. We acquire our

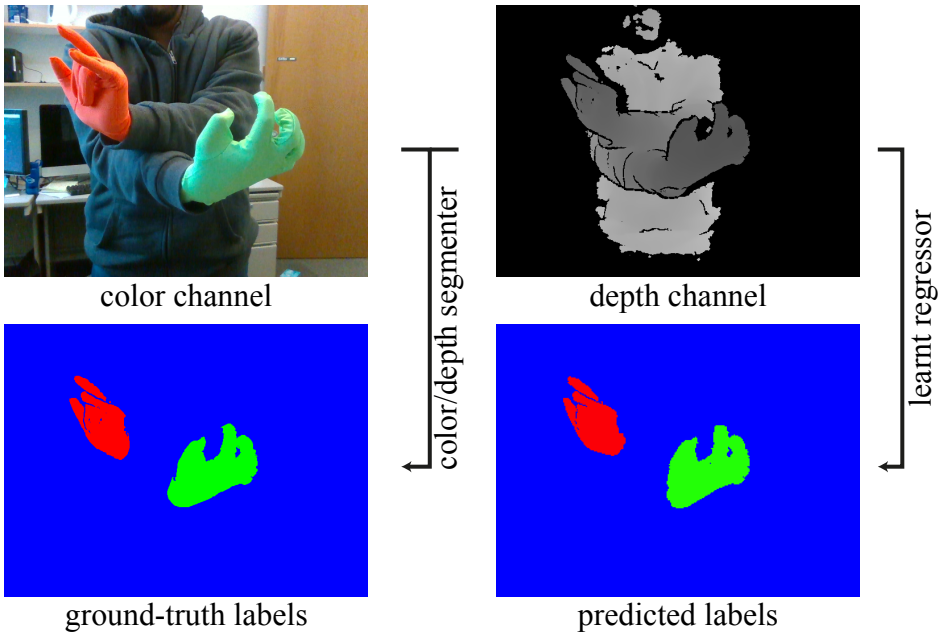

Figure 1.1: Our hand segmentation dataset is created by having a number of subjects performing in front of the RGB-D camera while wearing a pair of colored gloves. Color and depth are then jointly exploited to compute ground-truth labeling without any user intervention automatically. The mapping between input depth and ground truth labels is then exploited to learn a hand segmentation network for depth input.

dataset by having a number of users perform hand gestures in front of an RGB-D camera while wearing a pair of colored gloves; see Figure 1.1. The synchronized color and depth channels are then used to generate high-quality ground truth annotations with minimal manual intervention. This process allowed us to generate high-quality annotated hand segmentation dataset which is two orders of magnitude larger than what is available in the literature. We used this preliminary dataset to train some existing deep learning networks to perform real-time hand segmentation. We will dicsuss our method and dataset in more detail in Chapter 3.

1.2 Evaluation Benchmark for Local Features

Local features have played a vital role in a wide range of computer vision applications throughout the past 20 years, particularly since the inception of Scale Invariant Feature Transform (SIFT) [55, 85]. Despite the drastic advancements resulting from deep learning techniques, 3D reconstruction under challenging conditions remains somewhat of an outlier, as performance in small, constrained benchmarks does not

necessarily translate to real-world scenarios [85]. In practice, keypoint-based methods remain the most common solution to this problem, and techniques such as SIFT [55] or RANSAC [30] are still very much in use.

Historically, machine learning research on local features has focused on learning patch descriptors, for which training data is relatively easy to obtain [16, 91]. However, performance on patch matching benchmarks is not always meaningful, as descriptors are tightly coupled with the keypoints they work on and image properties which can significantly vary from one domain or dataset to another [109, 85, 67]. More representative metrics can be extracted further down the chain, for instance at the 3D reconstruction level, but this requires better ground truth. Therefore, instead, we provide the 3D reconstructions of larger datasets, which can be used as ground truth for evaluating the performance of local features over small bags of images.

In parallel, there has been a strong push over the last few years towards tackling the image matching problem with dense methods, that is, doing away with keypoints altogether [20, 101, 103, 113, 118]. While promising results have been demonstrated under narrow baselines, particularly with dense, deep networks, the general wide-baseline scenario remains unsolved [110, 117]. Moreover, these methods still suffer the problem of not being properly evaluated against more traditional baselines such as COLMAP [83, 84], which is mainly due to the fact that such an evaluation framework is not easy to develop.

Therefore, to enable research in this area, we create a large-scale benchmark for evaluating local features that incorporate various aspects. We go beyond the current datasets and benchmarks [58, 57, 16, 36, 85] that are constrained in terms of size, photometric variations, and viewpoint changes. We will discuss our new framework in more detail in Chapter 4.

1.3 Key contributions

In summary, the key contributions of this thesis are two fold.

- **Automatic dataset labeling for hand segmentation:** To allow deep learning to reach its full potential on depth-based hand segmentation, we propose a method to automate the task of dataset labeling for hand segmentation to ease up the process of annotating large datasets to train machine learning models.

- **Evaluation benchmark for local features:** To properly understand the per-

formance of local features under challenging conditions, we propose a new evaluation benchmark using stereo matching and multi-view reconstruction based metrics.

1.4 Overview

The rest of this thesis document is organized as follows:

Chapter 1 gives a brief introduction about the need for large datasets and new evaluation pipelines for modern computer vision. We particularly focus on the case studies of hand segmentation and Structure-from-Motion.

Chapter 2 gives a brief overview of the previous research that is done in the area of dataset generation for hand segmentation and current trends in evaluating local features for Structure-from-Motion.

Chapter 3 briefly describes the proposed method for automating the labeling of hand segmentation. We will also discuss the various semantic segmentation networks and their performance when trained on the dataset we created.

Chapter 4 discusses the new evaluation benchmark pipeline, the SDK documentation, and results on some local features with the new evaluation benchmark.

Chapter 5 provides some final remarks.

Source code, evaluation benchmark and datasets will be made publicly available.

Chapter 2

Related Work

2.1 Datasets and Benchmarks in Deep Learning

Throughout the history of Computer Vision research, datasets and benchmarks have played a critical role. They not only provide means to train and evaluate algorithms, but also push research into challenging directions. The introduction of stereo and optical flow datasets with ground-truth [78, 8] has driven the interest in these areas by the research community. The early evolution of object recognition datasets [29, 33, 22] provided a platform to directly compare hundreds of image recognition algorithms while simultaneously pushing the field towards more complex problems. The current revolution of convolution neural networks and deep learning is a product of large magnitudes of labeled datasets [95], especially millions of labeled images from ImageNet [24]. Here, we first discuss three landmark datasets.

ImageNet. This is a dataset with millions of labeled images based on 1000 categories. It was one of the first large scale datasets to target the image recognition task. Since its inception, many algorithms were proposed for visual recognition but AlexNet [48], a convolution neural network based algorithm, has become the winner for the first challenge. Since then, many different convolution neural network architectures have been proposed which led to greater accuracy in visual recognition. To collect highly accurate dataset, ImageNet [24] relied on a two-step process. It first collected candidate images from the internet by querying from several image search engines. These images along with queries are verified with the help of humans. This was achieved with the help of Amazon Mechanical Turk (AMT), an online platform on

which one can host tasks for users for a monetary reward. The images in the dataset also have some occlusions, many objects, and scene clutter to ensure diversity. For quality control, ImageNet [24] asked multiple users to label the same image and chose the best label by the highest vote metric. The models trained with ImageNet [24] have become popular for many things even outside of classification. For example, R-CNN [31] uses AlexNet [48], trained on ImageNet [24], to extract the features from the input image to perform object localization.

PASCAL VOC. For the detection of some basic object categories, several years (2005-2012) were devoted to creating benchmark datasets which were widely adopted. The PASCAL VOC [28] datasets contain over 11,000 images with 20 object categories. It has nearly 7,000 detailed object segmentation annotations and over 27,000 labeled object instances with bounding box information. Pascal VOC [27] also hosted a object detection challenge with nearly 200 object categories using a subset of 400,000 images from ImageNet [24] summing up to an impressive 350,000 labelled objects with bounding boxes. Pascal VOC [28] also created proper evaluation benchmarks for their challenges, and they proposed to use bootstrap sampling to identify clear winner among entrants with similar scores [105]. Pascal VOC [27] has become a standard for object recognition related tasks with over $c.a.5.5k$ citations and even today many new proposed approaches have to compare against the Pascal VOC's benchmark.

MS-COCO. The Microsoft Common Objects in COntext (MS-COCO) dataset contains 91 common object categories with 82 of them having more than 5000 labeled instances. The entire dataset has 328,000 images with 2,500,000 labeled instances. In contrast to the ImageNet [24], COCO has fewer categories but more instances per category. MS-COCO [52] has targeted the three categories of object recognition and crowd-sourced the dataset labeling tasks to Amazon's Mechanical Turk (AMT). Segmenting 2,500,000 object instances is very time consuming and it requires 22 worker hours for 1,000 segmentation tasks [52]. This sums to a total of 55,000 worker hours. Also, the task of category labeling took a total of *c.a.* 20,000 worker hours. From the above statistics, it is clear that creating a dataset for semantic scene labeling is costly and time-consuming. MS-COCO [52] has not only provided datasets for various challenges in object recognition, but also provided evaluation benchmarks for the same. It has over $c.a.4,000$ citations since its inception and has become standard

evaluation for various challenging problems in Computer Vision.

Object recognition datasets. Datasets related to object recognition can be broadly classified under three categories: object classification, object detection, and semantic scene labeling. The object classification task deals with predicting the class of the dominant object in an image. ImageNet [24] dataset falls under this category of object classification. Object detection solves the problem of predicting the class of the object and localizing it in the image. The location of the object is represented using a rectangular bounding box. Early algorithms in object detection dealt with face detection [38] using various algorithms. Later, more challenging face detection datasets were created [40]. Another popular object detection challenge is detecting pedestrians for which Caltech Pedestrian Dataset [26] with 350,000 labeled bounding box instances has been proposed.

Scene understanding datasets. While object detection deals with labeling the objects in a scene with bounding boxes, semantic scene labeling categorizes each pixel in an image to its relevant object class. Some datasets, with categories such as glass, streets, and walls, etc., exist for both indoor [90] and outdoor [88, 15] scenes and some datasets also included depth information [90]. The goal of semantic scene labeling is to measure the pixel-wise accuracy of object labels.

Other vision datasets. Datasets have driven the advancement in numerous subfields of Computer Vision. Apart from above mentioned datasets, there are some notable datasets such as Middlebury datasets for stereo vision [78], multi-view stereo [86], and optical flow [8]. The Berkely Segmentation Data Set (BSDS500) [5] had been used extensively to evaluate semantic segmentation and edge detection. Numerous areas of computer vision have benefited from challenging datasets and proper evaluation benchmarks. In the following sections, we discuss existing datasets for one of the areas of computer vision, hand segmentation, and the current evaluation benchmarks for local descriptors.

2.2 Datasets for Hand Segmentation

Due to the limited size of available datasets for hand segmentation, the application of modern deep learning algorithms to the problem of real-time hand segmentation has

received less attention. Probably as a result, while there is significant progress in the area of hand pose estimation [45, 51, 61, 65], hand segmentation, the preliminary step for localizing hands to estimate hand pose has received very little attention. To alleviate this problem, few heuristics have been proposed for real-time hand segmentation dataset collection.

2.2.1 Heuristics for Data Collection

Skin color segmentation. The pioneering approach by Oikonomidis et al. [66] leverages skin color segmentation, proposed by Argyros and Lourakis [6]. In this method, a small input of training images is selected. The images used are in YUV 4:2:2 format. Since the Y-component of the YUV representation is sensitive to illumination, it is removed. The resulting U, V channels are used to compute (i) the prior probability $P(s)$ of skin color, (ii) the prior probability $P(c)$ of the occurrence of each color c in the training set and (iii) the prior probability $P(c|s)$ of a color c being a skin colors can be computed by employing the Bayes rule:

$$P(s|c) = P(c|s)\frac{P(s)}{P(c)} \tag{2.1}$$

All the image points with probability $P(s|c) > T_{max}$ are considered as being skin-colored. For hand segmentation, this approach of skin color segmentation expects users to wear long sleeves, and since face color will be close to skin color, it has to be out of sight of camera's field of view for successful hand segmentation.

Depth camera-based tracking. Melax et al. [56] exploited short-range depth sensors by assuming that everything within the camera field of view is to be tracked. Oberweger et al. [64] expects the closest point to the depth camera belongs to the hand. These approaches limit the usage of their proposed, though robust, algorithms.

ROI based methods. These methods identify the region-of-interest (ROI) as the portion of the point cloud where a hand is present with the highest probability. Tagliasacchi et al. [96] uses a wristband with known color to compute the ROI of hand. It uses a YUV based color segmentation to identify the wristband, followed by a vector projection to determine ROI containing the hand. The limitations with this kind of approach are that wristband color calibration has to be done: (i) every

time we change the wristband or (ii) environment lighting changes. Similarly, Sharp et al. [87] employed a machine learning based skeletal body tracking to query the wrist position. Though Sharp et al. [87] is robust to illumination changes, the algorithm for skeletal tracking has been trained over a full body or upper-body. Hence, it expects a significant portion of the human body in the field-of-view of the camera to predict the wrist.

These heuristics have been mainly proposed for hand pose estimation and tracking. Therefore, they are not directly suitable for acquiring high-quality annotations for hand segmentation. Moreover, these algorithms rely on assumptions that may not necessarily hold in practice. In other words, there is a clear need for creating high-quality hand segmentation dataset.

2.2.2 Existing Datasets

Buehler et al. [17] and Bambach et al. [11] proposed datasets for hand segmentation from color images. These datasets contain pixel-level manually annotated ground truth for respectively $c.a.500$ and $c.a.15k$ color images. Manually annotating segmentation masks from color images is extremely labor intensive task. This makes the task of collecting large-scale datasets with hand segmentation labels difficult. Also, the quality of annotations depends on the skills of individual annotators, for example, Amazon's Mechanical Turks. By contrast, annotating bounding-boxes is easier when compared to pixel-wise labels and the following datasets of $c.a.500$ annotated images in Everingham et al. [27], the $c.a.5k$ images in Laboratory for Intelligent and Safe Automobiles, UCSD [49], and the $c.a.15k$ images in Mittal et al. [59] targeted the problem of hand detection. However, the annotations of these datasets are too coarse which makes them unusable for applications which require accurate hand segmentation for real-time hand tracking.

Automatic hand segmentation. Hand segmentation can be interpreted as a skin color segmentation problem [120]. However, segmenting this way, detects not only hands but also other regions close to skin color, such as faces, and forearms when the user is not wearing sleeves, and background objects which are close to skin color. Further, datasets of this kind [23, 119, 43] contain at most a few thousand manual annotations, which is magnitudes smaller than what is needed to train deep neural networks. Zimmermann and Brox [120] recently proposed a dataset of $\approx 44k$ *synthetic*

images. However, it is notoriously challenging to accurately design a model to classify skin colors and other complex effects such as subsurface scattering. This makes it challenging to develop segmentation methods that could work in the wild. Conversely, annotating hand segmentations on *depth* images fused with color information do not suffer from this problem.

Foreground-Background Segmentation. In the area of visual effects, green screen matting has been heavily used for foreground-background subtraction in movie post-production to combine two completely different scenes. One of the possible applications of green screen matting in the area of modern computer vision is to produce a large amount of automatically labeled dataset for human segmentation. Once we receive the masks of the humans, some post-processing operations can be applied to replace the background and train deep learning algorithms accordingly. Though a similar approach can be implemented for hand segmentation, one of the significant challenges with this approach is the setup of a controlled space with lighting, and we also need to make sure that the subjects that are providing the dataset are not wearing any shades that are closer to green color.

Segmentation via tracking. Recent datasets targeting hands have mostly focused on acquiring annotated 3D marker locations for joints [112]. Creating datasets via manual annotation is not only labor-intensive [92] but placing markers within a noisy depth map often results in inaccurate labels. Assuming marker locations are correct, simple heuristics can be employed to infer dense labeling. Following this idea, Wetzler et al. [106] first employ a complex/invasive hardware setup comprising of magnetic sensors attached to fingertips to acquire their locations, and obtain the segmentation mask via a simple depth-based flood-fill algorithm. While the dataset by Wetzler et al. [106] contains $\approx 200k$ annotated exemplars, these heuristic annotations should not be considered to be ground truth for learning a high-performance segmenter.

2.3 Local Feature Benchmarks

Matching local image features is a very important step in many 3D computer vision applications such as Structure from Motion (SfM) and Multi-View Stereo (MVS) [3, 37, 71, 80, 81, 82], image retrieval [69, 76, 98, 100] and image-based localization [75, 77, 114]. In Computer Vision, local image features refer to specific structures

or patterns in the image such as corners, edges, blobs, and points. A descriptor is a highly distinctive vector representation of the local image feature which is invariant to scale, illumination and view-point.

Determining which local descriptors provide the best matching performance and better discriminative power is of significant interest to the computer vision community. Many benchmarks have been proposed for evaluating traditional hand-crafted local features [55, 12, 74] and recent learned local features [109, 85]. Here, we provide a brief overview of existing local feature benchmark frameworks.

2.3.1 Oxford Benchmark [58, 57]

Mikolajczyk et al. [58, 57], measured the performance of local features in terms of three main criteria: the repeatability rate, the matching score, and the descriptor Receiver Operating Characteristics (ROC). In this benchmark, Mikolajczyk et al. used a dataset with structured and textured scenes, and different types of transformations such as scale, illumination, and viewpoint changes, blur and jpeg compression [58, Fig. 9].

Repeatability Rate: The percentage of keypoints simultaneously present in two images is defined as the repeatability rate. It aims to measure how many feature points are repeatedly detected accross image pairs. A high repeatability rate between two images indicates that more local features' keypoints can be potentially matched between two images. The repeatability rate (or score) between two images is computed as the ratio between region-to-region correspondences and the minimum of number of regions in the pair of images. To compensate for differences in scale of regions between two images, a scale factor is determined and applied to transform the regions to normalized size before computing the overlap error. Overlap error is defined as the error in image area covered by both the regions. Two regions are considered as a correspondence if they have a low overlap error:

$$1 - \frac{R_{\boldsymbol{\mu}_a} \cap R_{(H^\top \boldsymbol{\mu}_b H)}}{R_{\boldsymbol{\mu}_a} \cup R_{(H^\top \boldsymbol{\mu}_b H)}} < \epsilon_0 \ , \tag{2.2}$$

where $R_{\boldsymbol{\mu}}$ represents the elliptic region defined by its elliptic parameters $\boldsymbol{\mu}$, that is, $\mathbf{x}^\top \mathrm{diag}\,(\boldsymbol{\mu})\,\mathbf{x} = 0$, where $\mathbf{x} = [u, v, 1]^\top$ and u, v are the image coordinates of the keypoints, and $H \in \mathbb{R}^3$ is the homography matrix defining the transformation

between two images. The union of two regions is given by $R_{\boldsymbol{\mu}_a} \cap R_{(H^T \boldsymbol{\mu}_b H)}$ and the intersection is given by $R_{\boldsymbol{\mu}_a} \cup R_{(H^T \boldsymbol{\mu}_b H)}$. The area of union and intersection of any two regions is computed numerically. ϵ_0 is a threshold that is typically set to 40% for computing repeatability [58].

The repeatability rate metric can be used to measure the robustness of detectors in geometric and photometric transformations such as changes in viewpoint, scale, and transformation [58].

Matching Score: For a local feature to be truly useful in a practical setup, it not only needs to be repeated, but also needs to be able to be matched. For example, if all local features are repeated, but they look exactly alike, it is impossible to distinguish among them and create correspondences across images. Matching score takes this into account. First, a putative set of potential matches is generated across local features detected in multiple images by looking at descriptors, typically by finding the local features with the descriptor that look most similar to each other in a nearest neighbor sense. The matching score is then computed as a ratio between the number of correct matches and the minimum of the number of detected regions in an image pair. The definition of correct matches follow the same formula as in Eq. 2.2, but when computing matching scores, the threshold is relaxed to be less strict, with a typical value of 0.5.

Descriptor Metrics: Later, Mikolajczyk and Schmid [57] used the Receiver Operating Characteristics (ROC) curve to evaluate the local feature descriptors' performance. The ROC curve is created by modifying the threshold for accepting two points across image as matches. Two points **a** and **b** are deemed similar if the distance between their descriptors is below an arbitrary threshold. Mathematically,

$$d_M(\mathbf{D_a}, \mathbf{D_b}) < t \ , \tag{2.3}$$

where $d_M(\cdot, \cdot)$ is the distance metric, $\mathbf{D_x}$ is the descriptor of the point $\mathbf{x}$, and t is the threshold. Thus, the value of the threshold t is varied to obtain the ROC curves. Given two images representing the same scene with significant overlap, the true positive rate $p_{correct}$, is the number of correctly matched points concerning the

number of all ground truth matches, which we write,

$$p_{correct} = \frac{\#correctMatches}{\#groundTruthMatches} \, . \tag{2.4}$$

Here, a match or correspondence between two keypoints **a** and **b** is considered as correct via two criteria. The first criterion follows the overlap error in Eq. 2.2, with a threshold of 0.5. In addition to the first criterion, the error in relative location between two points **a** and **b** should not greater than or equal to 3 pixels, that is,

$$\|\mathbf{a} - H\mathbf{b}\| < 3 \, . \tag{2.5}$$

The false positive rate is the probability of false matches. Each descriptor in the current image is compared with each descriptor in the rest of the images, and the number of false matches is counted. The false positives probability is the total number of false matches with respect to the total number of matches. The previous formula was written as per the discussion in [58]

$$p_{false} = \frac{\#falseMatches}{\#matches} \tag{2.6}$$

2.3.2 Patch-based descriptor benchmark [16]

Brown et al., [16] make use of discriminant learning techniques such as Linear Discriminant Analysis (LDA) and Powell minimization to obtain state-of-the-art learned descriptors at lower dimensions. In their work, one of the most important achievements was the introduction of a standard benchmark protocol for descriptors based on patches. Learned descriptors were evaluated on a patch classification benchmark in which the task measures how well a descriptor can distinguish between related and unrelated patches based on their distance in descriptor space. The match/non-match descriptor distances are computed and ROC curves, as explained in Section 2.3.1, are generated by sweeping a threshold. In addition, they also proposed a *95% error rate* metric which is the percent of incorrect matches when 95% of true matches are found. The area under the ROC curve is used as the descriptor score, and higher the score the better is the descriptor's performance.

2.3.3 Benchmark for binary descriptors [36]

For binary descriptors, new additional metrics such as *putative match ratio, precision, matching score, recall* and *entropy* have been proposed in [36]. In this benchmark, Heinly et al. used Oxford dataset provided in Mikolajczyk et al. [58] for evaluating the effects in image exposure, blur, JPEG compression, combined scale and rotation transformations, and perspective transformations of planar geometry. Additionally, they used fountain-P11 and Herz-Jesu-P8 from Strecha et al. [94] for evaluating the effects in perspective transformation of non-planar geometry [36].

Putative match ratio. Addresses the selectivity of the descriptor and describes what fraction of detected features are initially identified as a match. The keypoint matching criteria directly influence the putative match ratio. Less restrictive matching criteria will generate a high putative score, and restrictive match criteria can discard potentially valid matches which leads to less putative match ratio. Putative match is defined as the single pairing of keypoint, in which each keypoint cannot be matched with more than one other keypoint.

$$PutativeMatchRatio = \frac{\#PositiveMatches}{\#Features} \ . \tag{2.7}$$

Precision. This is the number of correct matches out of the set of putative matches. This can also be called as *inlier ratio*. The number of correct matches refers to the geometrically verified putative matches based on known camera positions [36].

$$Precision = \frac{\#CorrectMatches}{\#PutativeMatches} \ . \tag{2.8}$$

Matching score. Equivalent to multiplication of putative match ratio and precision. It describes the number of features that result in correct matches and how well the descriptor is performing.

$$MatchingScore = \frac{\#CorrectMatches}{\#Features} \ . \tag{2.9}$$

Recall. This describes the number of actually found correct matches. The correspondences are the matches between the keypoints in both the images. A low recall could mean that the descriptors are indistinct, matching criterion is too strict or data is too complex.

$$Recall = \frac{\#CorrectMatches}{\#Correspondences} \ . \tag{2.10}$$

Entropy addresses the influence of local feature detectors on descriptors. Entropy computes the amount of spread or randomness in the spatial distribution of the keypoints in the image. Entropy is computed using evenly spaced 2D bins across the image. Each feature point's contribution to the given bin is weighted by a Gaussian relative to its distance to the bin's center. A bin $b(p)$ at position $p = (x, y)$ is given by

$$b(p) = \frac{1}{Z} \sum_{m \in M} G(||p - m||) \ . \tag{2.11}$$

where m is a keypoint in the full set of detected keypoints, M, and G is the Gaussian. A constant of $1/Z$ is multiplied to normalize the sum of all bins to 1. This binning helps us to compute *entropy*:

$$Entropy = \sum_{p} -b(p) * logb(p) \ . \tag{2.12}$$

2.3.4 HPatches Benchmark [10]

Balntas et al. [10]'s HPatches dataset contains 116 scenes with a total of 696 unique images. The first 57 scenes of HPatches dataset [10] exhibit large illumination changes and the other 59 scenes exhibit large viewpoint changes. Specifically, HPatches dataset used a lot of already existing datasets such as Aanæs et al. [1], Cordes et al. [21], Jacobs et al. [41], Mikolajczyk and Schmid [57], Vonikakis et al. [104], and Yu and Morel [111]. Balntas et al. [10] proposed three evaluation protocols in their benchmark: *patch verification*, *image matching* and *patch retrieval*. When *image matching* is similar to evaluation proposed by Mikolajczyk et al. [58], *patch verification* measures the ability of descriptor to classify a patch pair coming from same measurement [10], and *patch retrieval* tests how well a descriptor can match one patch to a pool of patches extracted from multiple images. mAP is used to evaluate the performance of several local features against the proposed metrics.

2.3.5 SfM based Benchmark [85]

Schönberger et al. [85] proposed metrics based on SfM on very large datasets along with metrics proposed by [36, 58, 57] to evaluate the raw matching performance on an image pair. To be as realistic as possible, Schönberger et al. [85] used evaluation protocol and datasets proposed by Heinly et al. [36] for stereo tests. For evaluating reconstructions, Schönberger et al. [85] used Strecha et al. [94] datasets, similar to Heinly et al. [36]. Additionally, the completeness of depth maps is computed by following the evaluation protocol proposed by Hu and Mordohai [39]. Schönberger et al. [85] also used Hane et al. [34] South Building Dataset, a dataset with repetitive scene structures, and Wilson and Snavely [107] Internet Photo Collections which contains high variance in input data.

The newly proposed metrics are based on the quality of the reconstructed models on a SfM setup. A typical SfM pipeline takes multiple images as input, extracts local features, and builds a 3D model out of them. In the process poses of the cameras for each image are also discovered. We discuss the pipeline in more detail in Section 4.3. With the 3D reconstructions and the camera poses, the following metrics have been proposed [85]:

 # Registered Images and # Sparse Points: These two metrics quantify the completeness of the 3D reconstruction. The larger the number of registered images signify the completeness of multi-view stereo reconstructions and larger the number of 3D points constitute more dense and accurate scene representation.

 # Observations per image: The number of verified image projections of sparse points.

Track Length: The number of verified image observations per sparse point. This metric, along with *# Observations per image*, is important for the accurate calibration of the cameras and reliable triangulation.

Reprojection Error: Bundle Adjustment, the core of SfM, is a joint non-linear refinement of cameras and points. The overall *reprojection error* in bundle adjustment indicates the overall reconstruction accuracy. The reprojection error is impacted by the completeness of the graph of feature correspondences and keypoints localization.

Mean Metric Pose Accuracy: This metric is calculated by aligning the reconstructed model to the ground-truth 3D model using a robust 3D similarity transformation estimation such as iterative closest point algorithm.

 # Dense points: Higher number of registered images can lead to additional multi-view photo-consistency constraints and hence more complete results. Hence, the number of dense points can act as a single measure to evaluate the metric accuracy and completeness of dense reconstruction results.

2.3.6 Limitations of existing benchmarks

The benchmarks as mentioned earlier have few limitations in terms of evaluating the local features. *Repeatability Rate* can be considered as a good metric only when we have a finite number of keypoints. If every pixel in the given image is considered as a keypoint, then the *Repeatability Rate* will be artificially inflated to a very high value close to 100%. Many recent local learned descriptors evaluate their performance using patch-pair classification benchmark [16], which can overfit easily. A better performance on patch-based benchmark [16] doesn't necessarily translate to better feature matching quality [9]. Some pruning tests such as nearest neighbor constraints or Lowe's ratio test [55] might compensate for a higher false positive matching score in terms of descriptor distance. Also, the learned descriptors in Brown et al., [16] are generated with Differences of Gaussians (DoG) as keypoints. The high matching score doesn't necessarily imply better performance in subsequent steps in applications such as SfM and MVS. For example, in SfM, finding additional correspondences between images doesn't necessarily guarantee a more accurate reconstruction. Similarly, some descriptors with good average matching performance might not find enough correspondences for challenging image pairs [85].

The evaluation metrics proposed for SfM-based benchmark by Schönberger et al. [85] also are not perfect. Their metrics give an idea about how detailed the reconstructed model is, but do not provide an absolute outcome. For example, number of registered images and sparse points do not tell if this is because of high accuracy or recall. Observations per image is only applicable to the registered sparse points. Mean metric pose accuracy exists to account for how well the images are registered, but this is only possible for rare cases when there is absolute ground-truth pose, and even when it is available, the average error is not a good indicator, especially when

there are many image matching failures.

For any evaluation, having a robust ground-truth is necessary to compare the performance of proposed approaches. One of the key challenges for creating an evaluation benchmark is having a ground-truth to evaluate the real-world performance of local features. Computing depth and accurate poses in the wild without SfM is challenging. Also, since SfM can perform decently on large datasets [80, 85], we propose evaluating the local features on smaller subsets of a larger dataset and compare the change in relative pose information for an image pair with the SfM results on a larger dataset. We show that this type of evaluation can act as a better proxy than simple stereo or path-based matching for understanding the real-world performance of local features.

To properly evaluate the local features, we also require challenging datasets. Verdie et al. [102] proposed a new benchmark dataset to evaluate the performance of repeatability of local features under illumination changes. This benchmark dataset has only illumination changes but no significant viewpoint changes. Oxford dataset has mostly blurred and jpeg compression artifacts. It only consists of 5062 images which makes the dataset small to compute the groundtruth SfM. It also has very few perspective changes and provides very few homographies. We consider all these challenges in existing datasets and propose a new dataset with 25 image collections collected from online sources such as Flickr and Heinly et al. [37] ranging from 75 to *c.a.*4000 images per sequence. This dataset provides high illumination and viewpoint changes which makes the evaluation of local features challenging and close to real-world.

Chapter 3

Systematic Generation of Dataset for Hand Segmentation

In this chapter, we explain our approach for the systematic generation of a dataset for hand segmentation. We first discuss the acquisition device that was used for the thesis. We then explain our dataset acquisition pipeline in Section 3.2, then briefly discuss various machine learning models that were applied in Section 3.3. We next present evaluation of those models on our generated hand segmentation dataset in Section 3.4.

3.1 Acquisition Device

Besides the traditional color cameras which provide RGB information of the scene, consumer depth cameras such as Microsoft's Kinect [115], Intel's RealSense [44] etc., can provide rich information such as structure and shape of objects in the scene additional to RGB information. With these RGB-D cameras, instead of manual annotations, such as Amazon's Mechanical Turk, the synchronized color (RGB) and depth frames from any RGB-D device can be exploited to generate hand segmentation annotations at a larger scale automatically. In this thesis, we automatically generate a hand segmentation dataset with the help of Intel RealSense [44] SR300 RGB-D camera. Our dataset is acquired at a constant framerate of 48Hz and with an image resolution of 640×480.

Figure 3.1: Our dataset is generated by recording a subject performing various hand movements wearing a pair of bright colored gloves in front of a RGB-D camera. To the best of our knowledge, our dataset is the first *two-hand* dataset for hand segmentation.

3.2 Dataset acquisition

For dataset generation, we record subjects performing hand motions in front of a depth camera while wearing *skin-tight* gloves, refer Figure 3.1. Since the gloves fit the subject's hands tightly, minimal geometric aberrations occur to the depth maps. The consistent color of the gloves can be used to the segment the region of interest (ROI) of hands by employing a joint color and depth based segmentation.

After an initial color calibration session, we acquire our dataset following a three-step process:

- We ask the user to perform a few motions according to the *protocol* described below while wearing a pair of colored gloves, and record sequences of (depth, color) image pairs at a constant 48Hz rate with an Intel RealSense SR300 [44].
- We then execute a joint color/depth segmentation to generate masks with a minimal false-positive rate.
- Finally, we discard images containing erroneous labels through manual inspection. This verification per image can be done quickly since the task for human verification is to simply retain or discard the image from dataset.

This task is significantly simpler than manually editing individual images. In our experiments, 20% of the automatically labeled images were discarded. The sequences

Table 3.1: A summary of datasets for hand segmentation from depth imagery.

Dataset	Annotation	#Frames	#Sub.	Hand?	Sensor	Res.
HandSeg[13]	automatic	265,000	14	L/R	SR300	640 × 480
Freiburg [120]	synthetic	43,986	20	L/R	Synthetic	320 × 320
NYU [99]	automatic	6,736	2	L	Kinect v1	640 × 480
HandNet [106]	heuristic	212,928	10	L	SR300	320 × 240

we acquired are in exo-centric configuration, with one or two hands of subjects in the 20-50 age range; see Table 3.1 and Figure 3.1. In Table 3.1, SR300 refers to Intel RealSense SR300 [44], Kinect v1 refers to Microsoft Kinect version 1 [115], #Sub. refers to number of unique subjects used to create the respective datasets, L refers to left hand labels and R refers to right hand labels, and Freiburg [120] proposed a synthetic rendering approach to create the dataset for hand segmentation. We chose Intel RealSense SR300 for our experiments because it is one of the most popular consumer RGBD cameras during the time of this research. Each depth camera has a different noise pattern, and synthetic rendering approaches lack that noise during rendering, and it's still an unsolved problem. Our contribution is to create a high quality automatically labeled hand segmentation dataset with minimal human intervention. In the following subsections, we will explain our dataset acquisition pipeline.

3.2.1 Color glove calibration

To simplify the task of color segmentation, the lighting conditions during acquisition are kept constant, and the camera is not moving. As shown in Figure 3.2, the gloves have a Lambertian material with a constant albedo and a very weak specular component. The Lambertian materials, a.k.a matte, have a consistent brightness from any angle of view from the observer. This simplifies the calibration process, as the color of a pixel on the glove can be explained mainly by the relative orientation of surface normal and light, with only minor brightness variations caused by self-occlusions/shadows.

To calibrate the gloves, we then use a simple yet effective solution. As shown in Figure 3.2, we wrap the glove onto a sphere and acquire calibration images by sparsely sampling the field of view with our probe. Due to its simple geometry and consistent color, the probe can be easily located, and all pixels within its circular profile are then

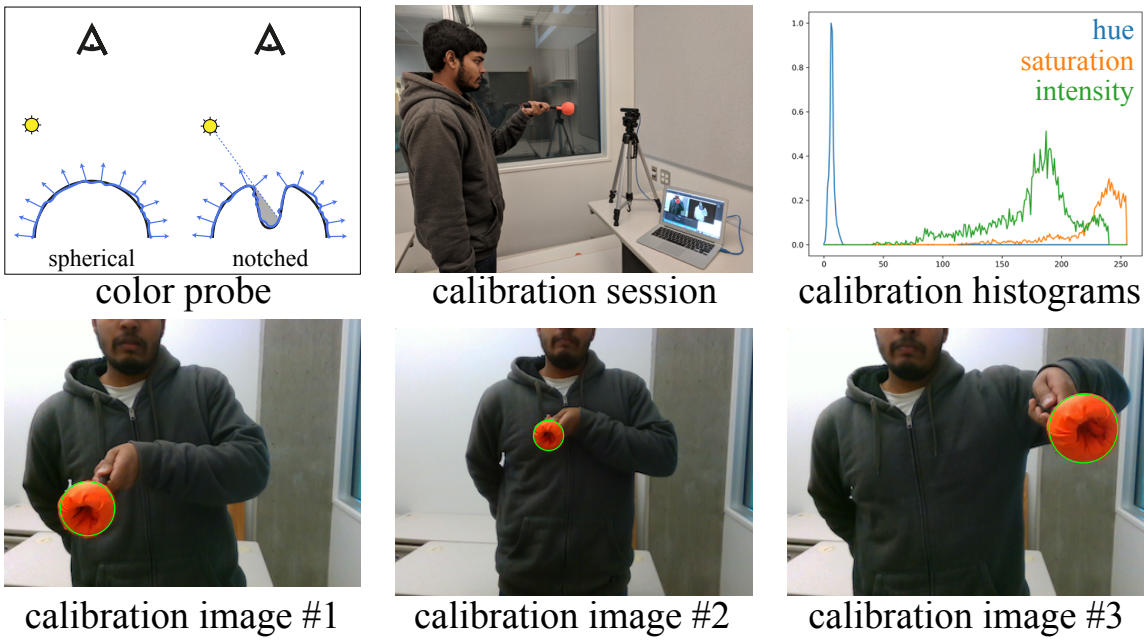

Figure 3.2: Our color calibration setup. (top-left) A hemisphere wrapped in the glove's material contains all of the potential surface normals visible from the camera's point of view. By notching the sphere, we also obtain color variations caused by ambient occlusion and self-shadowing. (top-middle) We calibrate the system by probing various parts of the view frustum, generating a set of calibration images (bottom). All the pixels within the detected circles participate in the computation of the color space (top-right).

used for color calibration. Similarly to what is commonly done for skin segmentation [70], we then convert the calibration images to HSV space, from where conservative min/max thresholds for every channel are then extracted, as shown in Eq. 3.1.

$$M_i = \begin{cases} 1 & \text{if } (h_{min} \leq h_i \leq h_{max}) \\ & \text{and } (s_{min} \leq s_i \leq s_{max}) \\ & \text{and } (v_{min} \leq v_i \leq v_{max}) \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

3.2.2 Acquisition protocol

Similarly to Yuan et al. [112], we attempt to maximize the coverage of the articulation space by asking each user to assume many examples of extremal poses, while capturing the natural motion during each transition. Due to limitations in our automatic

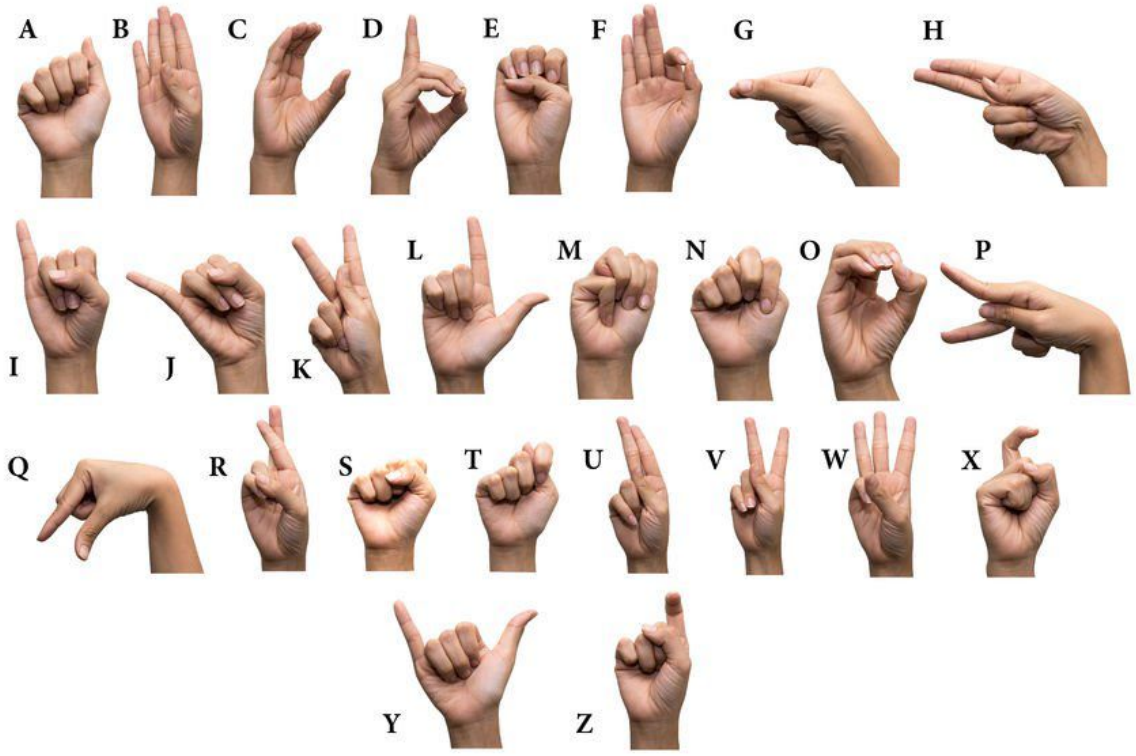

Figure 3.3: Gestures - Set 1 (Credits: Bloximages)

segmentation algorithm, we require the user to keep the hands sufficiently far from the body, as well as from each other (for the two-hands portion of the dataset). Since our camera pipeline acquires frames at 48 FPS, we need to minimize the duplicate static pose frames when users are stuck at providing the next unique pose. So, we also provided some pose sheets to help the user as a reference to create new poses if they run out of ideas. A summary of those pose sheets are provided in Figures 3.3, and 3.4. We also asked the users to perform a grab and hold gestures for different kinds of objects such as a rectangular box, compact disc, spherical ball, and coffee mug.

3.2.3 Segmentation

Even when properly calibrated, segmentation via simple color-space thresholding is sensitive to lighting variation, resulting in noisy annotations; see Figure 3.5a. To remove these small outliers, we implement a morphological opening operation. A morphological opening operation is defined as the dilation followed by erosion of a

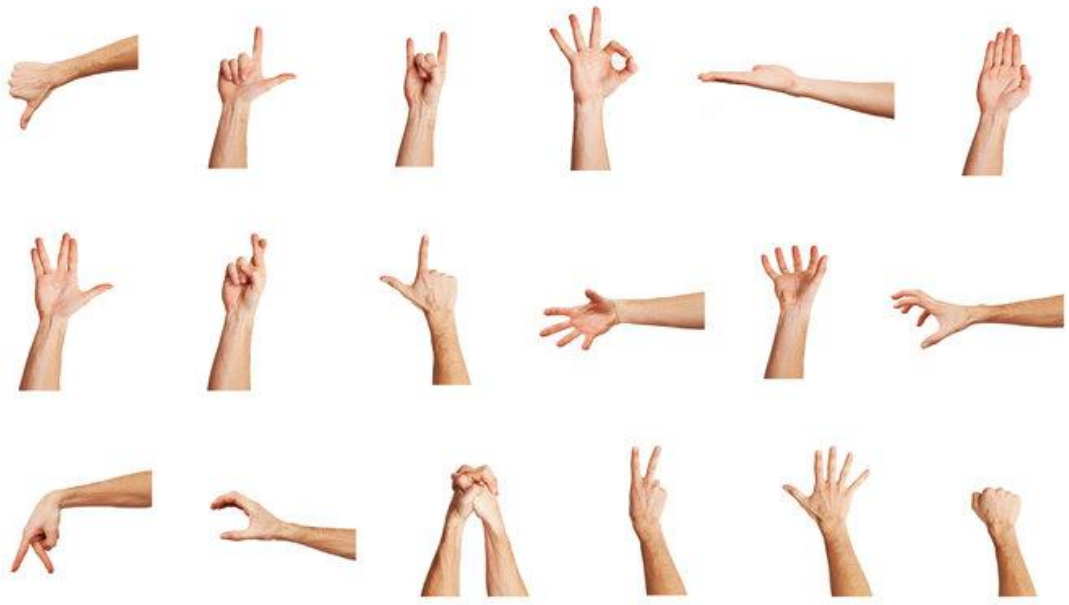

Figure 3.4: Gestures - Set 2 (Credits: News Pakistan)

set A using a structuring element B,

$$A \circ B = (A \ominus B) \oplus B \,, \tag{3.2}$$

where A is the mask we get from simple HSV based color thresholding and B is a 5×5 circular kernel; see Figure 3.5b. The opening operation can remove the majority of noise created during the color segmentation stage. We then connect nearby connected components, with constraint elements closer than 25 pixels, and compute the convex hull of the largest connected component; see Figure 3.5c. We then retrieve the depth values of the pixels within the convex hull and compute their median. As the hull is expected to contain pixels corresponding to the hand mostly, we can identify the hand depth by computing the median of the depth pixels within the hull. We then discard any pixel with a depth sufficiently far from the mean (i.e., the radius of a sphere enclosing the hand in rest pose). Hence, our underlying assumption is that the hands are sufficiently distant and separated from other objects in the scene. When labeling two hands the algorithm is executed merely twice, one for each label, and the results combined to generate the final mask.

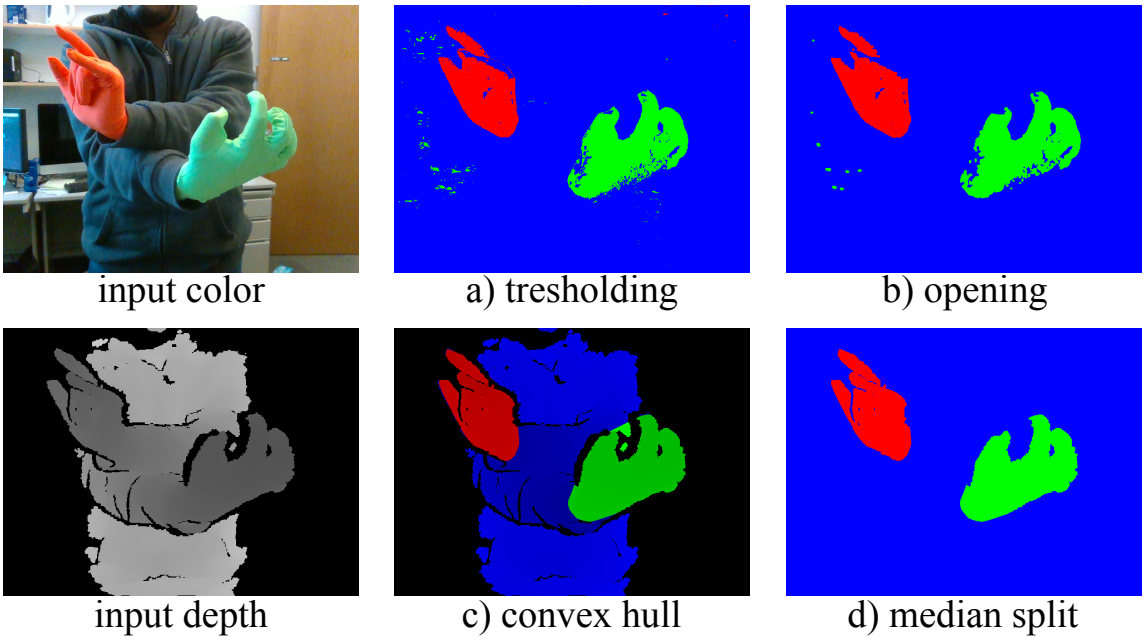

Figure 3.5: We combine color and depth input to extract ground truth segmentation. We first segment the color image via HSV thresholding (a) and remove noise with a morphological opening (b). The convex hull of the labels is used to extract a portion of the depth map (c). As most of the pixels within the hull correspond to the hand, the median of its depth values can be used to discard background pixels (d).

3.3 Learning to segment hands

We now detail the structure of several learning-based semantic segmentation methods which we will then quantitatively cross-evaluate on our dataset in Section 3.4.

Random forests Our first baseline is the *shallow* learning offered by Random Forests popularized for full-body tracking by Shotton et al. [89]. Tompson et al. [99] pioneered its application to binary segmentation of one hand, while Sridhar et al. [93] extended the approach to also learn more detailed part labels (e.g. palm/phalanx labels). Analogously to Shotton et al. [89] and Sridhar et al. [93], our forest consists of 3 trees each of depth 22, and uses the typical depth differential features proposed by Shotton et al. [89]. At a given pixel x, the features to compute can be represented as:

$$f_\theta(I, x) = d_I\Big(x + \frac{o_x}{d_I(x)}\Big) - d_I\Big(x + \frac{o_y}{d_I(x)}\Big) \tag{3.3}$$

where $d_I(x)$ is the depth at pixel x in the image I and parameters $\theta = (o_x, o_y)$ describe offsets o_x and o_y. The normalization of the offsets with the parameter $\frac{1}{d_I(x)}$ will ensure depth invariance in features.

At inference time, random forests are highly efficient, making them suitable for applications like real-time hand tracking. However, while their optimal parameters (offset/threshold) are learned, the features themselves are fixed, and this can result in overall lower accuracy when compared to deep architectures.

3.3.1 Deep convolutional segmenters

To overcome the challenges of shallow learning, we evaluate several recently proposed deep learning convolutional architectures, as well as propose a novel variant with enhanced forward-propagation efficiency and precision; see Figure 3.6. As we have a multi-class labeling problem, we employ the soft-max cross entropy loss. In all our experiments we train our networks with ADAM optimizer with a learning rate of 0.0002, and $\beta_1 = .5$, $\beta_2 = 0.999$ for 50 epochs on an NVIDIA Tesla P100. It has to be noted that these results are preliminary and more details on training can be found in Bojja et al. [13].

Fully convolutional neural network (FCNN) Long et al. [54] proposed an architecture where a coarse segmentation mask is produced via a series of convolutions and max-pooling stages (*encoder*), where the low-resolution image is then upsampled (*decoder*) via bilinear interpolation – the *FCN32s* variant in Long et al. [54, Fig.3]. As this process produces a blurry segmentation mask, a sharper mask can be obtained by combining this image with the higher-resolution activations from previous layers in the network; the *FCN16s* and *FCN8s* variants. Unfortunately, the initial layers in the network only encode very localized features. Hence while this process does produce sharper results, it also introduces high-frequency misclassifications in uncertain regions. Another problem of FCNN is their difficulty in dealing with the problem of *class imbalance*: in our training images, the cardinality of background pixels is significantly larger than the one of hand pixels. We overcome this problem by incorporating the class frequency in the loss [60, 108], which effectively prevents the network from converging to one that trivializes the output to be always classified as background. Even with these changes, the limited accuracy achieved by this network can be understood by noting that the encoder layer is learned, while the decoder

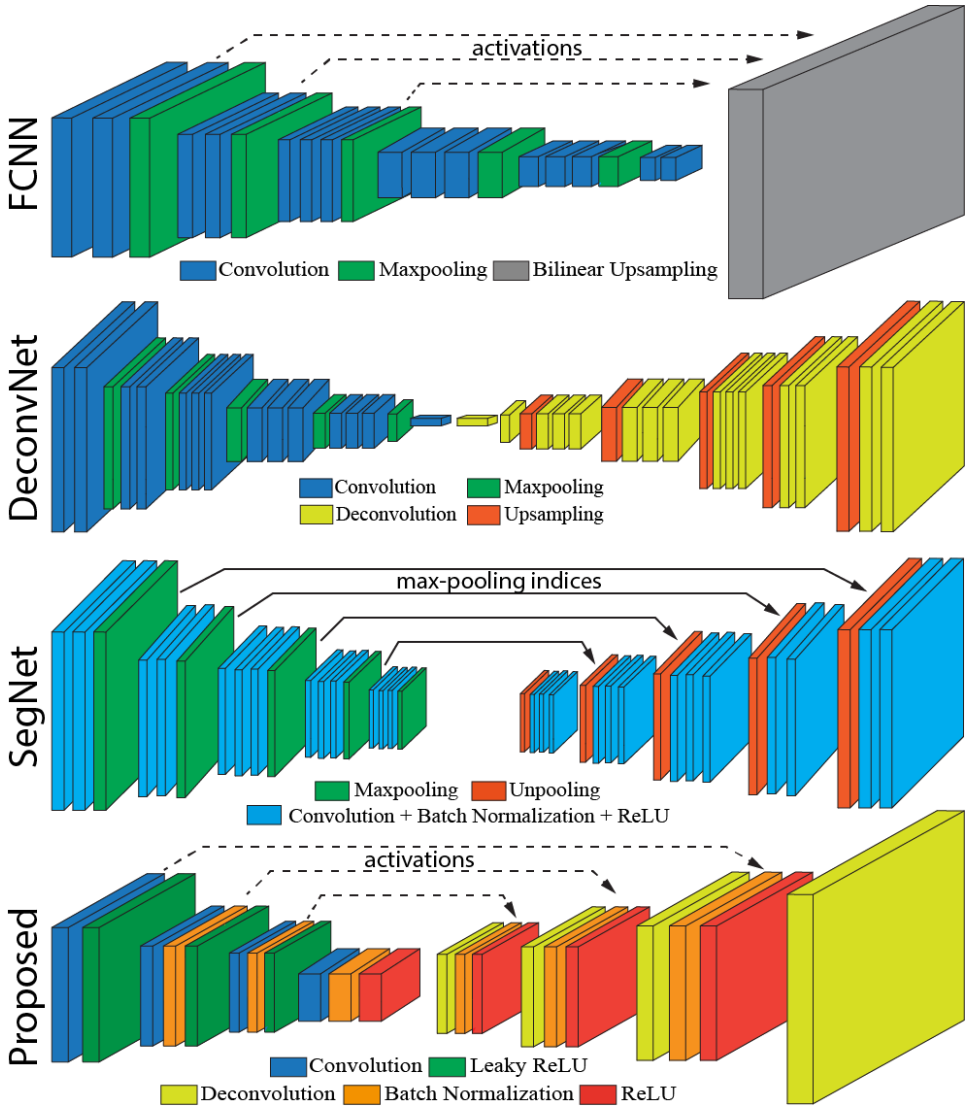

Figure 3.6: Semantic segmentation CNN architectures.

layer is not.

Learnt encoder-decoder networks The popular *SegNet* [7] and *DeconvNet* Noh et al. [63] semantic segmentation networks follow an encoder-decoder architecture. Similarly to FCNNs, the encoder is realized via a sequence of convolutions and max-pooling operations. However, rather than relying on interpolation, the decoder used to generate high-resolution segmentation is also learned. Both architectures employ an *unpooling* operation that inverts the *max-pooling* in the encoder. Similarly to DeconvNet, SegNet upsamples the feature maps via memorized max-pooling indices in the corresponding encoder layer. Further, while a simple series of convolutions follow unpooling in SegNet, DeconvNet employs a series of *deconvolution* layers. This deconvolution is the *transpose* of a convolution, in turn, represented by the gradient of a convolution layer. Along with this characteristic, a large number/size of deconvolution layers makes DeconvNet significantly more computationally intensive to train end-to-end without a justifiable increase in accuracy [7].

Proposed baseline Our novel architecture is a *hybrid* encoder-decoder: we employ a hierarchy of deconvolution layers (a-la DeconvNet), and to improve sharpness and local detail of the predictions we forward information from encoder to decoder through skip-connections (a-la FCNN or U-Net [73]). Differently, from other architectures, note how our encoders/decoders do not contain any max-pooling/unpooling layer. Pooling layers are meaningful in classification tasks, where we are interested in the maximal activation in a bank of filters without retaining fine-grained information about its spatial structure. However, the definition of a downsampling layer is essential, as a bottleneck in the network is necessary to learn the low-dimensional manifold of hand appearance. In our encoder network, this is achieved by *stride-2 convolution* layers. As pooling indices are not available, in the decoder we symmetrically employ *stride-2 deconvolution* layers, as this enables the network to learn an appropriate upsampling filter. The simplicity in our design results in efficient forward propagation, while simultaneously achieving superior accuracy as shown in Table 3.2.

3.4 Evaluation

We quantitatively evaluate our dataset and learning architecture from three different angles according to the metrics defined below. In Section 3.4.1, we evaluate the

Table 3.2: Accuracy of each segmentation method

Method	**Random Forests**	**FCN**	**DeConvNet**	**SegNet**	**HandSeg**
mIoU	0.69	0.78	0.91	0.94	0.93

Table 3.3: Runtime of each segmentation method

Method	**Random Forests**	**FCN**	**DeConvNet**	**SegNet**	**HandSeg**
Train time	3h	149h	57h	83h	29h
Test time	1ms	41ms	16ms	30ms	5ms

performance of several classical learning architectures on our data, revealing how our proposed architecture can produce state-of-the-art accuracy while remaining efficient in terms of forward-propagation. A thorough evaluation in terms of accuracy of these networks was conducted by the first author of Bojja et al. [13] and the results are shown in Table 3.2.

3.4.1 Segmenting with different architectures

In Table 3.3, we compare the different learning approaches in terms of training and test time. Although Random Forests are the fastest to train and to infer on, they perform poorly when compared to deep neural networks. Due to its simple upsampling scheme, FCN(32s) performs the worst among the evaluated networks which manifest in low precision/recall scores for hands, while performing well on the background class.

Thanks to its learned decoder network, SegNet obtains much better results. However, its architecture is too heavy, resulting in a runtime that is not suitable for real-time tracking applications. Our proposed architecture not only outperforms the others in terms of accuracy, but it is also fast to forward-propagate, running at *c.a.*200 FPS. The increase in accuracy of our network can be justified by the fact that downsampling operators are learned, rather than max-pooled, and by the connections bringing high-frequency information into the decoder layer.

Due to slow training, we were not able to perform a quantitative comparison to DeconvNet (a single epoch took over 12 hours to complete). Note how DeconvNet has 15 deconvolution layers, while we only have 4. Given its architecture, we expect

it to have an inference time even larger than the one we measured on SegNet. Comparatively, our network should also be more accessible to train, as vanishing gradients are resolved via skip-connections, while batch-normalization further helps speed-up training.

3.4.2 Qualitative evaluation

In Figure 3.7, we provide qualitative segmentation results on our proposed dataset. As expected, the bilinear upsampling of FCNN loses many of the details, resulting in blob-like segmentation masks. By learning the decoder SegNet can perform better, but fine-grained details can still be blurred out; see sample #3 and #4. In comparison, our network can resolve fine-grained details thanks to the reuse of encoder feature maps in the decoder.

Sample #5 and #6 show typical failure cases of our architecture, where a part of the left hand is misclassified as right – these type of mistakes would create large outliers in a tracking optimization and could be avoided via regularization layers [19, 116], or by training with a loss that accounts these configurations [46].

Figure 3.8 shows other challenging frames. Sample #1 illustrates how the network can still segment the hands of multiple persons, although it was trained on frames containing a single individual. This reveals the generalization capabilities of our network, which did not only learn to segment *one/two* regions, but also learned a latent *shape-space* for human hands. Sample #2 shows a person holding a cup, while Sample #3 has the hand lying flat on the body. These scenarios are difficult, as the network has never seen a hand interacting with objects. Accuracy can be improved by accounting for additional information in the color channel, or by learning the appearance of the object via training examples.

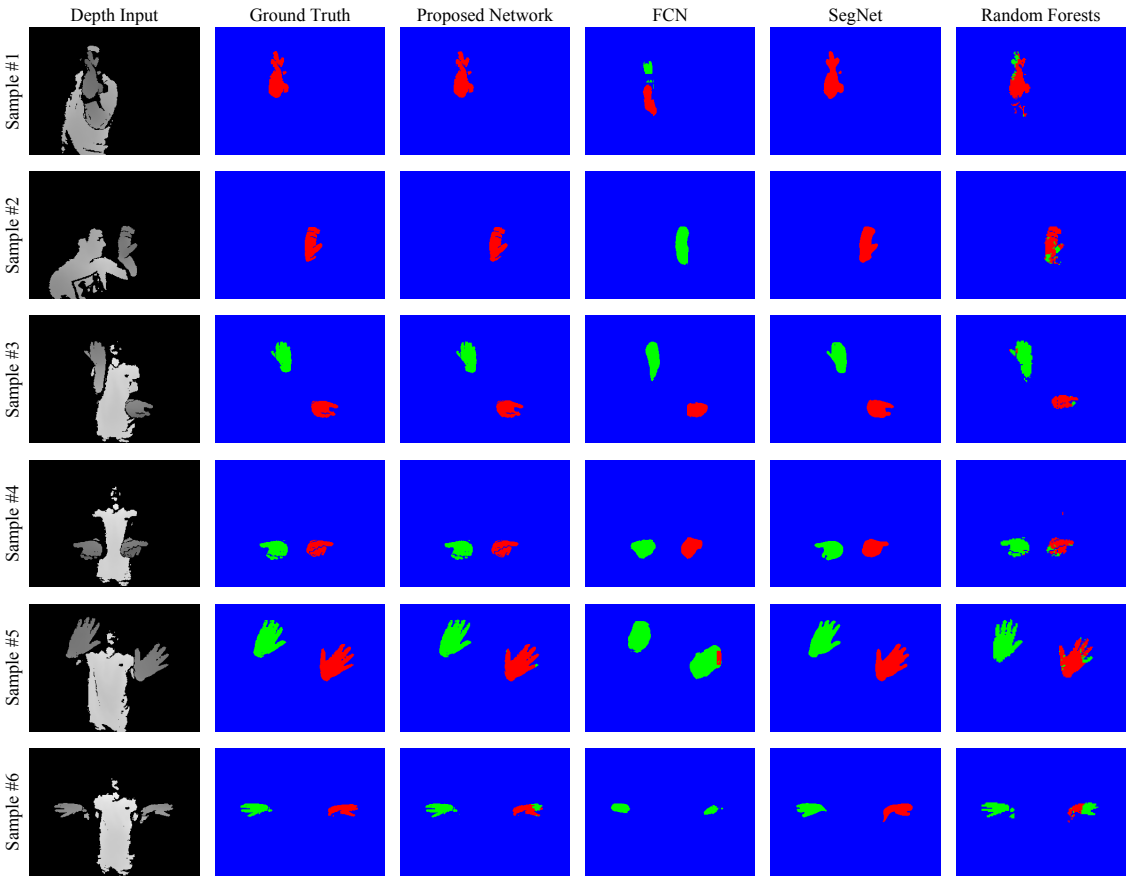

Figure 3.7: We illustrate a few examples of hand segmentation performance across the considered learning techniques.

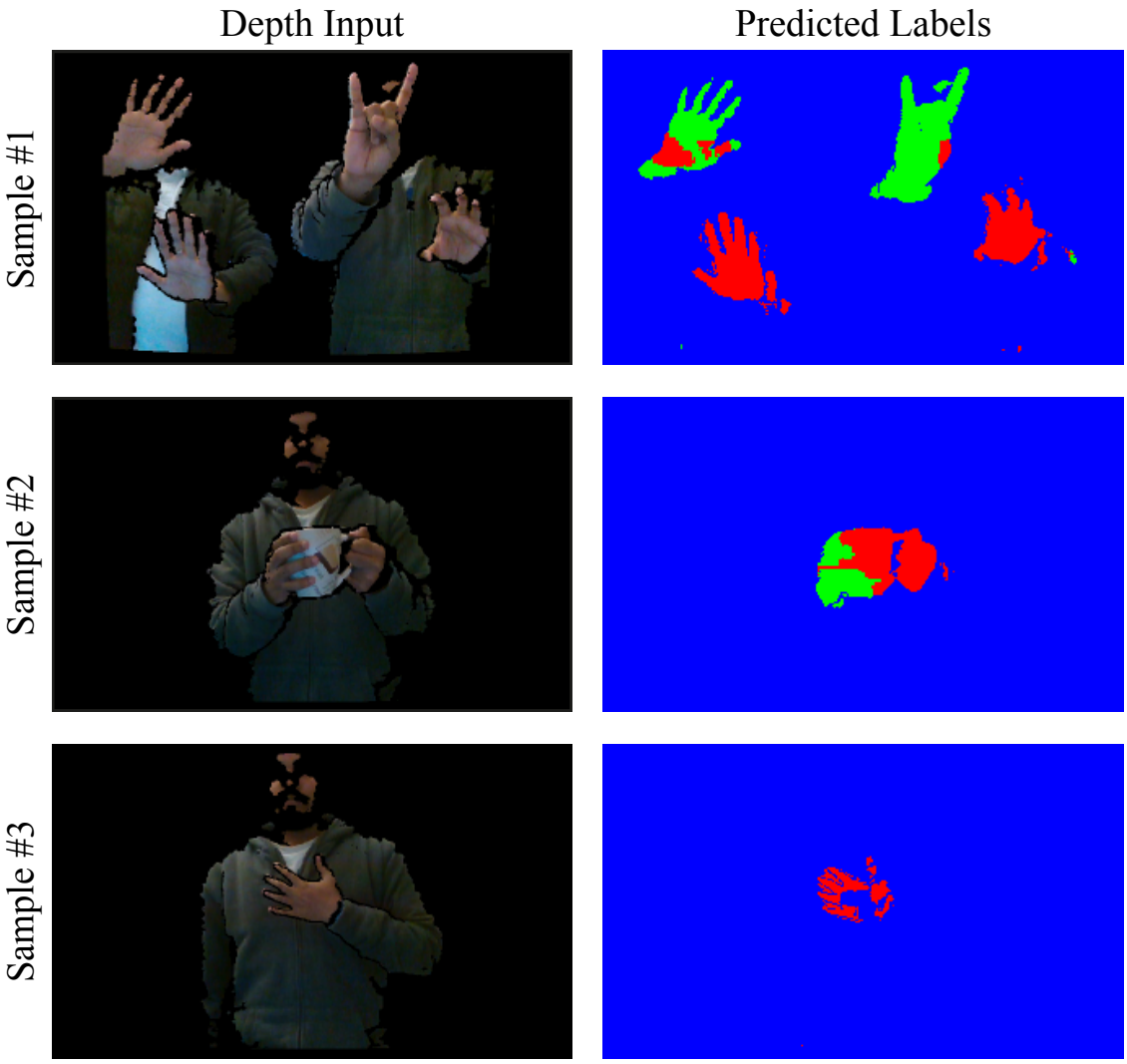

Figure 3.8: A selection of challenging segmentation failures.

Chapter 4

A Structure-from-Motion-based Local Feature Benchmark

As discussed earlier in Section 2.3.6, existing local feature benchmarks [58, 57, 16, 36, 85] have various limitations. Therefore, in this chapter, we propose a new evaluation benchmark which does not rely only on individual stereo matching results, nor requires a fully 3D annotated dataset. To evaluate local features, we propose two main tasks as a part of our evaluation benchmark - wide-baseline stereo matching and Structure-from-Motion (SfM) from small subsets. With the two tasks, and the benchmark developed for this thesis, we will host a challenge involving modern local features.

We first explain more about the benchmark tasks in Section 4.1. We then discuss about the dataset we use in Section 4.2. We finally present the entire pipeline of our evaluation benchmark in Section 4.3 and conclude this chapter in Section 4.4 with some observations made on some datasets with various local features: SIFT [55], SURF [12], ORB [74], AKAZE [4] and SuperPoint [25].

4.1 Benchmark Tasks

In our evaluation benchmark, we perform two main tasks: Wide-baseline stereo matching, and SfM from small subsets generated from a larger dataset. The first task is aimed to show local feature performances with more traditional metrics on the new dataset, while the latter focuses on providing a more practical point of view.

4.1.1 Task 1: Wide-baseline stereo matching

In this task, we attempt to match keypoints over two images across wide baselines. Image pairs are selected according to visibility constraints so that at least part of the scene is guaranteed to overlap. Specifically, we employ a visibility constraint of 0.1%, which is translated to 10% of shared overlap region in the image pair.

We employ the following two main metrics in this task to evaluate the performance of local features.

- **Matching score (sparse methods only):** This is defined as the ratio of ground-truth correspondences that can be recovered either with nearest neighbor matching (and optionally with Lowe's ratio test [55]). The ground-truth correspondences are generated using the ground-truth depth (generally a 3D point cloud) to translate pixel coordinates (u, v) from one image to another, and a fixed threshold to decide if two keypoints match. Occluded points that cannot be matched are excluded from the calculation. The matching score metric isolates the performance of local features.

  The spirit of this metric is identical to existing benchmarks, but with a twist. Our setup, as we will explain later, does not assume planar geometry nor has perfect depth estimates. We, therefore, rely on epipolar geometry [35].

- **Pose estimation (sparse or dense):**

  - For *sparse* methods, we apply a robust matching strategy (e.g. RANSAC with the five point algorithm [62]) and use the surviving inliers to retrieve the relative pose between the two cameras.
  - For *dense* methods, we assume that the pose is already computed.

  We then measure performance with the angular difference between the estimated and ground truth vectors for both rotation and translation. To reduce this to one value, we use a variable threshold (the same value for rotation and translation) to determine each pose as correct or not and compute the area under the curve up to the angular threshold τ. This value is thus the mean average precision up to τ, or mAP^{τ}. We consider τ values as: 5°, 10°, 15°, 20°, and 25°.

For the challenge, the ranking metric is pose estimation; in specific, we rank by $mAP^{15^{\circ}}$, which we have found empirically to be an adequate proxy for wide-baseline stereo matching performance.

4.1.2 Task 2: SfM from small subsets

While modern solutions have shown very promising results in stereo, it is not clear how much of these improvements remains after large-scale reconstruction with Bundle Adjustment. An alternative approach is thus to evaluate local features directly for SfM, as done by the Schönberger et al. [85]. Unfortunately, it is not feasible to acquire truly accurate depth measurements for large image sequences collected from various sensors, so that under most circumstances the best we can do is collect statistics such as the number of observations obtained with the reconstruction, the track length, or the reprojection error [85]. While this is informative, most methods seem to perform similarly under this scenario on very large datasets [85].

By contrast, we propose to build SfM reconstructions from small (3, 5, 10, 25) subsets of images randomly generated from a much larger dataset (see Section 4.3) and use the poses obtained from the entire dataset as ground truth. We believe this can provide a better proxy for learning and evaluating feature extractors and matching algorithms for the task of pose estimation.

In specific, we subsample the test sets to 100 images and generate 100 different subsets of 3, 5, 10, and 25 images. The subsets are sampled randomly from each dataset, accounting for visibility constraints. To compute the mAP, we use the same procedure as for stereo, with every possible combination of two images (i.e., 3 combinations for 3 images, 10 for 5 images, etc.) and average the results. Note that this penalizes reconstructions that are missing images or fail. As for stereo, we use mAP^{15° to rank the performance of different local features. We explain in more detail about the pipeline of evaluation benchmark in Section 4.3.

4.2 Photo Tourism Dataset

To learn and evaluate models that can perform well under a wide range of situations, it is of paramount importance to collect information from multiple sensors obtained at different times, from different viewpoints. In Section 2.3.6, we briefly discussed the limitations of some datasets which were used for evaluating local features in previous benchmarks. To compensate for the known limitations, we turned to photo-tourism data for our proposed evaluation benchmark. In this dataset, we rely on 25 photo-tourism image collections of famous landmarks collected initially by the Yahoo Flickr Creative Commons 100M (YFCC) dataset [97] and Reconstructing the world in six

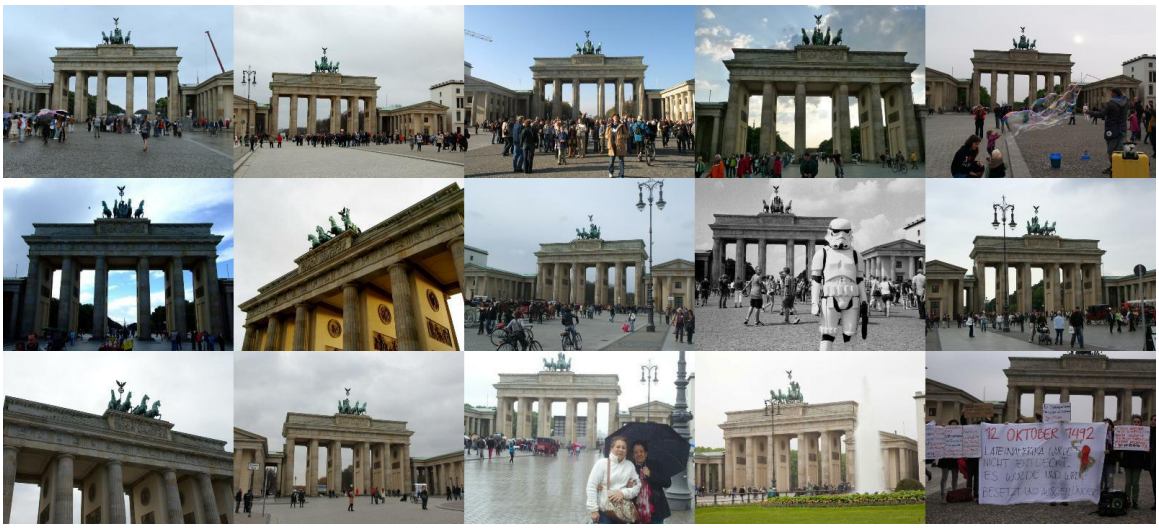

Figure 4.1: Examples from brandenburg_gate

days [37]. The sequences range from 75 images to almost 4k per sequence. Figure 4.1 shows an example of how one of the datasets look like.

We can obtain dense 3D reconstructions from these large collections of images with off-the-shelf Structure from Motion (SfM) algorithms [83, 84]. We rely on COLMAP [83, 84, 85], a state-of-the-art method. In addition to a sparse point cloud, COLMAP can densify the estimates to produce noisy but useful depth maps for every image. We post-process these depth maps by projecting each image pixel into 3D space at an estimated depth, and we mark it as invalid depth point if the closest 3D point from the reconstruction is further than a threshold. The resulting depth maps will be still noisy, but most of the occluded pixels are filtered out. While not perfect, these estimates can be used to project points across images and train keypoint detectors and descriptors, as done for example by LF-Net [67]. These 'clean' depth maps along with images are made publicly available to train custom local feature detectors.

We perform a visibility check to guarantee a reasonable degree of overlap (> 0.1 overlap ratio) for each image pair with the SfM points visible over both images. Our metric is based on the size of the bounding box containing all of the points visible in either image and applied over both views. We use these criteria to select valid image subsets for testing and provide the entire visibility matrix for training, which can be easily thresholded to generate a list of valid pairs.

We use the five representative datasets from the test set of Ono et al. [67] as our test sequences: *reichstag*, *milan_cathedral*, *mount_rushmore*, *sagrada_familia*

Table 4.1: Photo Tourism - Training and Validation Data

Training Sequences	**# Images**	**# 3D Points**
brandenburg_gate	1363	100040
buckingham_palace	1676	234052
colosseum_exterior	2063	259807
grand_place_brussels	1083	229788
hagia_sophia_interior	888	235541
notre_dame_front_facade	3765	488895
palace_of_westminster	983	115868
pantheon_exterior	1401	166923
prague_old_town_square	2316	558600
sacre_coeur	1179	140659
st_peters_square	2504	232329
taj_mahal	1312	94121
temple_nara_japan	904	92131
trevi_fountain	3191	580673
westminster_abbey	1061	198222
Total	25.6K	3.7M

and *united_states_capitol*. For the challenge, we will suggest fourteen sequences for training, and eleven for testing, as provided in Ono et al. [67].

The information regarding these datasets are provided in tables 4.1 and 4.2.

4.3 Benchmark Pipeline

In this section, we describe the architectural design of our evaluation benchmark and explain each component in our pipeline.

4.3.1 Input

As illustrated in Figure 4.2, our evaluation pipeline encapsulates the entire SfM pipeline. We can therefore provide local features to evaluate at various stages, for example, the feature extraction stage or the feature matching stage with a custom matching method.

Our framework expects the contents inside the input 'directory' in the following format:

Table 4.2: Photo Tourism - Testing Data

Testing Sequences	**# Images**	**# 3D Points**
british_museum	660	73569
florence_cathedral_side	108	44143
lincoln_memorial_statue	850	58661
london_bridge	629	72235
milan_cathedral	124	33905
mount_rushmore	138	45350
piazza_san_marco	249	95895
reichstag	75	17823
sagrada_familia	401	120723
st_pauls_cathedral	615	98872
united_states_capitol	258	35095
Total	4107	696K

- **images/:** The directory containing the images for testing.
- **images.txt:** The text file containing the names of images along with their relative path.
- **visibility/:** The directory containing information about the overlap ratio between images. The files in this directory replicate the names of the images. Overlap ratio can help us in generating valid pairs to compute stereo matching tests. For all our experiments, we kept a constant overlap ratio of 0.1 (or 100 keypoint matches).
- **visibility.txt:** The text file containing the names of the visibility files along with their relative path.
- **calibration/:** The directory containing the information related to the cameras that captured the images. The information in these files contains camera intrinsic parameters and ground truth camera pose (obtained from COLMAP by running reconstruction over entire dataset).
- **calibration.txt:** The text files containing the names of the calibration files along with their relative path.

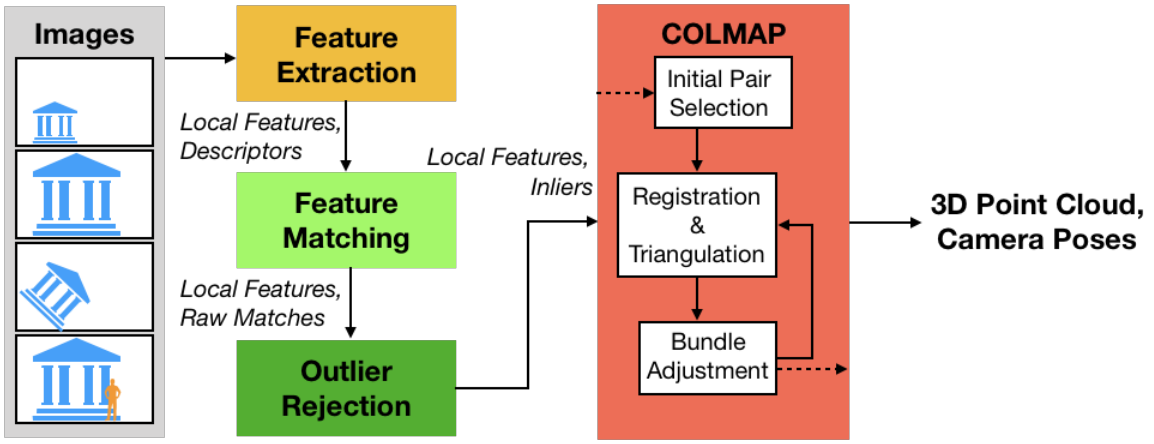

Figure 4.2: Benchmark pipeline

4.3.2 Generation of (3, 5, 10, 25) bags

To evaluate different local features, we create several test bags. To guarantee that all of these test bags share the same image corpus, we apply a two-stage process. In the first stage, we randomly pick 100 images from the total dataset. We use these 100 images to generate 100 sets of 3, 5, 10 and 25 bags.

While picking the images randomly, we enforce a *valid triplet* constraint. A triplet is a set of three images. According to the *valid triplet* constraint, a triplet is said to be valid when each image in the triplet set can be paired with two other images in the same triplet set. These pairs are considered valid if they satisfy the visibility constraint (> 0.1 overlap ratio).Mathematically we write,

$$\{I_a, I_b, I_c\} \in T_D, (I_a \leftrightarrow I_b) \cup (I_a \leftrightarrow I_c) \cup (I_b \leftrightarrow I_c) \ , \tag{4.1}$$

where I_x is an image from the larger dataset D, T_D is the complete list of valid triplets for dataset D, $\leftrightarrow$ signifies the visibility constraint between image pairs, and if union ($\cup$) of all those constraints are satisfied, then the triplet is valid. We use this *validtriplet* constraint to generate the list of all valid triplets.

The generated triplet list is used to compute 100 images set and the following 3, 5, 10, and 25 bags from the 100 images set. The steps for generating the sets are described below:

- **100 set:** While creating the 100 set, we need to make sure that every image in the 100 set satisfies the *valid triplet* constraint. Picking a random sample of three images, checking if it's a valid triplet and making sure that the random

Figure 4.3: Brandenburg Gate - COLMAP 3D Reconstruction

sample is unique can be computationally expensive. Therefore, instead of picking a random sample from the entire dataset, we pick a random *valid triplet* from the triplet list T_D. This ensures that valid triplet constraint is satisfied for every image in the 100 set.

While picking a random *valid triplet* from triplet list, there can be a possibility that either one or two or all images in that triplet set are already present in the 100 set. We discard the duplicate guesses of random samples by maintaining a local cache of already selected triplets. For those *valid triplets*, which have one or two images already in the 100 set, we add the missing images and stack this *valid triplet* into the local cache of already guessed *valid triplets*.

For edge cases like when the 100 set contains only 98 or 99 elements, we only look for the *valid triplet* sets with two or one unique images respectively. We continue this iteration until we generate the complete 100 set. We will then generate the directory for 100 set in the 'Inputs' structure as mentioned in the

previous subsection.

- **(3, 5, 10, 25) bags:** In this stage, we create 100 unique sets of (3, 5, 10, 25) bags. Before generating the bags, we reduce the search space of valid triplets by generating the new list from the *100 set*. We use this new triplet list T_{100} for randomly selecting a valid triplet. Generating the 3 bags is straight-forward. We randomly select a valid triplet from T_{100} and keep a local cache of already selected triplets to avoid duplicates. For 5, 10 and 25 bags, we employ a similar strategy we applied for generating 100 set. Instead of selecting a valid triplet from T_D, we select it from T_{100}. Finally, we create directories as 'Inputs' structure for the 3, 5, 10 and 25 bags.

4.3.3 Feature Extractor

Computing local features is a vital step in 3D computer vision tasks such as Structure-from-Motion (SfM) and Multi-View Stereo (MVS) [3, 37, 71, 80, 81, 82]. As such, since the introduction of SIFT [55], several hand-crafted and learned local features have been proposed [12, 74, 4, 25, 67, 109]. To scale up our evaluation pipeline for any local feature extractor, we implement a wrapper class named 'FeatureExtractor' that encapsulates various local feature implementations as in OpenCV [14]. This class provides a single method 'detectAndCompute' which takes an image as input and extracts keypoints and descriptors. This wrapper class currently supports local feature implementations such as SIFT [55], SURF [12], ORB [74] and AKAZE [4] from the popular open-source computer vision library, OpenCV [14]. This gives flexibility for challenge participants to design their custom local feature extractor using the library of their choice and use our 'FeatureExtractor' wrapper class to easily plugin into the evaluation pipeline. We also provided helper methods to convert precomputed features into format required by our benchmark pipeline.

Technical Details. We use the 'FeatureExtractor' to extract keypoints and descriptors from the images in the $< input > /images/$ directory. We can set the number of keypoints and descriptors we need to extract from each image while initializing the 'FeatureExtractor'. To reduce the computations, we extract the maximum number of features that can be extracted from each image. We sort them using the feature score. We store the keypoints and descriptors in NumPy array format in 'keypoints/'

and 'descriptors/' directories respectively. The names of these stored files are the same as the image names.

Extracting keypoints and descriptors can become slow when we have to compute 100's of test bags. Therefore, we first optimize the implementation by parallelizing the feature extraction method on individual images. Parallelizing the feature extraction can speed up the algorithm by utilizing all the cores on the machine, but parallelizing doesn't necessarily translate to reducing the number of computations involved in extracting keypoints and descriptors. The keypoints and descriptors for a given image by any feature extractor is unique. Since the *100 set* already contains all the images out of which (3, 5, 10, 25) bags are created, we need to compute the keypoints and descriptors for the *100 set* and symbolic link to those files when the same image pops up in the bags.

4.3.4 Feature Matching

To compute feature matching between an image pair, we first generate all valid pairs of images in a given bag that passes the visibility constraint, $\leftrightarrow$. Once we have generated the image pairs, we compute feature matches. For computing feature matches, we need to compute the distance matrix for the descriptors and consider the $k(=1)$ nearest neighbors as the successful matches. The elements in the distance matrix change according to the data-type of descriptors. The element d_{pq} in distance matrix is the distance between descriptors p and q. For non-binary descriptors, the distance is measured in Euclidean distance as shown in Eq. 4.2.

$$d_{pq} = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2} \tag{4.2}$$

Where p and q are non-binary descriptors, and n is the length of each descriptor. For binary descriptors, the distance is measured using the Hamming distance metric as shown in Eq. 4.3.

$$d_{pq} = \sum_{i=1}^{n} |q_i - p_i| \tag{4.3}$$

Where p and q are binary descriptors, and n is the length of each descriptor. We can observe that the Hamming distance metric is similar to the Manhattan distance.

Technical Details. For our experiments, we fixed the maximum number of features to be used for feature matching to 8000. Since we already sorted the keypoints and descriptors, we merely clip the length of the descriptors and keypoints if the size of their NumPy arrays is beyond 8000. We use these clipped descriptors to compute the distance matrix.

4.3.5 Compute Match Scores

In Section 4.1, for wide-baseline stereo matching we mentioned that matching score and pose estimation are two metrics we used to evaluate the local features. In this method, we describe how we compute the matching scores and pose estimation.

- **Matching Scores:** For computing matching scores, we use the ground-truth pose information for an image pair $\{R_1, t_1\}$ and $\{R_2, t_2\}$ which is generated by running COLMAP [83, 84] on large datasets. With these absolute poses, we compute the change in pose $\{dR_{gt}, dt_{gt}\}$. However, as we do not have accurate depth estimates, we rely on epipolar geometry for evaluating the correctness of matches. Specifically, we use the symmetric epipolar distance [35]. While this is not perfect, in most cases corresponds to actual geometric verification. Moreover, the epipolar constraint is what is used at the end to compute the camera pose in case of stereo setups.

  The symmetric epipolar distance between any two points is the Euclidean distance from the corresponding epipolar lines. The following equation can give the symmetric epipolar distance between two images with keypoints x and x':

$$\sum_i d(x'_i, Fx_i)^2 + d(F^\top x'_i, x_i)^2 = \sum x'^\top Fx \Big(\frac{1}{(x'^\top F)_1^2 + (x'^\top F)_2^2} + \frac{1}{(Fx)_1^2 + (Fx)_2^2} \Big) \tag{4.4}$$

  Where F is fundamental matrix between the image pair.

- **Pose estimation:** As a part of pose estimation metrics, we compute the error in $\{dR, dt\}$ between two images. We use the RANSAC with 5 point algorithm [62] to compute the essential matrix with a threshold of 0.01, in the calibrated camera coordinate system. We decompose the essential matrix into relative pose difference between the image pair $\{dR, dt\}$. We compare these values with ground-truth pose information to compute $\{err_q, err_t\}$, where err_q is the error

Figure 4.4: British Museum - COLMAP 3D Reconstruction

in rotation measured in quaternions and err_t is the error in translation. We later use this $\{err_q, err_t\}$ to compute the mAP^{τ°.

4.3.6 SfM using COLMAP

For multi-view task, we rely on COLMAP [83, 84]. COLMAP is a general purpose Structure from Motion (SfM) and Multi-view Stereo (MVS) pipeline. By default COLMAP uses SIFT [55] features to compute SfM and MVS. Also, COLMAP has been heavily optimized for SIFT descriptors, and some algorithm parameters have been hardcoded to achieve speed-ups on GPUs. This makes the use of general purpose SfM and MVS, COLMAP, for evaluating different local features non-trivial. We now describe how we created a wrapper around COLMAP to easily evaluate different local features for MVS tests. Note that we assume that local features and their matches across image pairs have already been computed as discussed in the previous sections.

- **Feature Importer:** As by default COLMAP only supports SIFT [55] feature extractor to compute keypoints and descriptors we need to use its non-default settings with workarounds. Instead of passing images into COLMAP to compute different local features. We pass the previously computed local features (SIFT, SURF, ORB, AKAZE, and SuperPoint) and descriptors into COLMAP. COLMAP expects the imported keypoints to have the image pixel location (x_i, y_i), angle and size, and the imported descriptors to have 128 integer values. While features with less than 128 dimensions can fit nicely by simply appending zeros, this is not true for all methods. Therefore, as a part of feature importer, we import original keypoints provided by local feature extractors and an empty array filled with integer 0's as descriptors into the COLMAP. Zero descriptors are to act as an assertion and ensure that if we do not provide custom matches, the pipeline fails.

- **Matches Importer:** Because of the descriptor import issues, and because we would also like to evaluate various matching methods, we provide our own matches to COLMAP. Specifically, we provide the feature matches using the methods discussed in Section 4.3.5. We flag the matches computed by us as 'raw' so that COLMAP is allowed to discard few keypoints as outliers during the reconstruction phase.

- **COLMAP Mapper:** In this phase, we use the keypoints, images, and feature matches we imported into COLMAP database to initialize the reconstruction. Our test bags contain (3, 5, 10, 25) bags and the results after reconstruction are discussed in Section 4.4. COLMAP implements an incremental reconstruction algorithm, see Figure 4.2. In the first stage, COLMAP will carefully look for an image pair for performing two-view reconstruction [80]. Bad initialization can lead to poor reconstruction results. A good initialization image pair will have significant overlap over the entire scene graph [80]. The new images are registered to the current 3D model using a Perspective-N-Point problem [30] using the computed feature correspondences (2D) to triangulated points (3D) in previously registered images.

  Generally, the 2D-3D correspondences are outlier contaminated. COLMAP uses a robust pose solver to minimize the error in camera pose estimation and computes a reliable triangulation [80]. In computer vision, the process of finding a point in 3D space based on its projections onto an image pair is known as

Table 4.3: Stereo - Averaged over all sequences. Results are sorted using mAP^{15°.

Method	**# KPs**	**MS**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	7883.7	0.030	0.001	0.012	**0.037**	0.097	0.178
SuperPoint	646.1	**0.035**	0.001	0.009	0.034	0.083	0.152
SIFT	7893.8	0.026	0.001	0.009	0.031	0.078	0.141
ORB	7521.9	0.029	0.001	0.008	0.029	0.073	0.126
SURF	7697.9	0.025	0.001	0.006	0.024	0.058	0.113

triangulation. Triangulation is a critical step in SfM. It increases the stability of the existing 3D model and enables registration of new images with computed 2D-3D correspondences from the current image [80]. Bundle adjustment aims to find the optimal camera pose and scene structure simultaneously by minimizing the reprojection error with non-linear methods and refining outliers. After a successful reconstruction, COLMAP outputs the 3D points and cameras with corresponding pose information into binary files. The example 3D reconstruction outputs from COLMAP can be seen in Figures 4.3 and 4.4.

After the above steps, we use the pose information given by the COLMAP on individual (3, 5, 10, 25) bags and compare them with ground truth pose information to estimate the (err_q, err_t). In the following section, we discuss benchmark outcomes.

4.4 Results

For all our experiments, we kept a fixed number of keypoints to 8000, visibility threshold of 100, inlier threshold of 0.0001, unless stated otherwise. To compute the camera pose and the essential matrix, we applied RANSAC with the five-point algorithm.

Our tests include evaluating the performance of local features: SIFT [55], SURF [12], ORB [74], AKAZE [4] and SuperPoint [25] on the datasets - *reichstag*, *milan_cathedral*, *mount_rushmore*, *sagrada_familia* and *united_states_capitol*. We used OpenCV implementations for SIFT, SURF, ORB, and AKAZE. For SuperPoint, we used the authors implementations. One important thing to note here is that for SuperPoint, due to the way the method is developed, we we only able to obtain *c.a.*650 local features per image on average, which is much lower than other compared methods.

The averaged results over all the sequences for wide-baseline stereo matching and multi-view stereo tests are provided in Tables 4.3 and 4.4, where

Table 4.4: MVS - Averaged over all sequences. Results are sorted using mAP^{15°.

Method	**SR**	**TL**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	**79.0**	2.89	0.238	0.342	**0.403**	0.453	0.499
SIFT	64.5	2.81	0.203	0.277	0.325	0.365	0.395
SURF	48.8	2.70	0.102	0.157	0.196	0.226	0.252
SuperPoint	40.1	**3.01**	0.077	0.133	0.169	0.199	0.222
ORB	37.7	2.66	0.080	0.130	0.162	0.186	0.208

- **Method** is the feature extractor method implemented to extract keypoints and descriptors.
- **# KPs** is the average number of keypoints over all sequences.
- **MS** is the average matching score over all sequences.
- mAP^{x° is the mean average precision at angular threshold x°.
- **SR** is the success ratio (percentage) in the reconstruction with COLMAP, out of 100 subsets.
- **TL** is the average track length.

4.4.1 Task 1: Wide-baseline stereo matching

Matching Scores. From Table 4.3, we can observe that SuperPoint [25] has the highest matching score despite it's lesser average number of keypoints. Following the SuperPoint, we have AKAZE [4], ORB [74], SIFT [55] and SURF [12]. It is interesting to note that ORB has better matching score when compared to SIFT and SURF.

Camera Pose. Though SuperPoint [25] has the highest matching score, it clearly is not the best corresponding to camera pose metric. AKAZE [4] has the highest mAP^{15° of 0.037 followed by SuperPoint, SIFT, ORB and SURF. ORB performs poorer than SIFT even though it has a higher matching score than SIFT.

From the above observations, it is interesting to note that matching scores and camera pose are not completely correlated.

Breakdown by Sequence. Now, let us dive in and observe the breakdown by sequence for stereo tests in Tables 4.5, 4.6, 4.7, 4.8, 4.9. Again, the table is sorted in decreasing order of mAP^{15°. From the tables we can observe that, SuperPoint [25] better on *milan_cathedral*, *mount_rushmore* and *sagrada_familia*. But its performance is low on *reichstag* and *united_states_capitol*. The matching scores magnitudes are pretty low because of very strict inlier threshold 0.0001. Also, it has to be noted that we cannot control the number of keypoints we can extract using the learned feature extractor SuperPoint [25]. SURF [12] is showing a consistently poor performance on all the datasets in wide-baseline stereo matching tests. ORB [74] has a medium to poor performance on all the datasets. SIFT [55] and AKAZE [4] show consistently good performance on all the datasets.

Matching score under different thresholds. Finally, as the default threshold of 0.0001 is very strict, we would like to see the performance of different local features with respect to matching scores when we change the inlier threshold of $\{0.0001, 0.001, 0.01\}$. As these thresholds are in calibrated camera coordinates, where the long-side of the image plane is considered to be of length one, on a 640×480 image, 0.0001 would correspond to 0.01 pixels, that is, exact match. Often, benchmarks

Table 4.5: Stereo - *milan_cathedral*. Results are sorted using mAP^{15°.

Method	**# KPs**	**MS**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
SuperPoint	694.8	0.023	0.001	0.008	0.049	0.118	0.200
AKAZE	7754.2	0.020	0.000	0.008	0.027	0.080	0.158
SIFT	7761.9	0.018	0.001	0.007	0.025	0.078	0.158
SURF	7544.4	0.017	0.000	0.004	0.021	0.054	0.109
ORB	7805.4	0.018	0.000	0.001	0.010	0.044	0.081

Table 4.6: Stereo - *mount_rushmore*. Results are sorted using mAP^{15°.

Method	**# KPs**	**MS**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
SuperPoint	516.3	0.052	0.001	0.003	0.018	0.062	0.133
AKAZE	7840.3	0.049	0.001	0.004	0.015	0.070	0.160
SIFT	7835.0	0.042	0.000	0.002	0.015	0.064	0.141
ORB	7588.0	0.048	0.000	0.001	0.013	0.056	0.109
SURF	7561.6	0.041	0.000	0.000	0.012	0.048	0.110

Table 4.7: Stereo - *reichstag*. Results are sorted using mAP^{15°.

Method	**# KPs**	**MS**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
ORB	7288.5	0.034	0.000	0.014	0.050	0.114	0.192
SIFT	7978.1	0.030	0.001	0.010	0.041	0.088	0.153
AKAZE	7961.9	0.037	0.001	0.015	0.040	0.105	0.197
SuperPoint	661.3	0.050	0.001	0.011	0.037	0.091	0.166
SURF	7719.0	0.027	0.000	0.009	0.035	0.067	0.131

Table 4.8: Stereo - *sagrada_familia*. Results are sorted using mAP^{15°.

Method	**# KPs**	**MS**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
SuperPoint	721.1	0.018	0.001	0.007	0.032	0.068	0.114
AKAZE	7862.3	0.017	0.000	0.007	0.021	0.048	0.082
ORB	7732.0	0.016	0.000	0.003	0.016	0.029	0.052
SIFT	7894.2	0.014	0.000	0.005	0.016	0.033	0.057
SURF	7732.5	0.014	0.000	0.003	0.009	0.024	0.040

Table 4.9: Stereo - *united_states_capitol*. Results are sorted using mAP^{15°.

Method	**# KPs**	**MS**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	8000.0	0.028	0.002	0.026	0.082	0.180	0.290
SIFT	8000.0	0.025	0.001	0.020	0.056	0.127	0.197
ORB	7195.6	0.027	0.003	0.021	0.055	0.124	0.196
SURF	7932.0	0.025	0.002	0.015	0.043	0.095	0.175
SuperPoint	637.0	0.032	0.002	0.014	0.034	0.077	0.145

allow tolerance of 5 pixels [67], which roughly corresponds to 0.01.

The Tables 4.19, 4.20, and 4.21 provide results for the variation of matching scores across the 100 sets for different local features. We consider a match between two keypoints as an inlier if the geodesic distance, see Eq. 4.4, is less than the inlier threshold. We can observe that SuperPoint [25] has consistent high performance in matching score across all sequences for all the inlier thresholds. We like to confirm from this observation that very high matching score doesn't necessarily imply good reconstruction, see Table 4.4. SIFT [55] and SURF [12] have almost same matching scores but their performance varies a lot in multi-view stereo tests. Though ORB

[74] has almost the same matching scores as AKAZE [4], the later has been the top performer in multi-view stereo tests.

4.4.2 Task 2: SfM from small subsets

Camera Pose. From Table 4.4, we can observe that AKAZE has a very high reconstruction success ratio followed by SIFT, SURF, SuperPoint, and ORB. When it comes to Average Track Length, SuperPoint performs better than AKAZE. AKAZE has a high mAP^{15° of 0.403 followed by SIFT, SURF, SuperPoint, and ORB. The possibility of a poor performance by SuperPoint in MVS tests is because of less number of keypoints but surprisingly though it has fewer keypoints, it performed better than ORB in multi-view stereo. Our results show that ORB is not a good choice for pair-wise (stereo) or multi-view stereo in real-world applications.

Breakdown by Sequence. Now, let us dive in and observe the breakdown by sequence for multi-view stereo tests. The Tables 4.10, 4.11, 4.12, 4.13 and 4.14 have been generated by sorting the results with respect to the metric mAP^{15°. From the results we can observe that AKAZE [4] is the clear winner for all the datasets. It is also interesting to note that AKAZE [4] has a very high success ratio when compared to other feature extractor methods. Success ratio refers to the number of successful reconstructions done by COLMAP. Following AKAZE [4], SIFT [55] has shown consistent performance in multi-view stereo tests, except in *united_states_capitol* dataset. SuperPoint [25] can be considered as an exception for poor performance because COLMAP needs more than 4000 keypoints to get a decent reconstruction.

From breakdown by sequence results in both wide-baseline stereo and multi-view stereo tests, we can observe that high matching scores don't necessarily imply good 3D reconstructions. AKAZE [4] has no consistent top performance in stereo tests, but in SfM it is the clear winner in all the dataset sequences. So, it clearly shows that just matching score based metrics don't necessarily tell about the performance of the local features in challenging cases. The SuperPoint [25] scores consistently good in wide-baseline stereo matching based tests but its performance deteriorates in SfM based metrics.

Breakdown by Bag Size: Multi-View Stereo tests. In Tables 4.15, 4.16, 4.17, and 4.18, we breakdown sequences by bag size i.e., (3, 5, 10, 25). We can observe that

Table 4.10: MVS - *milan_cathedral*. Results are sorted using $mAP^{15°}$.

Method	**SR**	**TL**	$mAP^{5°}$	$mAP^{10°}$	$mAP^{15°}$	$mAP^{20°}$	$mAP^{25°}$
AKAZE	88.5	3.16	0.388	0.546	0.621	0.664	0.705
SIFT	84.8	3.10	0.401	0.540	0.610	0.649	0.679
SURF	57.0	2.89	0.180	0.269	0.324	0.359	0.385
SuperPoint	44.0	3.21	0.128	0.204	0.249	0.278	0.302
ORB	20.8	2.77	0.044	0.076	0.093	0.108	0.123

Table 4.11: MVS - *mount_rushmore*. Results are sorted using $mAP^{15°}$.

Method	**SR**	**TL**	$mAP^{5°}$	$mAP^{10°}$	$mAP^{15°}$	$mAP^{20°}$	$mAP^{25°}$
AKAZE	80.0	3.22	0.180	0.293	0.363	0.424	0.470
SIFT	62.7	3.08	0.146	0.235	0.286	0.335	0.367
SuperPoint	67.0	3.60	0.108	0.201	0.265	0.320	0.362
SURF	51.5	3.05	0.073	0.123	0.168	0.204	0.238
ORB	35.5	2.83	0.030	0.076	0.115	0.142	0.167

Table 4.12: MVS - *reichstag*. Results are sorted using $mAP^{15°}$.

Method	**SR**	**TL**	$mAP^{5°}$	$mAP^{10°}$	$mAP^{15°}$	$mAP^{20°}$	$mAP^{25°}$
AKAZE	90.5	3.19	0.335	0.462	0.533	0.583	0.629
SIFT	88.8	2.99	0.215	0.330	0.402	0.465	0.509
ORB	72.0	2.82	0.183	0.298	0.371	0.418	0.456
SURF	77.0	2.90	0.151	0.269	0.342	0.396	0.436
SuperPoint	59.0	3.21	0.102	0.191	0.242	0.280	0.310

Table 4.13: MVS - *sagrada_familia*. Results are sorted using $mAP^{15°}$.

Method	**SR**	**TL**	$mAP^{5°}$	$mAP^{10°}$	$mAP^{15°}$	$mAP^{20°}$	$mAP^{25°}$
AKAZE	79.0	3.05	0.499	0.586	0.618	0.636	0.659
SIFT	72.0	3.09	0.486	0.538	0.567	0.582	0.598
ORB	54.8	2.87	0.245	0.310	0.334	0.353	0.367
SURF	44.0	2.88	0.253	0.299	0.315	0.329	0.339
SuperPoint	20.0	3.11	0.101	0.131	0.145	0.157	0.164

Table 4.14: MVS - *united_states_capitol*. Results are sorted using mAP^{15°.

Method	**SR**	**TL**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	81.2	2.70	0.033	0.097	0.170	0.254	0.327
ORB	22.0	2.58	0.008	0.026	0.048	0.068	0.088
SURF	42.2	2.41	0.003	0.016	0.043	0.070	0.100
SIFT	37.7	2.55	0.004	0.016	0.041	0.072	0.095
SuperPoint	14.0	2.86	0.004	0.013	0.027	0.039	0.051

AKAZE [4] and SIFT [55] are the consistent top performers. Again, the results in these tables have been sorted with respect to the mAP^{15° metric. It is interesting to observe that the performance of SURF [12] gradually increased with increase in bag size. For a bag size of 25, every feature extractor has a $> 97\%$ success ratio for 3D reconstructions. This explains why the results are so high for all the feature extractor methods in Schönberger et al. [85]'s benchmark.

Table 4.15: MVS - All sequences - Bag Size 3. Results are sorted using mAP^{15°.

Method	**SR**	**TL**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	57.0	2.29	0.060	0.131	0.176	0.215	0.263
SIFT	50.2	2.27	0.041	0.085	0.125	0.168	0.199
SuperPoint	37.8	2.36	0.036	0.079	0.109	0.138	0.157
ORB	29.8	2.24	0.011	0.050	0.077	0.095	0.115
SURF	32.8	2.24	0.015	0.044	0.072	0.090	0.110

Table 4.16: MVS - All sequences - Bag Size 5. Results are sorted using mAP^{15°.

Method	**SR**	**TL**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	81.6	2.81	0.222	0.334	0.399	0.455	0.502
SIFT	60.2	2.72	0.175	0.252	0.302	0.342	0.372
SuperPoint	39.8	2.88	0.068	0.129	0.166	0.196	0.221
SURF	42.6	2.65	0.068	0.115	0.153	0.182	0.207
ORB	32.2	2.64	0.059	0.101	0.129	0.151	0.172

Table 4.17: MVS - All sequences - Bag Size 10. Results are sorted using mAP^{15°.

Method	**SR**	**TL**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	98.4	3.58	0.433	0.561	0.635	0.689	0.734
SIFT	83.2	3.43	0.393	0.495	0.548	0.586	0.614
SURF	71.0	3.21	0.223	0.311	0.364	0.407	0.441
ORB	51.0	3.11	0.169	0.239	0.281	0.312	0.337
SuperPoint	42.8	3.78	0.125	0.192	0.234	0.263	0.287

Table 4.18: MVS - All sequences - Bag Size 25. Results are sorted using mAP^{15°.

Method	**SR**	**TL**	mAP^{5°	mAP^{10°	mAP^{15°	mAP^{20°	mAP^{25°
AKAZE	100.0	5.27	0.657	0.759	0.807	0.835	0.855
SIFT	99.2	4.86	0.598	0.685	0.727	0.754	0.774
SURF	97.2	4.39	0.491	0.593	0.645	0.680	0.705
ORB	99.8	3.81	0.445	0.555	0.615	0.654	0.683
SuperPoint	97.4	5.34	0.404	0.513	0.571	0.608	0.636

4.4.3 Benchmark – Final Remarks

From our observations, we would like to conclude that SfM based metrics are necessary to evaluate the performance of the local features along with matching score metrics. We believe that our benchmark can provide the computer vision community a proper evaluation challenge to come up with robust local features. The benchmark is being hosted as a challenge and the most up-to-date results with more local features can be observed at the following link: $https://image-matching-workshop.github.io$. The teaser of our challenge website is shown in Figure 4.5.

The code for our benchmark is also available public and can be found at the following link - $https://github.com/vcg-uvic/sfm\_benchmark\_release$.

Figure 4.5: Image Challenge Workshop - Website

Table 4.19: Matching Scores, with inlier threshold = 0.0001

Sequence	SIFT	SURF	AKAZE	ORB	SuperPoint
milan_cathedral	0.018	0.017	0.020	0.018	**0.023**
mount_rushmore	0.042	0.041	0.049	0.048	**0.052**
reichstag	0.030	0.027	0.037	0.034	**0.050**
sagrada_familia	0.014	0.014	0.017	0.016	**0.018**
united_states_capitol	0.025	0.025	0.028	0.027	**0.032**

Table 4.20: Matching Scores, with inlier threshold = 0.001

Sequence	SIFT	SURF	AKAZE	ORB	SuperPoint
milan_cathedral	0.058	0.054	0.064	0.056	**0.073**
mount_rushmore	0.126	0.123	0.140	0.142	**0.150**
reichstag	0.094	0.084	0.112	0.106	**0.151**
sagrada_familia	0.045	0.045	0.052	0.050	**0.058**
united_states_capitol	0.079	0.078	0.087	0.084	**0.099**

Table 4.21: Matching Scores, with inlier threshold = 0.01

Sequence	SIFT	SURF	AKAZE	ORB	SuperPoint
milan_cathedral	0.178	0.166	0.192	0.172	**0.216**
mount_rushmore	0.360	0.351	0.380	0.390	**0.395**
reichstag	0.262	0.241	0.296	0.290	**0.385**
sagrada_familia	0.138	0.139	0.156	0.151	**0.175**
united_states_capitol	0.237	0.233	0.254	0.250	**0.286**

Chapter 5

Conclusions

In this thesis, we have focused on the importance of datasets and benchmarks, in the era of deep learning. As discussed in Chapter 2, datasets have played critical role in enlarging the capacity of deep neural entworks. At the same time, benchmarks have guided and stimulated the development of more practical computer vision methods, including deep learning-based ones. We have seen that for depth-based hand segmentation, and for modern local features, these two aspects lack.

Regarding the datasets, we have introduced a new systematic way of acquiring high-quality dataset for depth-based hand segmentation that is significantly larger than what is currently available, as well as the dataset itself. Specifically, we created a high quality hand segmentation dataset consisting of 265,000 frames captured at a resolution of 640×480 using Intel RealSense SR300 [44]. The dataset is publicly available for research purposes and can be accessed from the following link: https://gfx.uvic.ca/pubs/2018/bojja2018handseg/page.md.

Through the systematic process, we were able to create a dataset without the extensive manual labour that is required without our method. Our dataset contains high-accuracy dense pixel annotations, large pose variations, and many different subjects. We demonstrated the effectiveness and the quality of our datasets with various machine learning methods. We also proposed a novel segmentation network that is faster than existing baselines, and provides superior segmentation quality.

We then turned our attention to proper benchmarking of local features. We have shown the limitations of existing benchmarks, and have also demonstrated experimentally that the field of local features has been partially mislead by metrics that are not directly related to their practical performance.

In order to overcome this problem, we propose a new benchmark with two challenging tasks - Wide-baseline stereo matching and SfM on small subsets. We hosted our benchmark as a challenge on the following link - https://image-matching-workshop.github.io. We also open-sourced the source code for our benchmark pipeline at - https://github.com/vcg-uvic/sfm_benchmark_release

Through our benchmark, we revealed that existing metrics have not properly guided local feature development. For example, SuperPoint, a deep learning-based recent method, performs best in terms of matching scores, but when it comes to actual camera pose estimation it falls short of AKAZE. This became even more evident when multiple views are considered. These findings are discovered for the first time, even though local feature research has a long history. We believe our benchmark will become a new landmark in a long-standing important problem in computer vision.

Bibliography

[1] Henrik Aanæs, Anders Lindbjerg Dahl, and Kim Steenstrup Pedersen. Interesting interest points. *International Journal of Computer Vision*, 97(1):18–35, 2012.

[2] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. Tensorflow: A System for Large-Scale Machine Learning. In *USENIX Conference on Operating Systems Design and Implementation*, pages 265–283, 2016.

[3] Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. Building rome in a day. In *2009 IEEE 12th International Conference on Computer Vision*, pages 72–79. IEEE, 2009.

[4] Pablo F Alcantarilla and T Solutions. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Transactions in Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2011.

[5] Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011.

[6] Antonis A Argyros and Manolis IA Lourakis. Real-time tracking of multiple skin-colored objects with a possibly moving camera. In *European Conference on Computer Vision*, pages 368–379. Springer, 2004.

[7] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[8] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011.

[9] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *British Machine Vision Conference*, volume 1, page 3, 2016.

[10] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5173–5182, 2017.

[11] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *International Conference on Computer Vision*, 2015.

[12] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[13] Abhishake Kumar Bojja, Franziska Mueller, Sri Raghu Malireddi, Markus Oberweger, Vincent Lepetit, Christian Theobalt, Kwang Moo Yi, and Andrea Tagliasacchi. Handseg: An automatically labeled dataset for hand segmentation from depth images. *arXiv*, 2018.

[14] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobbs journal of software tools*, 3, 2000.

[15] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, 30(2):88–97, 2009.

[16] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):43–57, 2011.

[17] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman. Long term arm and hand tracking for continuous sign language tv broadcasts. In *British Machine Vision Conference*, 2008.

[18] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[19] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *Proceedings of International Conference on Learning Representations*, 2015.

[20] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems*, pages 2414–2422, 2016.

[21] Kai Cordes, Bodo Rosenhahn, and Jörn Ostermann. Increasing the accuracy of feature evaluation benchmarks using differential evolution. In *2011 IEEE Symposium on Differential Evolution (SDE)*, pages 1–8. IEEE, 2011.

[22] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893. IEEE Computer Society, 2005.

[23] J. Ruiz del Solar and R. Verschae. Skin detection using neighborhood information. In *Automatic Face and Gesture Recognition*, 2004.

[24] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[25] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, June 2018.

[26] Piotr Dollar, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[27] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012.

[28] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[29] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[30] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[31] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[32] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[33] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[34] Christian Hane, Christopher Zach, Andrea Cohen, Roland Angst, and Marc Pollefeys. Joint 3d scene reconstruction and class segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 97–104, 2013.

[35] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[36] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In *European Conference on Computer Vision*, pages 759–773. Springer, 2012.

[37] Jared Heinly, Johannes L Schonberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the world* in six days*(as captured by the yahoo 100 million

image dataset). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3287–3295, 2015.

[38] Erik Hjelmås and Boon Kee Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, 2001.

[39] Xiaoyan Hu and Philippos Mordohai. Least commitment, viewpoint-based, multi-view stereo. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 531–538. IEEE, 2012.

[40] Gary B Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[41] Nathan Jacobs, Nathaniel Roman, and Robert Pless. Consistent temporal variations in many outdoor scenes. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–6. IEEE, 2007.

[42] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM International Conference on Multimedia*, pages 675–678, 2014.

[43] Michal Kawulok, Jolanta Kawulok, and Jakub Nalepa. Spatial-based skin detection using discriminative skin-presence features. *Pattern Recognition Letters*, 2014.

[44] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. *arXiv*, 2017.

[45] Cem Keskin, Furkan Kıraç, Yunus Emre Kara, and Lale Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *European Conference on Computer Vision*, pages 852–863. Springer, 2012.

[46] Nicholas Kolkin, Gregory Shakhnarovich, and Eli Shechtman. Training deep networks to be spatially sensitive. In *International Conference on Computer Vision*, 2017.

[47] Matej Kristan, Jiri Matas, Ales Leonardis, Michael Felsberg, Luka Cehovin, Gustavo Fernandez, Tomas Vojir, Gustav Hager, Georg Nebehay, and Roman Pflugfelder. The visual object tracking vot2015 challenge results. In *International Conference on Computer Vision Workshops*, pages 1–23, 2015.

[48] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[49] Laboratory for Intelligent and Safe Automobiles, UCSD. The vision for intelligent vehicles and applications (VIVA) challenge. http://cvrr.ucsd.edu/vivachallenge/, 2015.

[50] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of International Conference on Machine Learning*, pages 609–616. ACM, 2009.

[51] Hui Liang, Junsong Yuan, and Daniel Thalmann. Parsing the hand in depth images. *IEEE Transactions on Multimedia*, 16(5):1241–1253, 2014.

[52] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[53] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[54] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[55] David G Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, page 1150. IEEE, 1999.

[56] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface*, 2013.

[57] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[58] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65 (1-2):43–72, 2005.

[59] A. Mittal, A. Zisserman, and P. H. S. Torr. Hand detection using multiple proposals. In *British Machine Vision Conference*, 2011.

[60] Mohammadreza Mostajabi, Payman Yadollahpour, and Gregory Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[61] Natalia Neverova, Christian Wolf, Graham W Taylor, and Florian Nebout. Hand segmentation with structured convolutional learning. In *Asian Conference on Computer Vision*, pages 687–702. Springer, 2014.

[62] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):0756–777, 2004.

[63] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision*, 2015.

[64] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. In *Proceedings of Computer Vision Winter Workshop*, 2015.

[65] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Training a feedback loop for hand pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3316–3324, 2015.

[66] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3D tracking of hand articulation using kinect. In *British Machine Vision Conference*, 2011.

[67] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: learning local features from images. In *Advances in Neural Information Processing Systems*, pages 6237–6247, 2018.

[68] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. Devito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic Differentiation in Pytorch. In *Advances in Neural Information Processing Systems*, 2017.

[69] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[70] Son Lam Phung, A Bouzerdoum Sr, and D Chai Sr. Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2005.

[71] Filip Radenovic, Johannes L Schonberger, Dinghuang Ji, Jan-Michael Frahm, Ondrej Chum, and Jiri Matas. From dusk till dawn: Modeling in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5488–5496, 2016.

[72] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[73] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[74] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. 2011.

[75] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Fast image-based localization using direct 2d-to-3d matching. In *2011 International Conference on Computer Vision*, pages 667–674. IEEE, 2011.

[76] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. Large-scale location recognition and the geometric burstiness problem. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1582–1590, 2016.

[77] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. Efficient & effective prioritized matching for large-scale image-based localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(9):1744–1756, 2017.

[78] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[79] Daniel Scharstein and Richard Szeliski. Middlebury stereo evaluation. *Middlebury stereo evaluationversion*, 2, 2008.

[80] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[81] Johannes L Schonberger, Filip Radenovic, Ondrej Chum, and Jan-Michael Frahm. From single image query to detailed 3d reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5126–5134, 2015.

[82] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016.

[83] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[84] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, 2016.

[85] Johannes Lutz Schönberger, Hans Hardmeier, Torsten Sattler, and Marc Pollefeys. Comparative Evaluation of Hand-Crafted and Learned Local Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[86] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 519–528. IEEE, 2006.

[87] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2015.

[88] Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[89] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[90] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European Conference on Computer Vision*, pages 746–760. Springer, 2012.

[91] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. *International Conference on Computer Vision*, 2015.

[92] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive markerless articulated hand motion tracking using RGB and depth data. In *International Conference on Computer Vision*, 2013.

[93] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[94] Christoph Strecha, Alex Bronstein, Michael Bronstein, and Pascal Fua. Ldahash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(1):66–78, 2012.

[95] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision*, 2017.

[96] Andrea Tagliasacchi, Matthias Schroeder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum (CGF) (Proc. SGP)*, 2015.

[97] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. Yfcc100m: The new data in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[98] Giorgos Tolias, Yannis Avrithis, and Hervé Jégou. Image search with selective match kernels: aggregation across single and multiple images. *International Journal of Computer Vision*, 116(3):247–261, 2016.

[99] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions Of Graphics*, 2014.

[100] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1808–1817, 2015.

[101] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5038–5047, 2017.

[102] Yannick Verdie, Kwang Yi, Pascal Fua, and Vincent Lepetit. Tilde: a temporally invariant learned detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5279–5288, 2015.

[103] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[104] Vassilios Vonikakis, Dimitris Chrysostomou, Rigas Kouskouridas, and Antonios Gasteratos. Improving the robustness in feature detection by local contrast enhancement. In *2012 IEEE International Conference on Imaging Systems and Techniques Proceedings*, pages 158–163. IEEE, 2012.

[105] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[106] Aaron Wetzler, Ron Slossberg, and Ron Kimmel. Rule of thumb: Deep derotation for improved fingertip detection. In *British Machine Vision Conference*, 2015.

[107] Kyle Wilson and Noah Snavely. Robust global translations with 1dsfm. In *European Conference on Computer Vision*, pages 61–75. Springer, 2014.

[108] Jia Xu, Alexander G. Schwing, and Raquel Urtasun. Tell me what you see and i will show you where it is. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[109] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*, pages 467–483. Springer, 2016.

[110] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 3, 2018.

[111] Guoshen Yu and Jean-Michel Morel. Asift: An algorithm for fully affine invariant comparison. *Image Processing On Line*, 1:11–38, 2011.

[112] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhand Jain, and Tae-Kyun Kim. Bighand2.2m benchmark: Hand pose dataset and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[113] Amir R Zamir, Tilman Wekel, Pulkit Agrawal, Colin Wei, Jitendra Malik, and Silvio Savarese. Generic 3d representation via pose estimation and matching. In *European Conference on Computer Vision*, pages 535–553. Springer, 2016.

[114] Bernhard Zeisl, Torsten Sattler, and Marc Pollefeys. Camera pose voting for large-scale image-based localization. In *International Conference on Computer Vision*, December 2015.

[115] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19 (2):4–10, 2012.

[116] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[117] Yinlin Hu Fei Wang Pascal Fua Mathieu Salzmann Zheng Dang, Kwang Moo Yi. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In *Proceedings of the European Conference on Computer Vision*, 2018.

[118] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.

[119] Qiang Zhu, Ching-Tung Wu, Kwang-Ting Cheng, and Yi-Leh Wu. An adaptive skin model and its application to objectionable image filtering. In *IEEE Multimedia*, 2004.

[120] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *International Conference on Computer Vision*, 2017.