An Evaluation Of Features For Pose Estimation And Its Application to Free Viewpoint Video

Corentin Térence Eloi Chéron

A dissertation submitted to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Intelligent Systems)

Supervisors: Dr Konstantinos Amplianitis and Prof Aljosa Smolic

August 2018

Declaration

I, the undersigned, declare that this work has not previously been submitted as an exercise for a degree at this, or any other University, and that unless otherwise stated, is my own work.

Corentin Térence Eloi Chéron

28th August 2018

Permission to Lend and/or Copy

Acknowledgments

I would like to thank my academic supervisor, Dr Aljosa Smolic, for the support, the precious feedback and the inspiration throughout my dissertation.

I am heartily thankful to my advisor Dr Konstantinos Amplianitis for his advice throughout the year, for pushing me further and for guiding me through the CVPR conference in Salt Lake City.

I would also like to thank my second reader, Dr Gerard Lacey, for his constructive feedback.

Many thanks to the V-SENSE laboratory and the Volograms company for providing insights around free-viewpoint videos and a dataset with the associated camera poses.

And finally, I would like to thank my family and friends for their continued support.

Corentin Térence Eloi Chéron

University of Dublin, Trinity College
August 2018

An Evaluation Of Features For Pose Estimation And Its Application to Free Viewpoint Video

Corentin Térence Eloi Chéron, Master of Science in Computer Science

University of Dublin, Trinity College, 2018

Supervisors: Dr Konstantinos Amplianitis and Prof Aljosa Smolic

As consumers start using augmented and virtual reality with personal devices, new ways of creating high-quality 3D textured videos (free viewpoint videos) become necessary. Smartphone video clips of an actor from different viewpoints can affordably produce FVV even in outdoor environments. However, the cameras need to be accurately localised, and current methods require up to twelve cameras with small baseline angles to produce high-quality models. In this dissertation, we evaluate the capacity of different hand-crafted and learned features to estimate relative 3D pose, a critical step in the Structure-from-Motion reconstruction process. We developed a unified workflow based on COLMAP to compare those features with the highest rigour and match the FVV production pipeline. We evaluate various configurations for each of the SIFT, ASIFT, Sift-Affine, LIFT and SuperPoint features against new wide-viewpoint datasets with varying geometric complexity. The results show that the traditional hand-crafted features SIFT and SIFT-Affine are the most efficient to estimate wide-baseline camera poses regarding the number of keypoints. SuperPoint overtakes LIFT and reaches state-of-the-art performances in some configurations and shows an impressive match ratio in all the situations but fails to register cameras with the widest baseline. When applied to an FVV dataset, SIFT provides the best speed due to its refined implementation. Using the latest work in semantic segmentation, we evaluate the effect of matching feature regrouped semantically and show an improvement in the pose accuracy. The improvement observer from LIFT to SuperPoint and recent work on auxiliary learning applied to camera relocalisation show good promises in designing a new deep learning feature for wide-baseline applications.

Contents

List of Tables

List of Figures

List of Abbreviations

AR	Augmented Reality
ASIFT	Affine-SIFT
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CVPR	Computer Vision and Pattern Recognition conference
DTU	Technical University of Denmark
EPnP	Efficient Perspective-n-Point
FVV	Free-Viewpoint Video
GPU	Graphics Processing Unit
IR	Infrared
LIFT	Learned Invariant Feature Transform
MPEG	Motion Picture Experts Group
RANSAC	Random Sample Consensus Scheme
RGB	Red Green Blue
SfM	Structure-from-Motion
SIFT	Scale-Invariant Feature Transform

SLAM	Simultaneous Localisation and Mapping
SQL	Structured Query Language
TF	TensorFlow
TV	Television
USB	Universal Serial Bus
VR	Virtual Reality

Chapter 1

Introduction

Once reserved for professional applications, augmented reality (AR) is shifting to broader consumer usage. New headsets and smartphones are developed to allow the real-time rendering of 3D content inside real environments. Instead of being designed in computed software, real persons or objects can be captured to create a type of realistic 3D videos called Free-Viewpoint Video (FVV).

Traditional techniques to create this type of content with high quality require an expensive acquisition studio with numerous camera rigs and expensive hardware for acquiring and processing the content [7]. Recently, promising work [8] achieved excellent results in creating FVV in outdoor environments using only hand-held smartphones. In their set-up, up to a dozen video clips surrounding an actor are processed into FVV. The creation of FVV by simple smartphone video clip could allow the public to contribute by creating virtual content video games, social media or upcoming virtual worlds.

One of the first steps in creating FVV from hand-held video clips is to compute the relative pose of the cameras. This step is required to ensure quality in the triangulation process used to create the 3D texture models using multi-stereo vision. The traditional method to compute the relative pose of two cameras relies on finding geometric corresponding points in the two images. These correspondences are then processed using different geometric algorithms to estimate and refine the camera poses as well as the

3D locations of the correspondences in a process called Structure-from-Motion (SfM). SfM can estimate the pose and calibration of smartphone cameras as long as all the images have enough 2D-to-2D points correspondences with other images.

The 2D correspondences between images are usually found using local features extracted from the images. Those features are composed of a keypoint that describe the location of the feature and a descriptor that describe the local area around the keypoint. Features are identified in an image in areas of significant contrast change like corners or edges. Good feature extraction algorithms can extract keypoints from multiple images of the same scene corresponding to the same 3D locations even if the photos are taken from different viewpoints. In the same way, feature descriptors are designed to be as invariant as possible to changes in the camera pose or the environment. Correspondences between two images are created by pairing features that share similar descriptors.

This research will evaluate the most recent hand-crafted features and see how they compare to the learned features developed using deep learning. Unlike previous articles, this study will focus on camera pose estimation in wide-baseline scenarios (up to 180°) using smartphone pictures. Current FVV creation methods require up to a dozen cameras to reconstruct high-quality 3D models and finding new ways of estimating wide-baseline camera poses with high accuracy could lead to a reduction of this number of viewpoints. This study could also help find optimal usages of the currently available features in term of quality and computing time.

Our evaluation will compare multiple hand-crafted and learned features over a wide range of test datasets and real FVV smartphone videos. We will study the influence of the number of keypoints for each feature on the pose accuracy, the widest baseline angle that can be correctly estimated and the total processing time. Our goal is to find whether learned features can provide better performances in the pose estimation of FVV compared to hand-crafted ones.

We will construct a unified workflow that allows the fair comparisons of multiple existing features. We will evaluate them on newly acquired datasets that provide very wide-baseline angles and test our results on a real FVV dataset provided by the V-SENSE laboratory and the Volograms company.

This dissertation is structured as follow:

After the introduction to FVV and its applications, the **second** chapter details the photogrammetric techniques for pose estimation and the most recent features used in this evaluation. Some detail on the latest advances in end-to-end pose estimation using deep learning are also presented.

The **third** chapter outlines the unified workflow that allows the precise comparison of features in the different configurations and introduces the mathematical definition of the metrics and the datasets used for the evaluation.

In the **fourth** chapter, we present the results of the evaluation with a comparison of the different feature configurations and the analysis of bias due to different algorithm implementations. We also conduct tests on a real FVV dataset.

In the **fifth** chapter, we discuss the results by highlighting some limitations of our study and opening possible future research in the field of learned features for wide-baseline pose estimation.

Finally, the **sixth** chapter concludes our research.

Chapter 2

Related Work

Before defining the challenges of camera pose estimation, we will present the domain of the free-viewpoint videos and its applications in Augmented and Virtual Reality (AR/VR). We will then introduce a pose estimation technique that originates from photogrammetry and computer vision: Structure from Motion (SfM). We will detail the state of the art keypoints and descriptors. Finally, we will see the recent advances in using neural networks to estimate camera poses and how it can apply to free-viewpoint videos.

2.1 Free-viewpoint videos

In this section, we will define the concept of free-viewpoint video (FVV), see what applications can benefit from affordably producing them and we will detail the different steps required to create them.

2.1.1 Definition

In the field of digital content, traditional 2D movies are acquired with a single video camera. Recently, 360° videos became popular as smartphones can be used to visualise them naturally. In 360° video, the user can choose in which direction to look from a spherical recorded video. This type of video is recorded using a special camera, usually composed of multiple lenses. The camera can be moved during the recording, but, while playing, the user can only change the view direction, not the viewpoint. FVV correspond to a video where the user can not only look in different directions but can also change the position where the camera stands during playback. As in video games, FVVs can generate new views of a scene, based on the user input. FVVs require a photo-realistic 3D-model of the scene (background, objects or persons) and camera pose estimation is a key step in their creation. FVVs are stored as 3D textured model animations. They are also called 4D videos or holographic videos.

The next part will present some applications of FVVs.

2.1.2 Applications

3D television content

With the development of 3D television, the industry started the development of devices allowing the rendering of geometric videos [9] based on technologies that also allow FVV. These developments touched both hardware components [10] and full system for view synthesis [11]. In 2010, the concept of free-viewpoint television (TV) was introduced [12]. Smolic [13] presented the technology requirements to acquire and store FVV using new coding standards (MPEG-4 model-based FVV).

In 2018, the hype for 3D TV has fallen, and their sells are down as manufacturers stopped developing new models for multiple reasons [14]. However, the lack of content might have contributed to their demise. FVVs might, in the future, be displayed on 3D TV and offer a wider variety of content.

Sport events

Another application that is referred to frequently for FVVs is the re-transmission of sport events [15, 16, 17, 18]. FVV allow the user to watch sports games with more flexibility as before, by allowing them to choose the point of view and the location of the camera. For example, in a football game, the user can watch a goal with the same perspective as the goalkeeper.

Mixed reality

The domain of mixed reality is defined as the fusion of the physical reality and the digital reality [19]. It can be seen as a type of virtual reality that includes a digital version of real-world objects or persons. The company DoubleMe[1] proposes to create FVV of humans performing different actions. They call the captured representation holograms and use them for video games, 3D animations or 3D printing.

The Irish start-up Volograms[2], a spin-out from the V-SENSE laboratory at Trinity College Dublin, is providing technology for high-quality volumetric videos for applications in mixed reality. Volumetric videos, or volograms, are animated texture 3D objects. One of the key advantages of its systems is that it is capable of using different camera configurations, both indoor and outdoor [20]. One example uses a set of smartphones held by humans to record videos and generate FVV of an outdoor scene, which is not possible with other methods [8].

Recently, O'Dwyer et al. [21] presented a new way of experiencing a theatre play using FVV and 6 degrees of freedom sounds.

2.1.3 Free viewpoint video creation pipeline

Cameras setup

Most of the camera setup rely on fixed sets of cameras positioned around a scene to capture an actor. In [22], the authors have set up multiple cameras on the walls and the ceiling surrounding an object at the centre of the scene. The cameras are mounted on a rigid frame and are calibrated before doing the 3D reconstruction of the actor. The

[1]DoubleMe: `http://www.doubleme.me/`
[2]Volograms: `http://www.volograms.com/`

calibration consists in determining both the internal and external camera parameters, as presented in Section 2.2. [7] also use fixed cameras mounted on a rig around the scene. They use 53 RGB cameras and 53 infrared (IR) cameras with unstructured lamps to achieve the best accuracy in the reconstructions.

In this dissertation, we are interested in situations where smartphones are used for acquiring the video data, as presented in [8]. In this setup, the cameras are hand-held during the recording and are located around a target. The number of cameras used for the reconstruction goes from 6 to 12. This number and their positions impact the quality of the reconstruction. In this dissertation, we analyse the effect of the baseline angle between cameras on the pose estimation accuracy.

Pre-processing

Whereas [7, 23, 24] use a centralised system connected via Ethernet or USB to synchronise the different cameras, [8] require the individual smartphone videos to be manually synchronised to extract sets of frames taken at the same time for each cameras.

Pose estimation

Traditional photogrammetry requires input images taken while respecting overlap and minimal baseline parameters [25, 26]. For FVV, the input dataset is composed of multiple video cameras that have a nearly complete overlap in their view but can have large relative angular baselines with regards to the target. Also, for each camera, the video frames overlap with each other for the full length of the video. This configuration creates a very complex matching graph that renders SfM techniques inefficient. Indeed, the SfM algorithm proposed by [1], with the default parameters, requires an exponential duration to complete with regards to the number of frames of the FVV. As an example, 10, 20 and 30 frames with 12 cameras take respectively 8, 25 and 116 minutes. The full 120 frames would take days to complete. Another difficulty with pose estimation of FVV images is that successive frames are capturing moving elements (mainly the actors in the foreground) that need to be matched only with frames taken at the same time. Failure to do so would create geometric inconsistencies.

[8] overcomes those difficulties by taking only one frame per second, and by doing so, reduces the number of input images by 30. This subset of images is then much lighter, and their poses can be estimated using SfM techniques such as [1]. The foreground can be isolated from the static background using the segmentation employed for the 3D reconstruction. The remaining frames have their poses estimated using a perspective pose algorithm like the Efficient Perspective-n-Point (EPnP) [27] by using feature matching between successive frames. However, this approach relies on the fact that the angular baseline between the neighbouring cameras will be small enough to allow correct feature matching. It has been shown in [28], while traditional feature matching using SIFT gives high accuracy for two cameras with a small baseline angle, it fails in situations where the angle reaches 90° or higher.

3D reconstruction

Once the poses of the images taken at a specific frame have been estimated, the next phase consists in computing a 3D textured model of the target actor. The quality of this phase is dependent on the accuracy of the camera poses from the previous step.

In [8], the actor is first segmented from the background using a semi-supervised method relying on CNN [29]. Then a dense point cloud is generated using the segmentation mask as a probabilistic prior and multi-view stereo technique from [30]. The point cloud of the actor is combined with a voxel-based 3D model that is independently generated using Shape-from-Silhouette taking into account a 3D skeleton Zarean and Kasaei [18] of the human as a probabilistic prior, to overcome strong occlusions resulting from the sparsity of the cameras. It is also possible to add constraints to the 3D objects to follow non-rigid changes from frames to frames to obtain better consistency of the objects shapes Dou et al. [31].

Finally, all the images from a frame are blended to make a consistent and photo-realistic texture of the model [32, 33].

2.2 Camera pose estimation

2.2.1 Pinhole camera models

A camera projects a scene from a 3D environment (World) to a plane (the 2D picture). The pinhole camera approximates the complex optic behaviour of a camera by assuming that all rays of light go through a single point in space, located at the camera centre and intersecting the projection plane, located at the focal length (f) distance from the origin. A camera has internal (intrinsic) and external (extrinsic) parameters. The intrinsic parameters correspond to the optical properties of the camera, whereas the externals correspond to the position and orientation.

The pinhole projection, as defined in [34], transforms a point $\mathrm{X}_{cam} = (x, y, z)^\intercal$ to the image plane $\mathrm{X}_{img} = (fx/z, fy/z, f)^\intercal$.

This simplistic model is completed by adding additional sensor parameters: pixel skew (s), optical axis position on the image plane (principal point p) and difference in focal length on the x and y axis of the image plane. We use homogeneous coordinates to allow matrix multiplication:

$$\mathrm{X}_{img} = \begin{pmatrix} f_x x + sy + zp_x \\ f_y y + zp_y \\ z \end{pmatrix} = \begin{bmatrix} f_x & s & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \mathrm{K}[\mathrm{I} \,|\, 0]\, \mathrm{X}_{cam} \tag{2.1}$$

With X_{cam} the point in the *camera coordinate system* and K, the camera calibration matrix, defined as:

$$\mathrm{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

Now, we consider that the camera is at position C and has a rotation R with regards to the *world coordinate frame*. We can transform a point X from the *world coordinate* *system* to the camera frame with:

$$\mathrm{X}_{cam} = \mathrm{R}(\mathrm{X} - \mathrm{C}) \tag{2.3}$$

Together with the projection formula we get:

$$\mathrm{X}_{img} = \mathrm{KR}\,[\mathrm{I}\,|\,-\mathrm{C}]\,\mathrm{X} \tag{2.4}$$

Often, we introduce the term $t = -\mathrm{RC}$ to simplify the notation:

$$\mathrm{X}_{cam} = \mathrm{RX} - t \tag{2.5}$$

$$\mathrm{X}_{img} = \mathrm{K}(\mathrm{RX} + t) = \mathrm{K}\,[\mathrm{R}\,|\,t]\,\mathrm{X} \tag{2.6}$$

Finally, depending on their quality, camera lenses introduce distortion to the projected image. A first order correction models the lens by introducing radial distortions. This step happens after the projection. We consider a point $\mathrm{X}_{img} = (x, y)^\intercal$ projected using the linear projection (eq. 2.6), $\hat{\mathrm{X}}_{img} = (\hat{x}, \hat{y})^\intercal$ the corrected coordinates and (x_c, y_c) the principal point or centre of the radial distortion. We have from [34, Chapter 7]:

$$\begin{cases} \hat{x} = x_c + L(r)(x - x_c) \\ \hat{y} = y_c + L(r)(y - y_c) \end{cases} \tag{2.7}$$

With $L(r) = 1 + k_1 r + k_2 r^2 + k_3 r^3$ for a radial distortion model of degree three and r, the distance from the point X_{img} to the principal point $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$.

2.2.2 Pose estimation

The pose of the camera corresponds to its position and orientation in the World reference system. The pose corresponds to 6 degrees of freedom: 3 coordinates for position and 3 for orientation. In practice, to simplify computation and prevent gimbal lock, orientation is stored using quaternions. As a consequence, finding the pose corresponds to finding seven real values. In S*f*M, the cameras poses are estimated incrementally, starting with 2D-to-2D correspondences to estimate the initial scene (cameras and 3D points) and then using 3D-to-2D correspondences to estimate the poses of the following cameras. Bundle adjustment is used throughout SfM as a way to refine the poses, 3D points locations and camera parameters.

2D-to-2D

It is possible to compute the relative pose of two cameras with a minimum of 5 2D-to-2D correspondences from the two images. However, finding those corresponding points or matches is a difficult task subject to noise (see Section 2.3). As a consequence, [35] combines the five-point algorithm with a random sample consensus scheme (RANSAC) [36] applied on a larger set of matches. This method is effectively robust against outliers in the matches. This 5-point algorithm relies on a priori knowing the focal lengths of the two cameras and is usually the first step of SfM to initialise the reconstruction.

3D-to-2D

After the reconstruction has started, it is possible to iteratively estimate the pose of new cameras based on existing 3D points. 3D-to-2D correspondences are computed from matches with previous cameras: 2D matches from previous cameras are linked to the 3D points, and the 3D-to-2D algorithms take as input a set of 3D points and their projections in the new camera.

The Perspective-n-Point problem finds the position and orientation of a camera from a set of n 3D points and their projection in the image. Lepetit et al. [27] solves this problem by requiring at least four world points and has linear complexity.

Bundle adjustment

The 2D-to-2D and 3D-to-2D methods usually require the camera to be pre-calibrated. Those methods work with approximate intrinsic parameters (only an estimate of the focal length) and estimate the positions and orientations of the camera and 3D points.

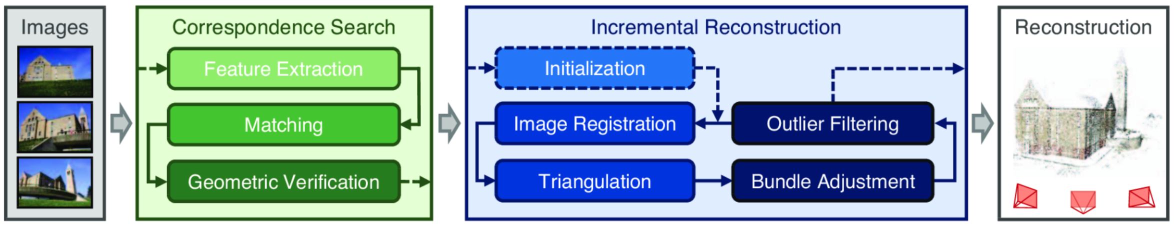


Figure 2.1: The Structure-from-Motion pipeline used by COLMAP from [1].

A method, called bundle adjustment, is used to compute the intrinsic parameters and refine even more the 3D points location and camera external parameters.

As presented in [37]: 'Bundle adjustment is the problem of refining a visual reconstruction to produce jointly optimal structure and viewing parameter estimates.'

Bundle adjustment is a nonlinear least-squares problem, for which the cost function is formulated to allow efficient removal of outliers [38]. Bundle adjustment also requires efficient use of the problem sparseness to ensure rapid convergence.

2.2.3 Structure from Motion

Structure-from-Motion is a process to reconstruct the 3D shape of a scene from a set of photos. In the process, both 3D points belonging to the scene and the camera parameters (extrinsic and intrinsic) are recovered. SfM can work with unordered images without any previous location information and can apply to both small objects and larger scenes like a city and can use ground or aerial images [25].

SfM relies on finding good correspondences between images and uses the pose estimation techniques seen in the part 2.2.2. After the first phase of correspondence finding, the reconstruction of the cameras is incremental by successively computing the pose of the cameras (registration), computing 3D points, and optimising all the parameters using bundle adjustment. See Figure 2.1 for a detail of the pipeline used by COLMAP [1].

2.3 Features

2.3.1 Keypoints and descriptors

As seen previously, the various methods for pose estimation rely on 2D correspondences between two images. Instead of matching images on a pixel level, which would have very high complexity, images are first transformed into a feature space. Features are composed of two elements: a keypoints extractor and a method to compute descriptors.

The keypoints extractor tries to find image points on contrast areas (usually on edges or corners) to find the same keypoints in different images of the same scene. Finding keypoints is essential in image matching, as it allows to get a simplified representation of the image with only a few thousand points.

For each keypoints, we then compute a descriptor which corresponds to a unique high dimension vector describing the feature. Descriptors are designed to minimise the distance (typically the L^2-norm) of vectors corresponding to the same point in the scene while being far from other points. Descriptors try to be invariant to changes of camera viewpoint (size, rotation, affine transformation) or the environment (lighting, noise).

In the following subsection, we will see different types of feature detectors and their descriptors: the traditional hand-crafted features and the more recent learned features.

2.3.2 Hand-crafted features

SIFT

The Scale-Invariant Feature Transform (SIFT) [2] is by far the most famous feature used in computer vision. Developed in 2004, it provides the best performances in term of matching accuracy in many situations [39, 40].

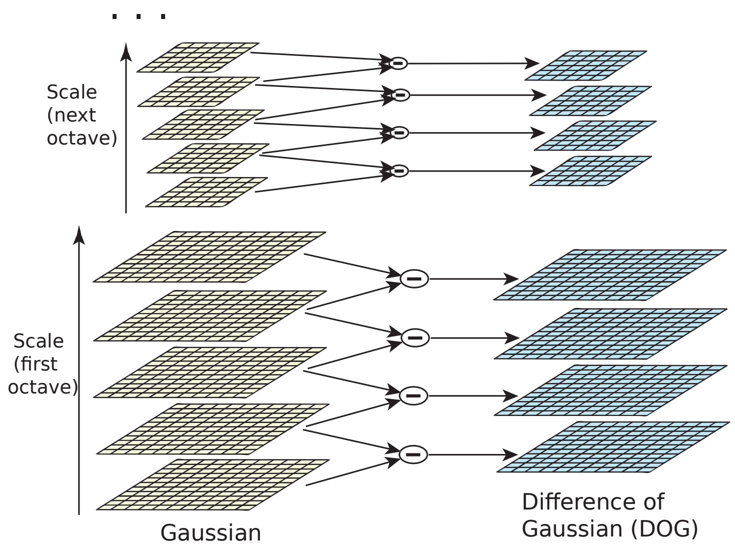


Figure 2.2: The scaling steps used by SIFT to generate scale invariant features [2].

SIFT uses Difference of Gaussian in a pyramid scale-space to find keypoints and histograms of gradients to compute the descriptors. SIFT is constructed to be invariant in scale, position and rotation. While changes in rotation of the input images are normalised (by computing the direction of highest gradient histogram), changes in scale are simulated by scaling the image (see Figure 2.2).

However, SIFT does not have any specific mechanism for handling changes in viewpoint in the input image.

ASIFT

The Affine-SIFT (ASIFT) [41, 42] feature was developed in 2009 to overcomes the limitation of SIFT by adding viewpoint invariance. ASIFT extends SIFT by simulating all the views corresponding to affine transformations. This affine transformation can be decomposed in changes of camera viewpoint by three rotations ψ, ϕ and θ (see Figure 2.3).

After simulating all the possible affine transformations (with a discrete step), the image has an area increased by a factor of 13.5. The same increase goes to the number of SIFT keypoints generated and the time required to generate them. The resulting matching

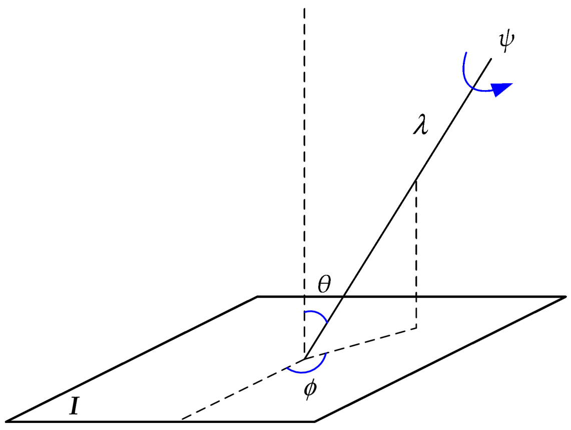


Figure 2.3: A geometric interpretation of affine decomposition from [3].

performances exceed the state-of-the-art methods [43, 44, 45, 46] for images with very high changes in viewpoint, however, the computation of the matches can be 180 times slower than for the traditional SIFT.

While initially applied to the SIFT features, the ASIFT technique of simulating viewpoints can also apply to other types of features.

Affine shape

The affine shape estimation is an alternative to the high complexity of the ASIFT approach that was presented by Mikolajczyk and Schmid [43] in 2002. Instead of simulating all the views corresponding to affine transformations, this method estimates the affine skew and stretch and normalises the images before extracting the descriptors.

Initially, the applied to the Harris detector [47], the method was called Harris-Affine. More recently, after SIFT was developed, the Harris-Affine descriptor was found not as good as ASIFT [42]. However, a promising implementation of this method applied to the SIFT algorithm is available in the VLFeat Library [48] used by COLMAP [1] to create the SIFT-Affine descriptor.

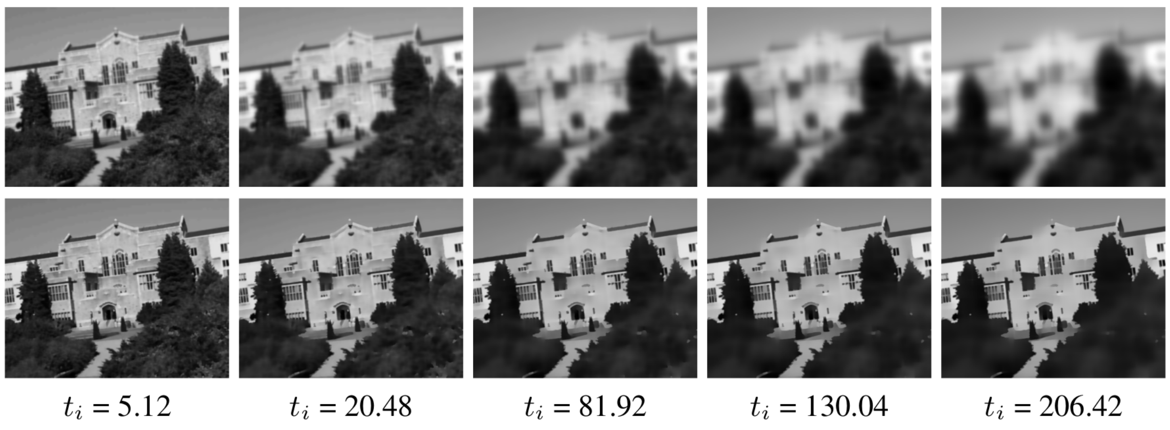


Figure 2.4: Comparison between Gaussian scale space (first row) and the nonlinear scale space technique used by the KAZE features (second row) for different evolution times t_i [49].

AKAZE

KAZE is a type of feature presented by Alcantarilla et al. [49] in 2012 that works in nonlinear scale spaces. It was designed to improve the accuracy of the localisation and the distinctiveness of the features by using an adaptive blurring algorithm that respects objects boundaries (see Figure 2.4).

The KAZE features present good results when matching deformable surfaces which make them candidates for matching changes in viewpoint. A more recent version of KAZE, called *AKAZE* [50] improves the speed of the feature detection and introduces a more efficient binary descriptor.

2.3.3 Learned features

With the advent of deep learning in computer vision, convolutional neural networks (CNN) have been applied with success to various tasks like image classification [51, 52, 53], object detection [54] or image semantic segmentation [55].

In this part, we will see two features generated using CNN that take advantage of the speed and the robustness of deep learning.

LIFT

The Learned Invariant Feature Transform (LIFT) [4] was developed in 2016 and is composed of three successive CNN that generate the keypoints locations, orientations and descriptors. This approach is the first to combine the three steps of detecting keypoints, computing their orientations and outputting their descriptors. Previous work in this domain worked on each aspects independently [56, 57, 58, 59].

The first network takes as input an image patch and detects a region of interest corresponding to the keypoint. Then the second network estimates the orientation of the keypoint, and finally, the third network computes a descriptor. The Figure 2.5 shows the full LIFT pipeline.

The LIFT networks are trained using different outdoor datasets used for 3D reconstruction. SfM using SIFT features is first ran on the datasets, and images patches are extracted in locations where SfM finds matches. Negative examples are also extracted for locations without SIFT features.

The LIFT features exceed state-of-the-art results in term of matching score on the *Strecha Fountain* [6] and the *DTU* [60] datasets which correspond to picture of a scenes from different viewpoints.

The code of the LIFT networks implemented using Tensorflow [61] is available online [3] and the authors provide weights for the networks trained on a dataset acquired at the Piccadilly Circus in London [62].

SuperPoint

SuperPoint [5] (2018) is the most recent learned feature presented at CVPR. SuperPoint is generated using a CNN that produces both keypoints and descriptors in a single forward pass. Designed for Simultaneous Localisation and Mapping (SLAM), the network is capable of running in real-time while performing better than state-of-the-art at repeatability and similarly to SIFT at matching.

[3]TF-LIFT is available at `https://github.com/cvlab-epfl/tf-lift`

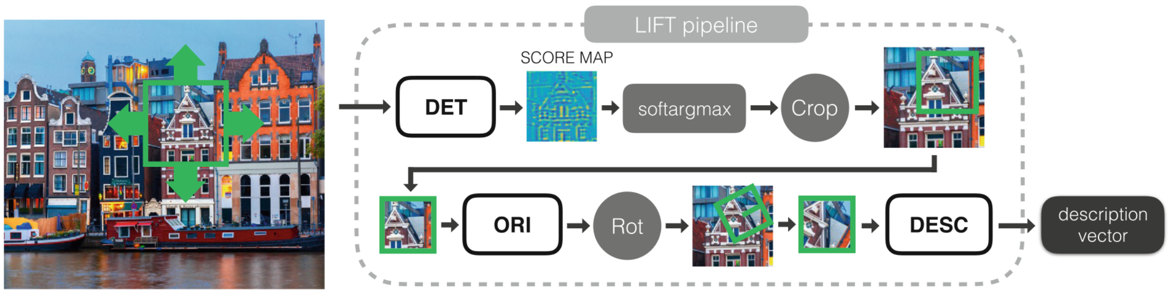


Figure 2.5: The LIFT feature extraction pipeline [4].

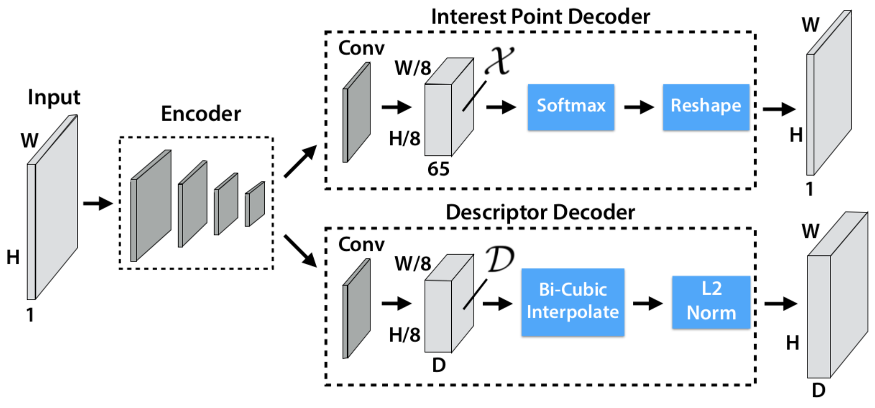


Figure 2.6: The deep network architecture used by SuperPoint from [5].

SuperPoint uses a CNN composed of two detectors connected to a shared encoder that compresses the input image. The first detector outputs a heat map of potential keypoints; the second output the corresponding descriptors (see Figure 2.6).

SuperPoint is trained using a self-supervised method. It uses synthetic images to initiate the keypoints detection, before going through a new approach, called *Homographic Adaptation*, that allows domain transfer and improves repeatability (see Figure 2.7).

A Pre-trained network and the TensorFlow code for inference are published[4].

[4]SuperPoint: `https://github.com/MagicLeapResearch/SuperPointPretrainedNetwork`

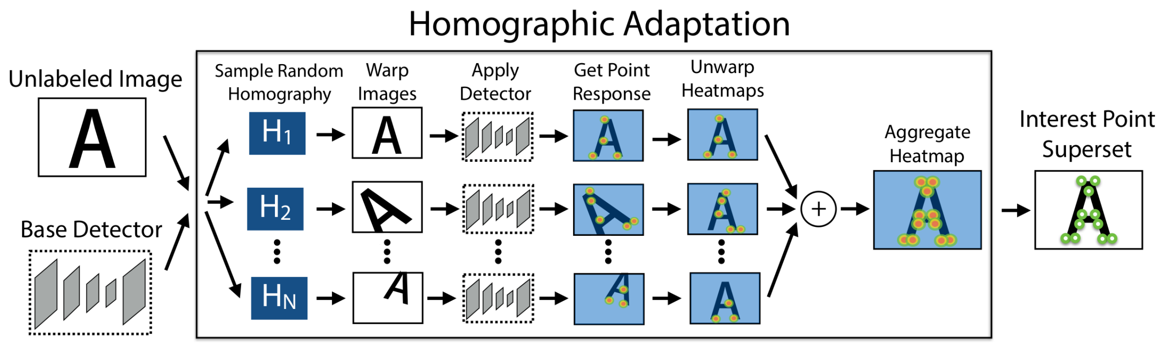


Figure 2.7: The novel method used for training the SuperPoint deep network [5].

2.4 CNN for pose estimation

In this section, we will present deep learning approaches to pose estimation, first for a previously known environment, and then the more generic approaches that do not rely on scene-specific learning.

2.4.1 Known environment

In the last two years, many researchers have focused on camera relocalisation using CNN. PoseNet Kendall et al. [63] was the first end-to-end network to estimate the 6 degrees of freedom of a monocular camera. The resulting pose error was around one order of magnitude higher than the state of the art hand-crafted method.

PoseNet triggered many research on camera localisation: using RGB-D cameras in night conditions [64], using LSTM Walch et al. [65], using synthetic data for training [66] or using temporal smoothness of a video stream to improve accuracy [67]. PoseNet was then improved [68, 69] and a new loss function allowed the pose error to be reduced by a factor two.

In 2017, CNN approaches for relocalisation reached the same accuracy than the hand-crafted state-of-the-art methods [70]. Recently, VlocNet and VlocNet++ [71, 72] were able to achieve a 67% improvement over state-of-the-art by combining learning the pose and the semantic segmentation. Learning multiple tasks at the same times allows better generalisation and reduces over-fitting of the network.

While presenting excellent pose accuracy, those techniques rely on acquiring a significant amount of data from the scene beforehand and are thus not compatible with the scenario of hand-held acquisition in an unknown environment.

2.4.2 On new scenes

Recently a few papers presented research on the domain of 3D reconstruction using deep learning. [73] presents a way to train a CNN in an unsupervised way to estimate depth from stereo cameras. Also on the depth topic, [74] computes high-quality disparity maps from multiple images with known camera poses.

In the domain of SLAM, where the successive camera movements are small, [75] presented an end-to-end visual odometry CNN, and [76] present a CNN for both depth and pose estimation from SLAM keyframes.

These approaches, while good advances, relies on small camera movements and cannot be used for wide-baseline pose estimation using multiple hand-held cameras.

Chapter 3

Method

In this chapter, we will present the workflow used to evaluate the different hand-crafted and learned features in wide-baseline situations. After presenting a unified workflow developed around an S*f*M software, we will see the development of a graphical user interface used to visualise the features and their matches. Then we will detail the process of extracting the metrics from the reconstructed scenes. Finally, we will introduce the four datasets used in this research.

3.1 Unified workflow

While traditional methods [4, 42] evaluate features on their matching capabilities level only, we decided to study the influence of the features in real FVV situations of camera pose estimation using the S*f*M software COLMAP [1].

3.1.1 COLMAP workflow

As seen in Section 2.2.3, COLMAP is an open-source S*f*M software developed initially by Schönberger and Frahm at the Department of Computer Science of the ETH Zürich. It regroups functionalities to extract and match SIFT features, camera pose estimation, 3D points triangulation and bundle adjustment, and finally, dense 3D multi-view reconstruction, which we will not use in this evaluation. We chose COLMAP as the main tool on which to develop our feature evaluation workflow as it presents multiple functionalities that are required by our study.

Firstly, COLMAP is a reference in the SfM open-source software, as a recent and very accessible one: It includes a graphical interface that is useful when debugging to visualise the camera poses and the 3D point clouds. It also has a command line interface to nearly all the functionalities, on a fine-grained level, with many parameters directly accessible. Besides, while written in C++, it is based on an SQLite database[1] which allows easy interface with any language. Furthermore, a set of helper functions in Python are available which allow easy creation, access, modification or extension of the SfM pipeline. Finally, The COLMAP documentation[2] is quite exhaustive.

Secondly, COLMAP integrates the SIFT features with many options. For the fastest execution, COLMAP uses a GPU implementation (SiftGPU[3]) of the SIFT feature extraction as well as a feature matching brute-force algorithm. SiftGPU is limited in the number of features for extraction and matching to the memory available on the GPU. On the NVIDIA 980 GTX graphic card used, the limit is $23,000$ features.

Finally, for more advanced feature algorithms, COLMAP uses the VLFeat library [48]. VLFeat provides an implementation of the Affine-Hessian shape algorithm seen in the Section 2.3.2. VLFeat algorithms run on the Central Processing Unit (CPU) only, and while they are optimised to use all the CPU cores, they are at least one order of magnitude slower than SiftGPU on a six-core i7-5820K.

3.1.2 Feature extraction and matching

As we need a unified way of estimating poses from 2D correspondences for all the features that we will evaluate, we insert the results of each feature matching into the

[1]SQLite is a public domain Structured Query Language (SQL) database engine: `https://www.sqlite.org/`

[2]The COLMAP documentation is available online at `https://colmap.github.io/`

[3]SIFT-GPU is available at `https://github.com/pitzer/SiftGPU`

COLMAP database before running the pose estimation. We also save the keypoints and descriptor information of all the features into the database for two reasons: it allows interrupting the process after each step, and also, reading the database to extract statistics and visualise the keypoints. To reach this goal, we have developed a Python interface to the database that we use for each step (see Figure 3.1).

Extraction

The feature extraction is the process of finding keypoints in an image and computing a local descriptor for each of them. The local descriptor is designed to be as invariant as possible to many acquisition or environmental factors.

For each feature, we set up a parameter that allows changing the number of features extracted. Ideally, we want to have exact control on this number, to be able to make the fairest comparisons. However, the actual implementation of the feature extractor varies in their possibilities. The OpenCV [77] implementation of ASIFT and the LIFT code are quite precise as they extract more features and then keeps only the n strongest. The COLMAP implementations of SIFT, both with VLFeat or with SiftGPU uses rough increment in the Gaussian scale space to find at least n features. Finally, SuperPoint and AKAZE do not come with any method to control the features directly. We control the number of feature by changing the input image resolution or the threshold respectively. Those methods provide a quite unstable number of keypoints within a dataset and between datasets. Some images with low contrast can end-up with fewer keypoints than expected. The Table 3.1 summarise the libraries used for extracting the features.

While most of the features take from a few seconds to a minute to extract the features for each of the datasets, the code of the LIFT feature extraction (see Section 2.3.3) is quite slow due to the patch algorithm and to the poor optimisation and takes around one minute per image. To reduce the total time required to evaluate the LIFT feature on hundreds of images, we implement a caching mechanism that saves the LIFT features for each dataset and number of features n. We used the *H5py*[4] Python library to save

[4]H5py (`www.h5py.org`) is a Python interface to the HDF5 (`www.hdfgroup.org`) binary storage format.

the LIFT keypoints and descriptors in a binary format on the disk. Caching the LIFT features saved a few days of computation time throughout the project.

Matching

The matching of keypoints is the process of finding corresponding features between multiple images. We use the L^2 and hamming distances for finding the closest descriptors when they are respectively n-dimension vectors or binary representation.

For each feature, we perform feature matching by following the ratio of the distance to the 2^{nd} closest point, as described in [2]. To ensure consistency in the matches, we use the same ratio parameter in the OpenCV code as in COLMAP with a value of 0.8. After finding correspondences with the ratio test, we perform a cross-check test to ensure that the closest match is the same in both directions.

For images with fewer than $15,000$ features, we use either the COLMAP or OpenCV brute-force matcher, as they provide the best accuracy with limited computation time. For more features, and especially for the ASIFT algorithm that can provide up to $500,000$ keypoints, we use the OpenCV interface to the Fast Library for Approximate Nearest Neighbours (FLANN) [78] which provides up to two orders of magnitude speed-up in the nearest neighbours search.

After the matches between two images are computed, we perform geometric filtering by keeping the matches that follow a fundamental matrix model with a threshold of 4 pixels. The fundamental matrix is found using the RANSAC [36] algorithm. We used the same threshold in COLMAP and the OpenCV implementation. To prevent unstable cases and as present in COLMAP by default, we set a minimum to the number of matches to fifteen inliers.

The Table 3.1 shows a summary of the matching algorithms used for each feature.

Feature	Library for extraction	Library for matching	Matching algorithm
SIFT	SiftGPU	SiftGPU	Brute Force
SIFT-Affine	VLFeat	SiftGPU/COLMAP	Brute Force
ASIFT	OpenCV	OpenCV	FLANN
AKAZE	OpenCV	OpenCV	Brute Force/FLANN
LIFT	lift-tf	OpenCV	Brute Force
SuperPoint	SuperPoint	Numpy	Brute Force

Table 3.1: A summary of the features used in this study, their libraries used for extraction and matching and the corresponding matching algorithms.

3.2 Semantic segmentation

3.2.1 Related work

Soon after applying deep CNN to image classification [51], new research was able to use the same machinery to the task of assigning a class to every pixel of an image: semantic segmentation. While the first implementation where working on a pixel level and were quite slow [79, 80], implementations using fully convolutional networks appeared soon after [81, 82].

Recently, semantic segmentation has been added to other localisation tasks and has been shown to improve the results. Indeed, performing multiple tasks helps a neural network to generalise the concepts more efficiently. For example, [83] improves the accuracy of a SLAM algorithm by providing per pixel class using a CNN. Zhao et al. [84] presents a fully end-to-end CNN that combines visual odometry and semantic segmentation using RGB-D images. Mustafa and Hilton [85] presents a framework for multi-view segmentation and 3D reconstruction and show that co-segmentation between multiple views improves the 3D reconstruction of complex scenes. Finally, recent work on multitask learning [71, 72] combines image semantic segmentation, visual odometry and global pose estimation in a joint CNN architecture and shows that adding semantic to the training divides by two the localisation error.

3.2.2 Improving local features

As seen previously, adding semantic information to a localisation task can improve the accuracy. In this work, we study the influence of adding semantic classes to feature descriptors on the accuracy of the camera pose estimation.

First, we compute a per pixel semantic segmentation using a CNN trained to identify 150 classes [55] for which the source code[5] is available for the framework PyTorch. We use a pre-trained network called ResNet-50-deepsup[6]. The network reaches 80% pixel accuracy on the ADE20K dataset[7].

Then, we classify each feature depending on the class of the nearest pixel of the centre of the keypoint.

Finally, when matching features from two images, we first group features belonging to the same class and then perform the same brute-force matching among each group.

3.3 Matches viewer

During the development of the unified workflow (see Section 3.1), it appeared a need to visualise information stored in the SQLite database. In response to this need, this study includes the development of a Graphical User Interface (GUI) for visualising the matches. This Viewer was developed using the library PyQt[8] and has the following functionalities:

- Connect the SQLite database and load all images and matches information.
- In a first list, select one of the images from the dataset and see a second list displays all the images matched to the first one.

[5]`https://github.com/CSAILVision/semantic-segmentation-pytorch`

[6]`http://sceneparsing.csail.mit.edu/model/pytorch/baseline-resnet50_dilated8-ppm_bilinear_deepsup/`

[7]`http://groups.csail.mit.edu/vision/datasets/ADE20K/`

[8]`https://sourceforge.net/projects/pyqt/`

- Show some statistics about the matches: the number of keypoints of each image, number of matches and geometrically verified matches
- Display both images side by side and show the keypoints, the matches and the geometrically verified overlaying the images.

The Figure 3.2 shows a screen capture of the GUI where we can see the menu with the options and the geometric matches of SIFT-Affine features for two images.

3.4 Pose estimation using COLMAP

We use the COLMAP S*f*M software for computing the pose of the cameras both for computing the ground truth and for evaluating the features. The datasets that we use (see Section 3.6) are all composed of images taken in a circle around a scene, with all the cameras pointing at a target (see Figure 3.4).

3.4.1 Ground truth

In this evaluation, we do not use any external method to compute the ground truth poses of the cameras. Instead, we rely on the densely sampled viewpoints to generate robust pose estimates that we use as references for our evaluation. We call those references *ground truth* to prevent confusion with the *reference camera* used to compute relative poses.

To compute the ground truth poses, we use the SIFT-Affine features with $10,000$ keypoints and then compute matches for all the input image pair combinations. This method provides a robust way to estimate the pose as both wide and small viewpoints matches are added to the optimised model. It is possible to run the match computation of all the pair combinations on the input images as our larger dataset (see Section 3.6) has only $n = 110$ images which correspond to $\frac{n!}{(n-2)!2!} = 5995$ combinations. Besides, to speed up the computation, we use the Graphics Processing Unit (GPU) implementation of the Brute Force matcher.

The Figure 3.3 shows the match matrix resulting from the computation of the matches for the ground truth of a dataset. The matrix, generated with COLMAP, shows the number of matches between all the image pairs where the rows and columns correspond to the pair images, ordered by angle computed from the normal of the planar surface. For the ground truth computation, the matrix is nearly full, with only the most extreme cameras having fewer than a dozen matching images. All the images have a total of more than a thousand matches with other images.

The last stage of the ground truth computation is to run the reconstruction process that includes pose estimation, point triangulation and camera intrinsic parameters refinement using bundle adjustment. The bundle adjustment is run until convergence is reached and, as a result, the relative pose estimations of the images are considered accurate enough to be considered ground truth.

3.4.2 Evaluation

In this study we consider the relative pose estimation using 2D correspondences of two images. We consider that, in a real process, both 2D-to-2D or 3D-to-2D correspondences can be used for the initial pose estimation and that the final accuracy is achieved using bundle adjustment. The camera intrinsic parameters are also refined in the process. To evaluate the robustness of the features in pose estimation, we evaluate the pose of a test camera i relative to a reference camera that is either normal to the wall or have an angle of 45° (see Figure 3.4). We compute the matches of every camera i with the reference camera. Figure 3.5 shows resulting match matrices.

In our unified workflow, we generate a list of the match pairs which is fed to the Feature Matching block (see Figure 3.1). After the matching is finished, COLMAP runs the pose estimation and computes the poses of all the cameras at once. This process allows evaluating quickly multiple sets of parameters while using only the correspondences between each image and the reference image.

At the end of the bundle adjustment, the cameras pose, intrinsic parameters and the 3D points minimise the reprojection error between the cameras [37] and are ready to be analysed.

3.5 Metric extraction

Now that the poses have been computed, we can extract interesting information from the SQLite database for the features and matches and from a binary output set of files called the *sparse model*.

3.5.1 Pose Error

The first metric that we evaluate on the feature is the relative pose error between two cameras. This error is composed of the relative translation error t_{error} and the relative orientation error R_{error}. To compute this error, we extract the camera pose from the output model. We convert the binary *spare model* into a text format using the model converter in COLMAP. The resulting text files contain the camera poses for all the images that were correctly registered. The COLMAP workflow rejects images with too few or noisy matches before writing them to the output files.

We call the evaluation model, the set of camera poses computed using the feature and parameters that we are evaluating. Similarly, we call the ground truth model, the camera poses computed using the ground truth method (see Section 3.4.1). We compare the relative pose between two cameras (a and b) from the evaluation model to the relative pose of the same cameras in the ground truth model. For each image, the camera pose is saved as a quaternion (convertible in a rotation matrix) and a translation vector equivalent to R and t as defined in Equation 2.5. We write R_{ea} and t_{ea} the rotation matrix and translation in the camera reference for the evaluation camera a. We use the same notation for camera b, and we substitute e by gt for the ground truth.

For each camera, we compute the camera position in the world coordinate using:

$$t_W = -\mathrm{R}^T t \tag{3.1}$$

Then, we compute the relative translation from camera a to b in the ground truth and evaluation models as:

$$\begin{aligned}\Delta t_{W\,e} &= -\mathrm{R}_{eb}^{\ T} t_{eb} + \mathrm{R}_{ea}^{\ T} t_{ea} \\ \Delta t_{W\,gt} &= -\mathrm{R}_{gt_b}^{\ T} t_{gt_b} + \mathrm{R}_{gt_a}^{\ T} t_{gt_a}\end{aligned} \tag{3.2}$$

Finally, we compute the relative translation as the following scalars:

$$t_{error} = \left\|\Delta t_{W\,e} - \Delta t_{W\,gt}\right\|_2 \tag{3.3}$$

For the relative orientation error, we first compute the relative rotations:

$$\Delta \mathrm{R}_{gt} = \mathrm{R}_{gt_a} \mathrm{R}_{gt_b}^{T} \tag{3.4}$$

$$\Delta \mathrm{R}_{e} = \mathrm{R}_{ea} \mathrm{R}_{eb}^{T} \tag{3.5}$$

From which we compute the orientation error as:

$$\mathrm{R}_{error} = \left|Angle(\Delta \mathrm{R}_e (\Delta \mathrm{R}_{gt})^T)\right| \tag{3.6}$$

where *Angle* is the function that extracts the rotation angle from a rotation matrix.

3.5.2 Matches

The second metric that we evaluate is the match ratio which corresponds to the ratio between the number of matches and the number of keypoints extracted by the feature. The match ratio is a metric that summarises the capacity for the feature to detect the same keypoints in images from different viewpoints and to match them correctly.

From the SQLite database, we extract the number of keypoints for each image and the number of geometrically verified matches for all the image pairs. We compute the match ratio $mr(a, b)$ between two images a and b as:

$$mr(a, b) = \frac{valid(a, b)}{min(\text{keypoints}_a, \text{keypoints}_b)} \tag{3.7}$$

With $valid(a, b)$ the number of geometrically verified matches between the image a and *b*.

3.6 Datasets

For this study, we want to have datasets that can challenge to most viewpoint invariant features and be similar to FVV situations with densely sampled viewpoints, taken outdoor with smartphones and with relative baselines going up to 180°. However, previous research on wide viewpoint focused either on angles smaller than 80° [60] or on indoor figurine scenes only [86]. To match our needs, we selected a public dataset and chose to acquire three new datasets with varying level of difficulties.

3.6.1 Public dataset

The first dataset that we will use in this study is a set of photos of a fountain taken with a digital camera with a resolution of 6 Mpix [6]. The dataset is composed of 25 images and span over 115°. See Figure 3.6 for a 3D representation of the dataset *Fountain*. This dataset can be considered easy as it presents a wide planar surface with stones that create a very sharp texture.

3.6.2 Own datasets

We acquired three different datasets using two different hand-held smartphones to create images close to those encountered in FVV.

Two datasets were acquired in Maynooth, in front of the walls of old buildings (see Figure 3.7). The first one, called *Wall*, is taken in front of a stone wall and is also considered simple as the texture is sharp with many corners. The second one, called *Ivy*, is a set of pictures of the wall of a building covered with Ivy. Both datasets span 180° and are taken from two different distance from a reference target. The *Ivy* dataset is considered of medium difficulty as the Ivy leaves have very complex shapes that are very sensitive to viewpoints. Each dataset is divided in two subset called *close* and *far* respectively.

The last dataset, called *Statue*, is a set of 110 frames extracted from a movie of a statue taken in the park Carton House[9]. The smartphone video frames span 360° around the statue at a relatively constant distance and offer the smallest baseline angle step of around 3.6°. This dataset differs from the previous as there is no vertical planar surface and most of the texture is on the floor with gravels and stones. As the frames were extracted from a movie taken while walking, some of them present motion blur. This dataset is considered to be the hardest of the set and pushes the limits of wide-baseline situations where two cameras can have baseline angles of up to 180°.

[9] `www.cartonhouse.com`

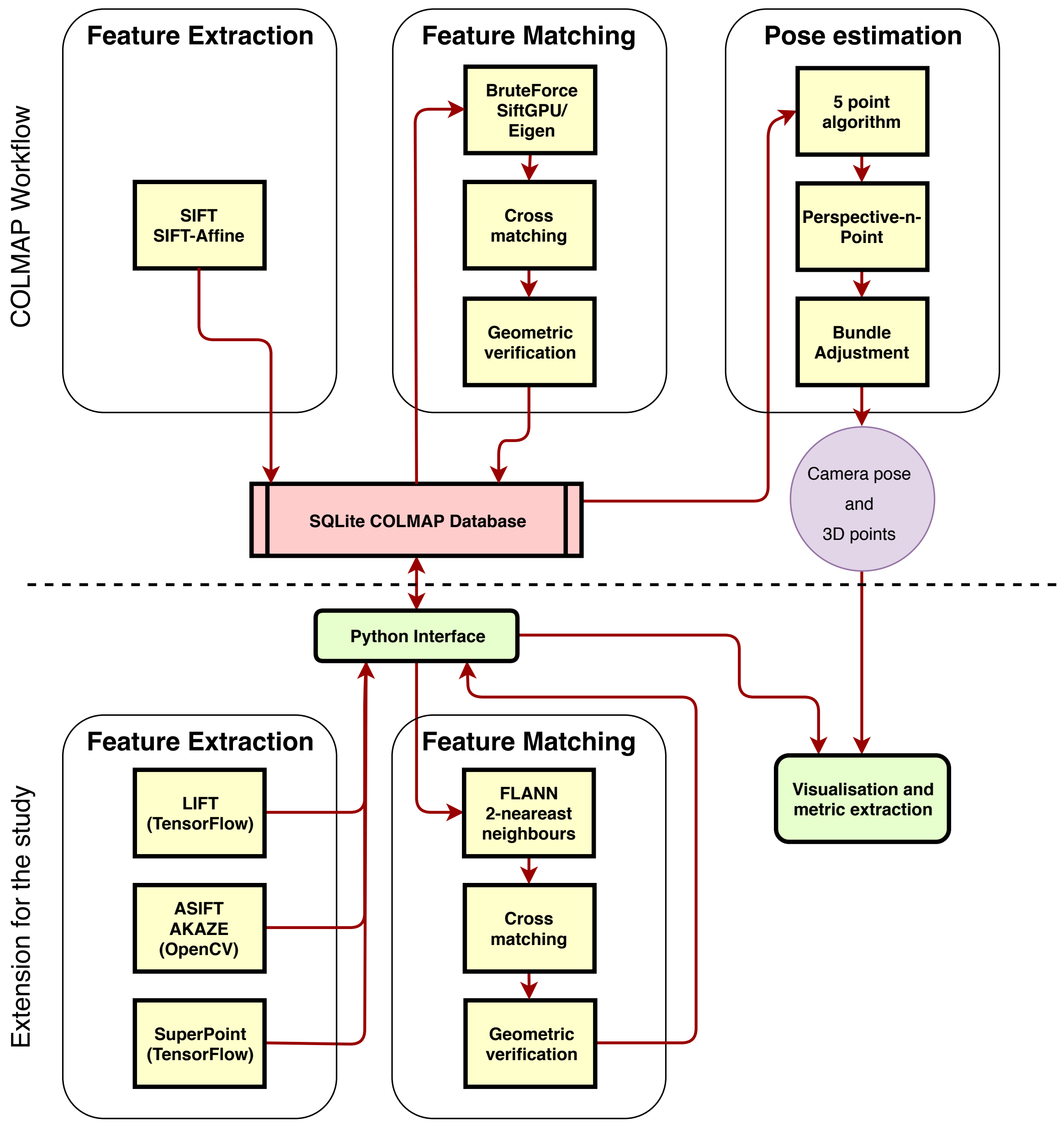


Figure 3.1: The COLMAP workflow for feature extraction and pose estimation is centred on the SQLite database. We extend by following the same scheme and adding new hand-crafted and learned features using OpenCV and TensorFlow.

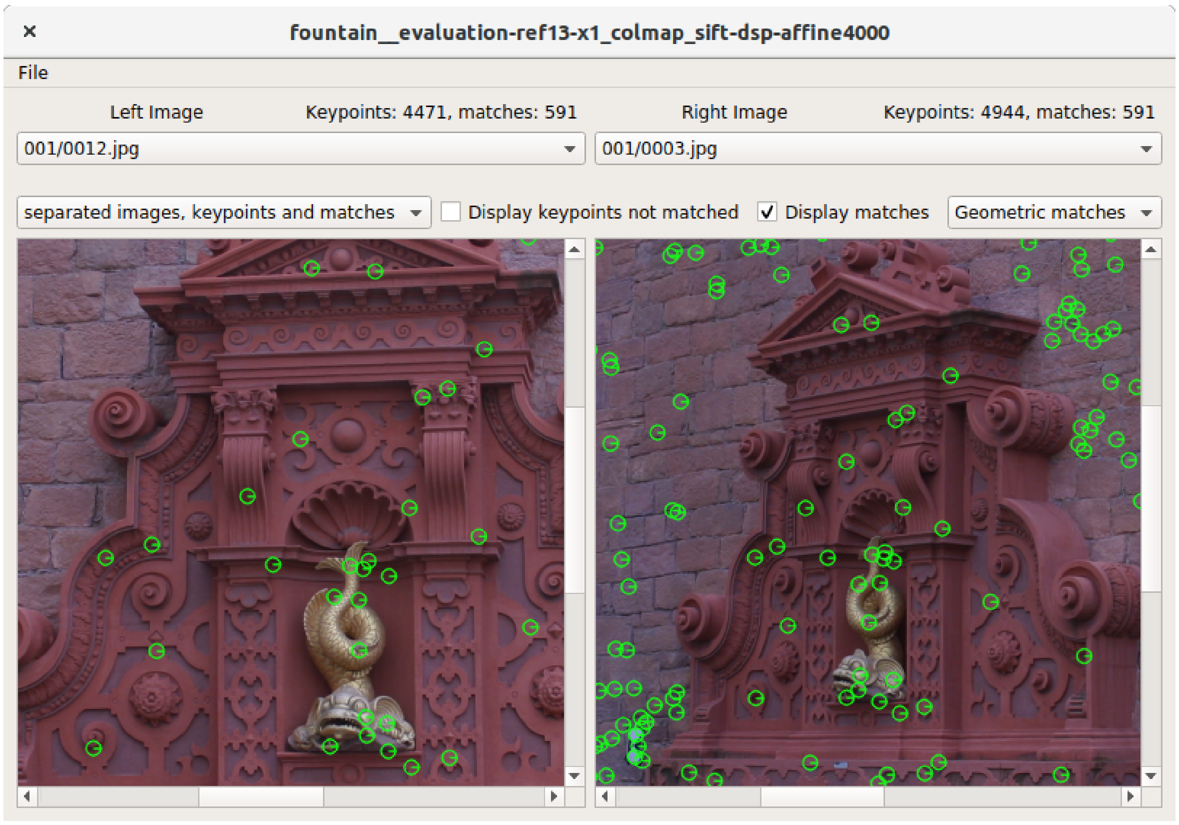

×
fountain__evaluation-ref13-x1_colmap_sift-dsp-affine4000
File
Left Image
Keypoints: 4471, matches: 591
Right Image
Keypoints: 4944, matches: 591
001/0012.jpg
001/0003.jpg
separated images, keypoints and matches
Display keypoints not matched
Display matches
Geometric matches

Figure 3.2: The Graphical User Interface developed to visualise features and their matches. The green circles correspond to the geometrically validated matches between SIFT-Affine features for the left and right images.

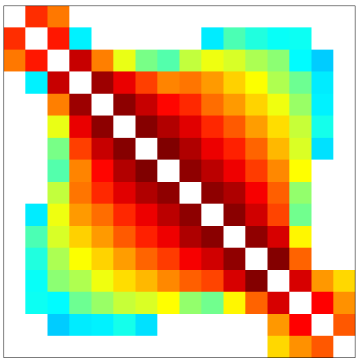


Figure 3.3: An example of a match matrix resulting of the exhaustive matches computed for the ground truth pose estimation of one of the datasets. Each cell of the matrix corresponds to the number of matches for a pair of images corresponding to the row and column. The colours code the number of matches from blue, for the lowest, to red, the highest.

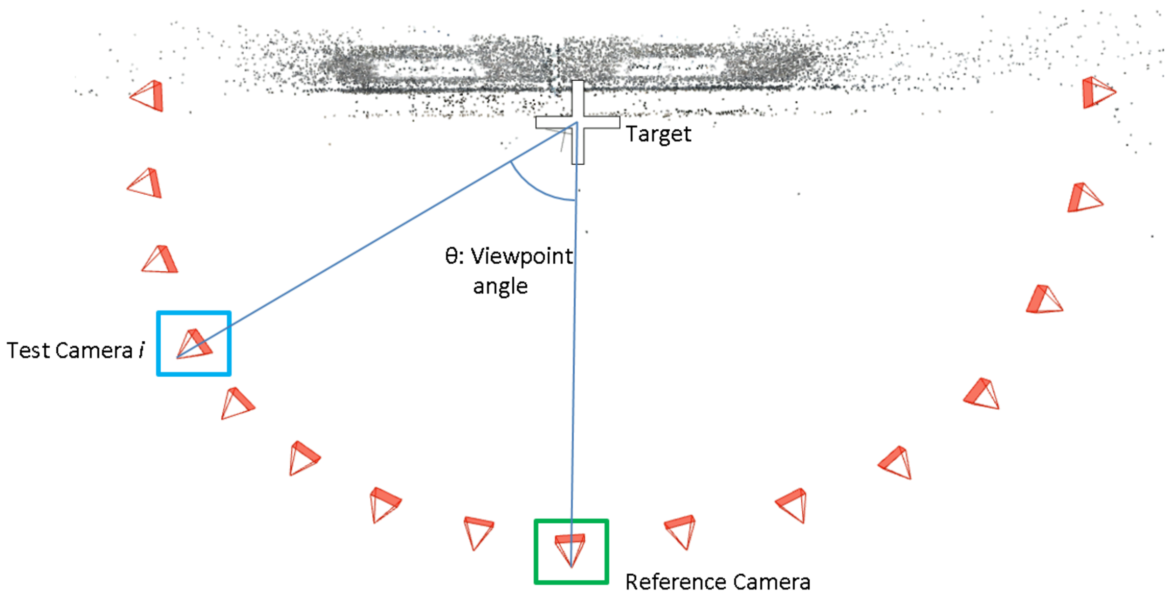


Figure 3.4: A scene representative of all the datasets that we study for Free-Viewpoint Video where a set of photos are taken around a planar surface (here a wall) pointing at a target. The relative pose is computed between a camera i and a reference camera chosen to be normal to the wall or with a 45° angle.

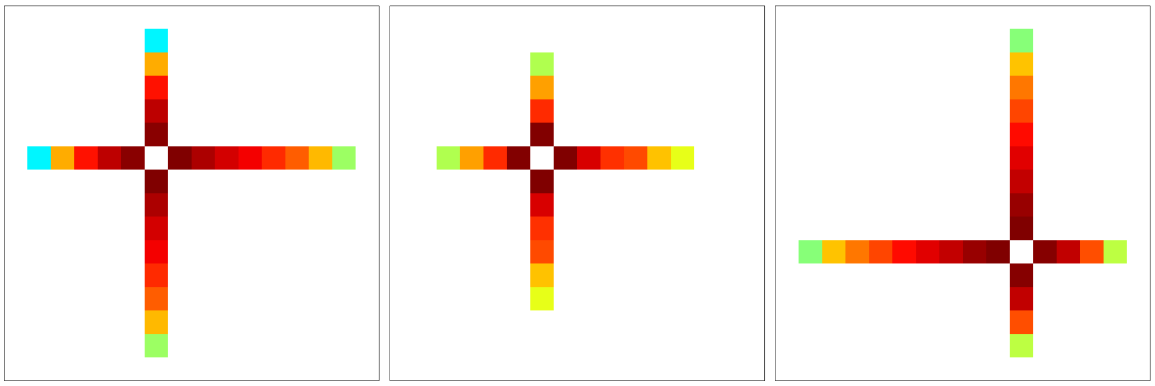


Figure 3.5: The match matrix obtained is the evaluation of features. The cross shape of the matrix indicate that the matches are computed only with a reference camera. The left matrix corresponds to the normal reference camera with a high number of keypoint where all but two cameras have matches. The same situation with fewer keypoints (middle) and another reference camera (right) are also presented.

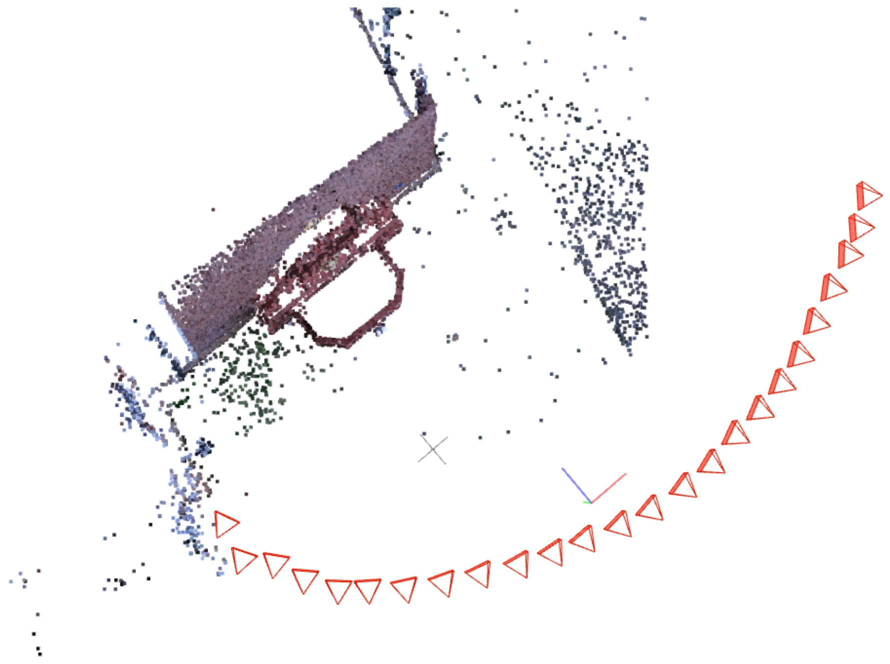


Figure 3.6: The 3D representation of the *Fountain* dataset from [6], comprising 25 images taken around a fountain attached to a stone wall. The 3D view of the sparse point cloud is taken from the zenith and the image locations are the red triangles.

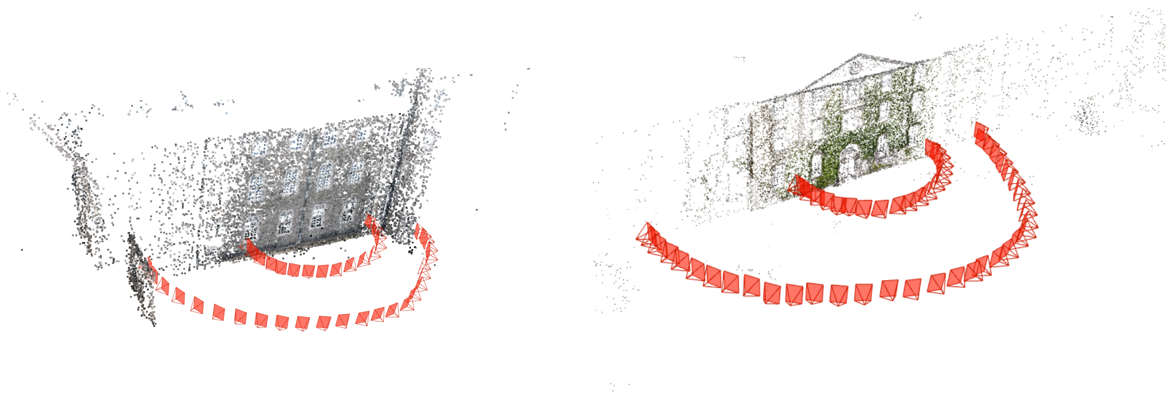


Figure 3.7: The *Wall* (left) and *Ivy* datasets taken in Maynooth comprise 41 and 56 images respectively and span two half circles around a target point on two buildings.

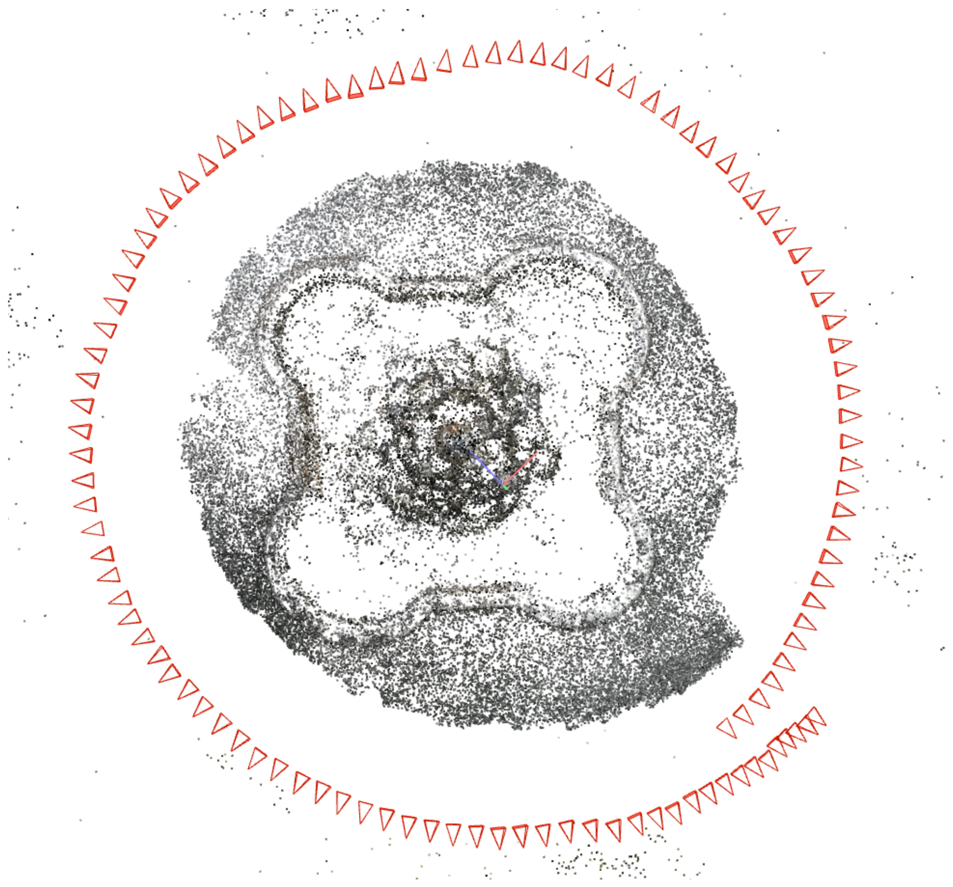


Figure 3.8: The *Statue* dataset is a set of 110 frames extracted from a movie taken around a statue. The point cloud shows the texture areas where the features were extracted and matched during the processing of the ground truth Structure-from-Motion.

Chapter 4

Results

In this chapter, we present the results of the various evaluation configurations. After a summary of the evaluation parameters, we will compare different implementations of SIFT that are used for either ASIFT, SIFT-Affine or when using the GPU for feature extraction and matching. Then, we see the first accuracy results on one dataset before analysing averaged results for all the datasets. We continue by showing the impact of the semantic classification of features. Finally, we apply the best feature configurations to a real FVV dataset.

4.1 Evaluation parameters

4.1.1 Features

For each of the following features: SIFT, ASIFT, SIFT-Affine, LIFT and SuperPoint, we study the influence of the number of keypoints to the pose estimation.

Table 4.1 details the features parameters and the number of keypoints extracted for each of them. We can see that LIFT is the most stable, followed by SIFT, SIFT-Affine and ASIFT. AKAZE and SuperPoint are the most unstable feature. Overall, all the implementations apart from LIFT are unable to compute a precise number of keypoints.

The SIFT implementations overestimate the number of keypoints, which results in higher matching quality and computation time whereas AKAZE and SuperPoint can underestimate dangerously the number of keypoints resulting in complicated matching. Good implementations are needed to overcome image change in contrast, viewpoint and luminosity and to provide a stable number of keypoints. Knowing this limitation, we will plot the real number of keypoints averaged for all the images of a dataset in the subsequent evaluations.

4.1.2 Datasets and reference camera

As the Ivy and Wall datasets are both split in two for evaluation, grouping images at a similar distance from the target, the 31 feature configurations from Table 4.1 are evaluated against six datasets with two reference cameras (90° and 45° from the planar surface) resulting in 372 evaluation configurations and around ten thousand relative pose evaluated.

4.2 Comparing SIFT implementations

In our evaluation, we use different implementations of SIFT for low and high numbers of SIFT features (SIFTgpu and COLMAP/VLFeat), ASIFT (OpenCV) and SIFT-Affine (COLMAP/VLFeat) which are extensions of the original SIFT. In this section, we study the difference between implementations of the original SIFT algorithm in OpenCV, SIFTgpu and COLMAP/VLFeat. The comparison tests both the feature extraction and the matching and uses the Ivy dataset with a reference camera normal to the wall.

The Figure 4.1 shows the results in term of match ratios, orientation errors of the estimated poses and computing times. While OpenCV outperforms the two other in matching ratio, it aligns images with lower baseline angles; this effect might originate from the way of selecting features when limiting the number of keypoints and is not present when using the highest number of keypoints. SIFTgpu uses some simplifications that result in slightly higher error rates compared to VLFeat. Finally, there is a clear

Table 4.1: The detailed list of all the feature configurations evaluated in this study. For each feature, the parameter indicates how the number of feature is controlled and the minimum and maximum number of keypoints extracted in all the images from the four datasets are computed to show the stability.

Name	Feature	Parameters	Min/Max features
SIFT250	SIFT	$max\_feature = 250$	252/670
SIFT500	SIFT	$max\_feature = 500$	500/1,109
SIFT1000	SIFT	$max\_feature = 1,000$	1,000/2,154
SIFT2000	SIFT	$max\_feature = 2,000$	2,000/4,262
SIFT4000	SIFT	$max\_feature = 4,000$	4,000/8,508
SIFT10000	SIFT	$max\_feature = 10,000$	10,000/22,589
SIFT40000	SIFT	$max\_feature = 40,000$	15,513/96,165
SIFT-Affine250	SIFT affine shape	$max\_feature = 250$	244/392
SIFT-Affine500	SIFT affine shape	$max\_feature = 500$	498/808
SIFT-Affine1000	SIFT affine shape	$max\_feature = 1,000$	980/1,729
SIFT-Affine4000	SIFT affine shape	$max\_feature = 4,000$	3,981/6,675
SIFT-Affine10000	SIFT affine shape	$max\_feature = 10,000$	10,028/18,845
SIFT-Affine40000	SIFT affine shape	$max\_feature = 40,000$	13,163/76,919
ASIFT500	ASIFT	$max\_feature = 500$	20,758/21,504
ASIFT1000	ASIFT	$max\_feature = 1,000$	40,338/42,986
ASIFT4000	ASIFT	$max\_feature = 4,000$	89,850/171,759
ASIFT10000	ASIFT	$max\_feature = 10,000$	92,653/428,058
AKAZE0.5	AKAZE	$threshold = 0.0005$	6,340/65,840
AKAZE	AKAZE	$threshold = 0.001$	3,038/50,052
AKAZE2	AKAZE	$threshold = 0.002$	1,375/30,176
AKAZE4	AKAZE	$threshold = 0.004$	485/10,801
AKAZE6	AKAZE	$threshold = 0.006$	214/4,508
AKAZE8	AKAZE	$threshold = 0.008$	108/3,124
LIFT500	LIFT	$num\_keypoint = 500$	445/495
LIFT1000	LIFT	$num\_keypoint = 1,000$	910/987
LIFT4000	LIFT	$num\_keypoint = 4,000$	3,727/3,911
LIFT10000	LIFT	$num\_keypoint = 10,000$	4,266/8,916
SuperPoint4	SuperPoint	$image\_size = (640, 480)$	365/1,535
SuperPoint5	SuperPoint	$image\_size = (800, 600)$	535/2,227
SuperPoint7	SuperPoint	$image\_size = (1120, 840)$	900/3,589
SuperPoint9	SuperPoint	$image\_size = (1440, 1080)$	1,414/5,065

difference of computation time, with nearly one order of magnitude between SIFTgpu and OpenCV and also between OpenCV and VLFeat.

We can conclude that COLMAP SIFTgpu and VLFeat provides relatively similar results and that OpenCV implementation also provides similar performances for high numbers of keypoints as it is the case for ASIFT.

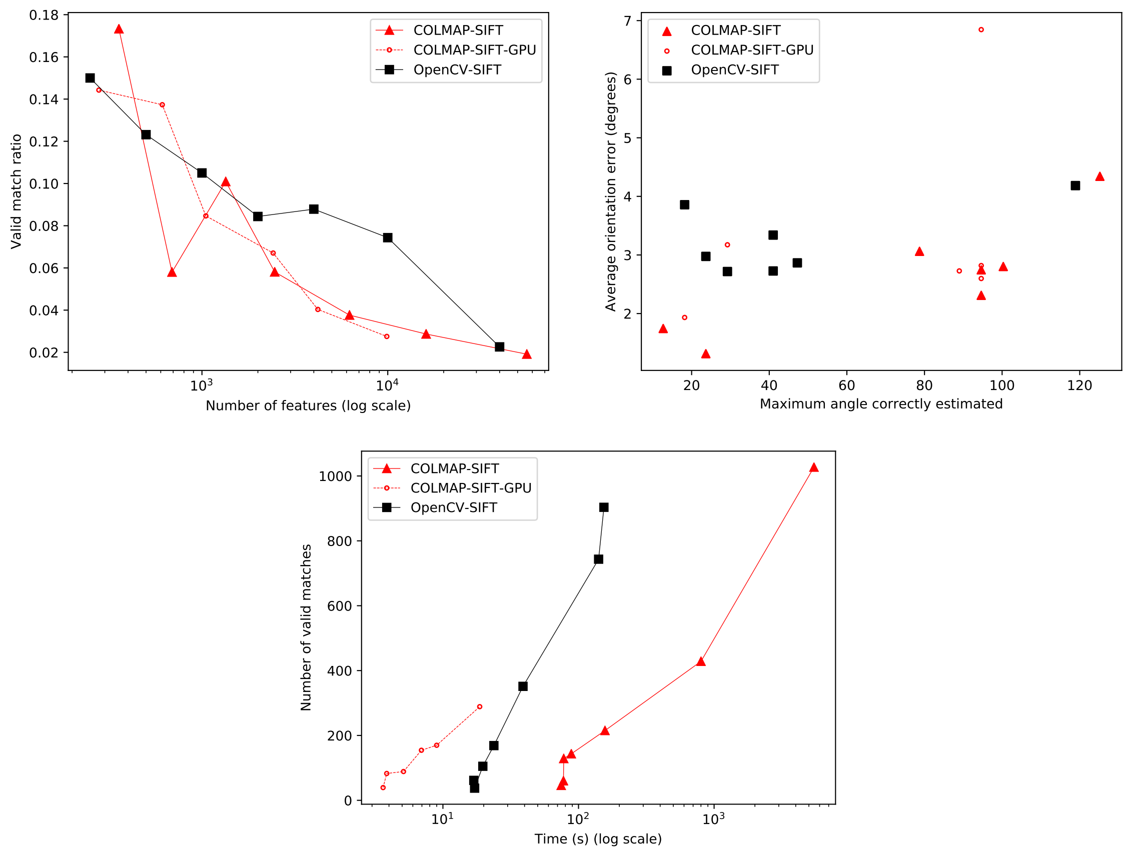


Figure 4.1: Comparison of the SIFT implementations in COLMAP (VLFeat and SIFT-gpu) and in OpenCV applied to the Ivy dataset for pose estimation, using a reference camera normal to the wall. The ratio of valid matches is plot against the number of features extracted (top left). The relative orientation error is shown in function of the maximum viewpoint angle correctly estimated (top right). The feature extraction and matching is evaluated by comparing the time taken to compute different numbers of valid matches (bottom).

4.3 Results on the *Wall* dataset

Before running all the evaluations configurations, we first select one dataset and one configuration per feature in order to find the most interesting metrics.

Figure 4.2 shows the results obtained by choosing the feature configuration closest to $4,000$ keypoints for each type of feature on the *Wall-close* dataset. For the ASIFT dataset, we choose 500 keypoints as the parameter before doing the affine augmentation, which results in around $20,000$ keypoints. In term of match ratio, SuperPoint performs best for all angles, followed by Sift-Affine. ASIFT is the less efficient feature with the lowest match ratio for all relative viewpoint angle.

When looking at the relative position error, we see an error directly proportional to the viewpoint, which also corresponds to the relative camera distance. This error seems to come from error in the estimation of the focal length in the intrinsic parameter of the camera. Indeed, in the evaluation situation, the focal length is not constrained enough and can be optimised to different values by then bundle adjustment. As a result, the scale of the 3D scene can change from one reconstruction to the other and renders the relative distance between two cameras inadequate as a metric.

From the match ratio and the relative orientation error, we find that relative angles between $-30°$ and $30°$ are not very relevant as most of the features perform very well in this range. For the rest of the study, we decide to compute the average of the metrics by keeping only relative angles for which the absolute values are above 30°.

4.4 Comparison of all the feature configurations

In this section, we will compare all the feature configurations (see Table 4.1) against all the datasets using two reference cameras at 90° and 45° from the planar surface. The metrics will only include cameras with relative baseline angles larger than 30° from the reference to emphasis on the wide baselines.

4.4.1 Match ratio

In this part, we compare the ratio between the number of matches and the number of keypoints averaged to all image pairs evaluated. The Figure 4.3 shows the results as a function of the number of keypoints for both reference angles. For every feature, the match ratio follows a decreasing trend with the increasing number of features. While ASFIT has the lowest match ratio with a stable 1%, SuperPoint presents an impressive ratio with nearly the double of SIFT for both reference cameras.

4.4.2 Average relative orientation error

The accuracy of the relative pose estimation is evaluated using the relative orientation error. Figure 4.4 presents the relative orientation error for all the feature configurations. For both reference angles, it is difficult to extract clear trends. There is a bias in those graphs as the error is measured only for images whose pose is estimated. The images with no match or that do not have enough inliers in the bundle adjustment are removed.

As a result, the more keypoints are extracted for a feature, the more images with wide-baseline angles are considered in the averaged error. We can also consider that wider baseline image pairs have weaker feature matches and thus larger pose estimation error. This hypothesis confirms the trend that is observed.

To improve the analysis of the error, instead of drawing it as a function of the number of keypoints, we will use the maximum baseline angle correctly estimated (Figure 4.5). There is a general trend that all features have an orientation error proportional to the maximum angle estimated. On both the 90° and 45° references, SIFT, SIFT-Affine and SuperPoint outperform the other features.

4.4.3 Maximum angle estimated

To understand more precisely the capacity for a feature to compute a correct wide-baseline pose estimation, we compare the maximum baseline angle estimated as a function of the number of features in Figure 4.6. On both reference angles, we can see a hard limit with the shape of a logarithmic curve. All features need an exponential number of features to estimate the pose of the highest baseline angles correctly. While SIFT and SIFT-Affine are the best for less than $1,000$ features, SuperPoint has the highest values for a few thousand keypoints. The AKAZE feature displays results in the lower range for all number of features. Finally, SIFT, SIFT-Affine and ASIFT provide the highest results when using more than a few tens of thousands of keypoints. Overall, handcrafted features are well suited for a wide range of feature number and reach up to 110° of camera baseline angle.

4.5 Semantic classification

Using the semantic segmentation algorithm presented in Section 3.2, we classify every pixels of the dataset images and regroup the features in the same classes based on their location. The Figure 4.7 shows four examples of the resulting classifications, for the simple *Wall* and the more complex *Statue* datasets.

The Figure 4.8 presents a comparison of the pose estimation metrics with and without the feature semantic classification. The classification can either slightly improve the accuracy or, for two feature configurations, more than double the error. For all the configurations, the classification decreases the match ratio and the maximum angle estimated. This effect comes from the fact that the classification is only reducing the number of matches and as a result improves the average matches accuracy.

Overall, the effect of the semantic classification of the features provides a slight improvement in the pose estimation accuracy.

4.6 Application to free-viewpoint video

In this chapter, we evaluate the results of the features for pose estimation on a real FVV dataset.

4.6.1 FVV dataset

The dataset *Rafa Dance* (provided by Volograms) is composed of twelve different hand-held smartphones videos of an actor dancing in an outdoor city environment. Figure 4.9

shows the first frames of the twelve video clips. The images have a Full HD resolution and span on a half circle around an actor dancing (see Figure 4.10). Out of the 271 frames available for each camera, we select only the five first frames and create a subset of sixty images. Once the poses are correctly estimated for this subset using SfM, other techniques could be used to estimate the pose of the remaining frames.

4.6.2 Evaluation

For the FVV, we remove LIFT from the features set as the pose estimation results are lower than the other features for all the metrics considered. We evaluate the FVV image pose estimation by emphasising on the metrics relevant for commercial applications. The pose estimation is evaluated in the number of images registered as a function of the processing time to find the fastest method available. As the pose accuracy is also essential for the final model accuracy, we also compare the relative orientation error with a set of camera poses provided by Volograms and computed using the method described in [8]. Instead of using a reference camera for computing the relative orientations, we compute the average error over all the images pairs.

Figure 4.11 shows that SIFT is the fastest algorithm for any number of images. It is followed by SIFT-Affine, AKAZE and ASFIT. SuperPoint lies in the middle in term of speed but fails to register images from two cameras with the widest angle gaps from the main group of cameras.

In term of accuracy, all the feature provide an average relative orientation error lower than 1° except for two configurations with the lowest number of keypoints. SIFT-Affine provides the lowest and most stable errors for every number of features.

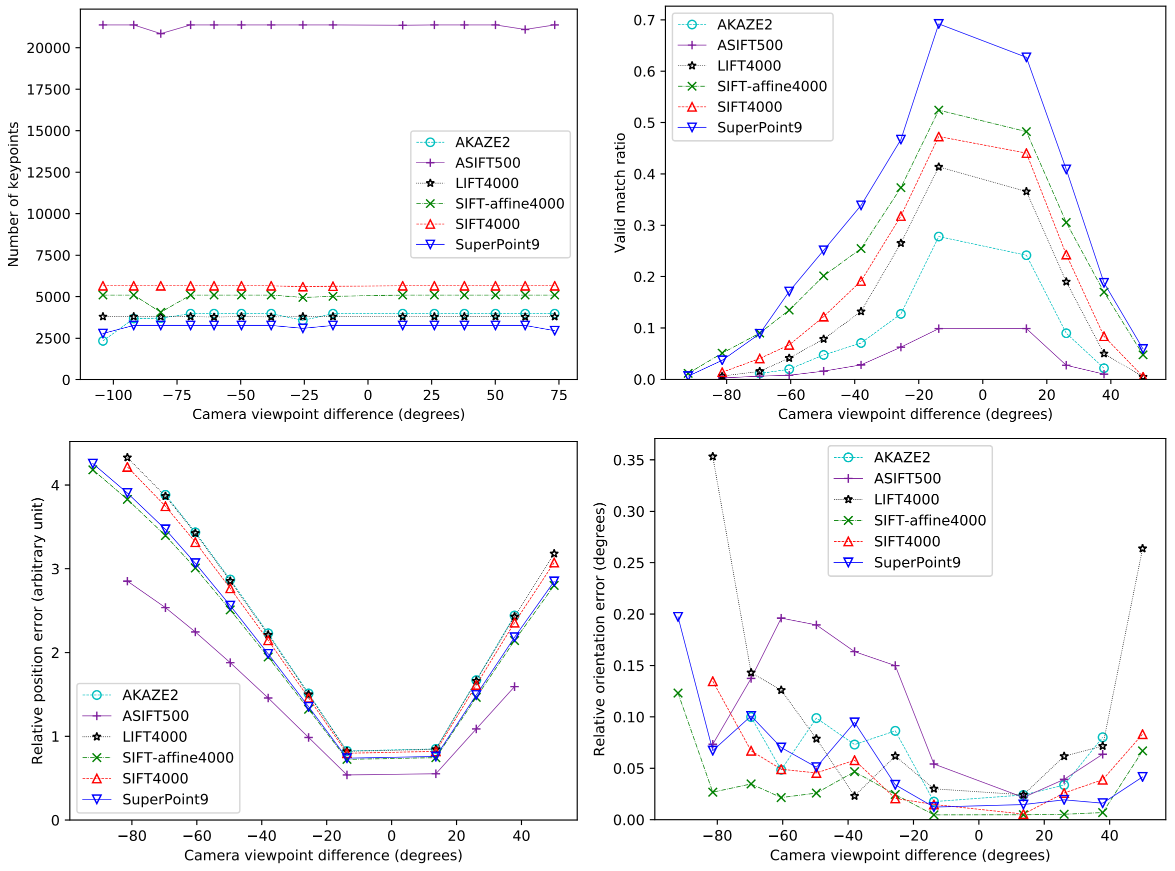
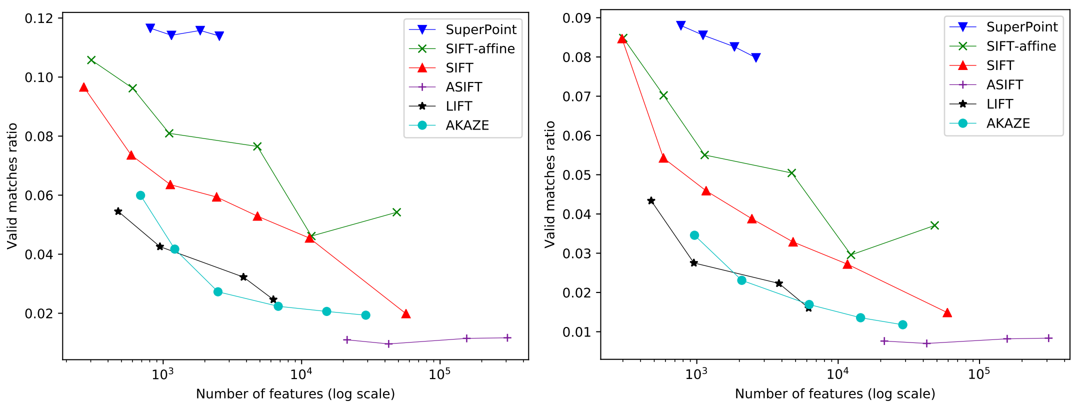
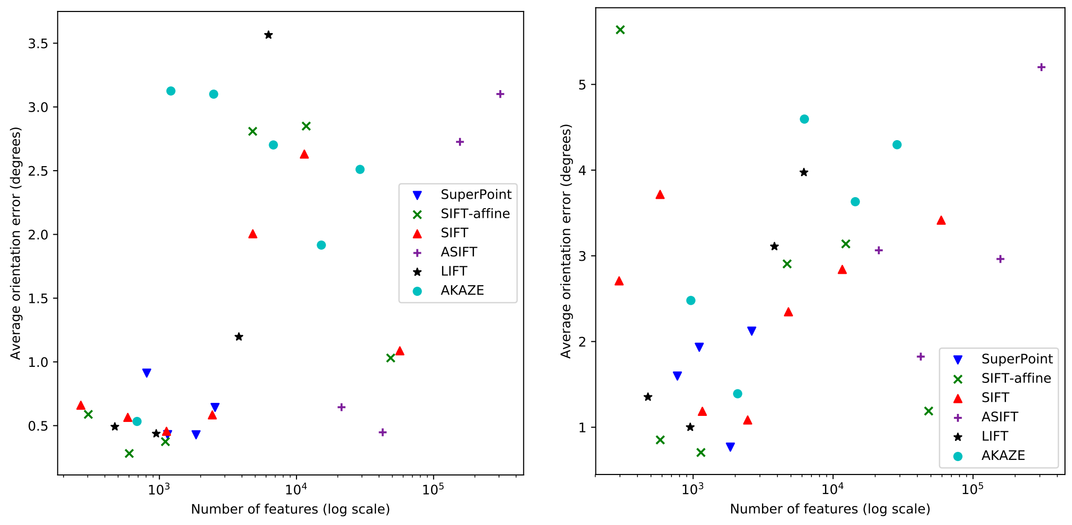


Figure 4.2: Metrics obtained by running one configuration with around 4,000 keypoints for the six features evaluated in this study on the *Wall-close* dataset with a reference camera normal to the wall. The graphs shows the number of keypoints (top left), the match ratio (top right), the relative position error (bottom left), and the relative orientation error (bottom right) as a function of the relative camera viewpoint angle.



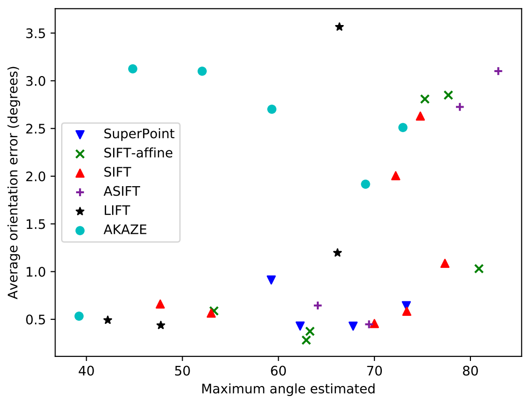
(a) Reference camera at 90° (b) Reference camera at 45°

Figure 4.3: The match ratio for all the feature configurations, averaged over all the datasets, and for two reference cameras.

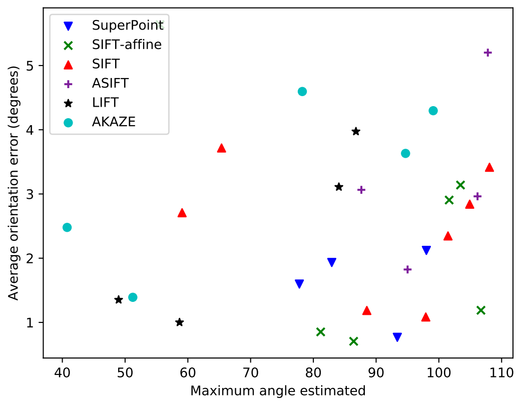


(a) Reference camera at 90° (b) Reference camera at 45°

Figure 4.4: The relative orientation error as functions of the number of keypoints for all the feature configurations averaged over all the datasets for two reference cameras.

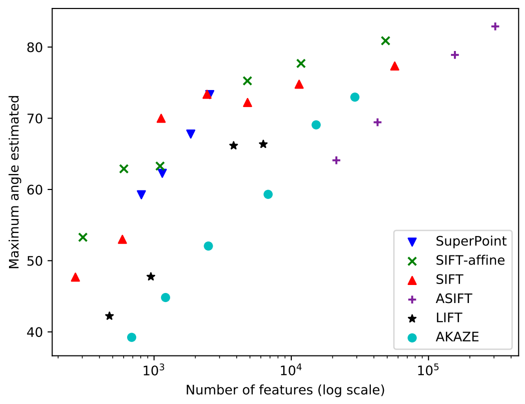


(a) Reference camera at 90°

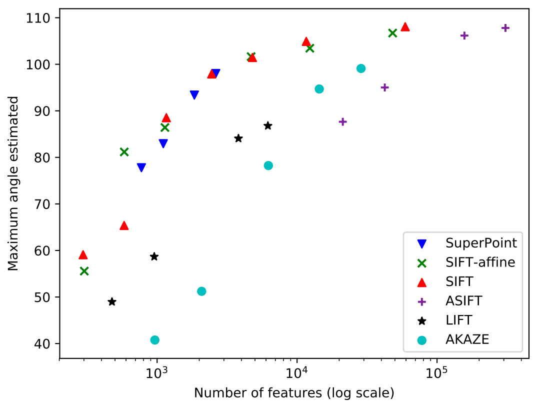


(b) Reference camera at 45°

Figure 4.5: The relative orientation error as functions of the maximum baseline angle correctly estimated on all the feature configurations, averaged over all the datasets and for two reference cameras.

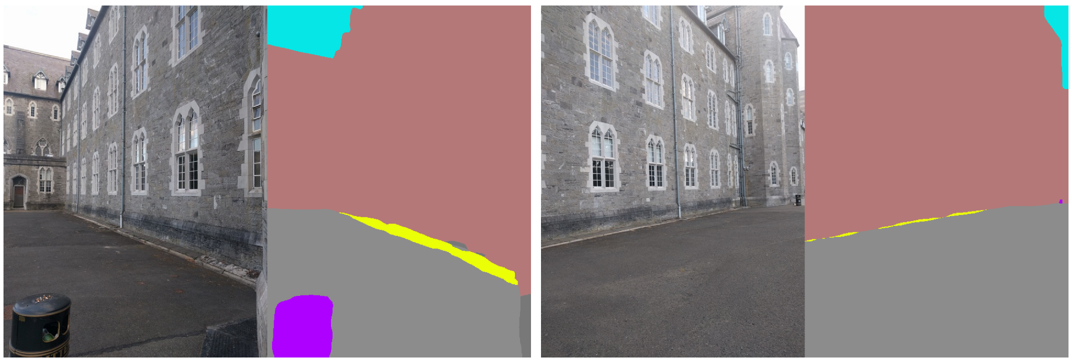


(a) Reference camera at 90°

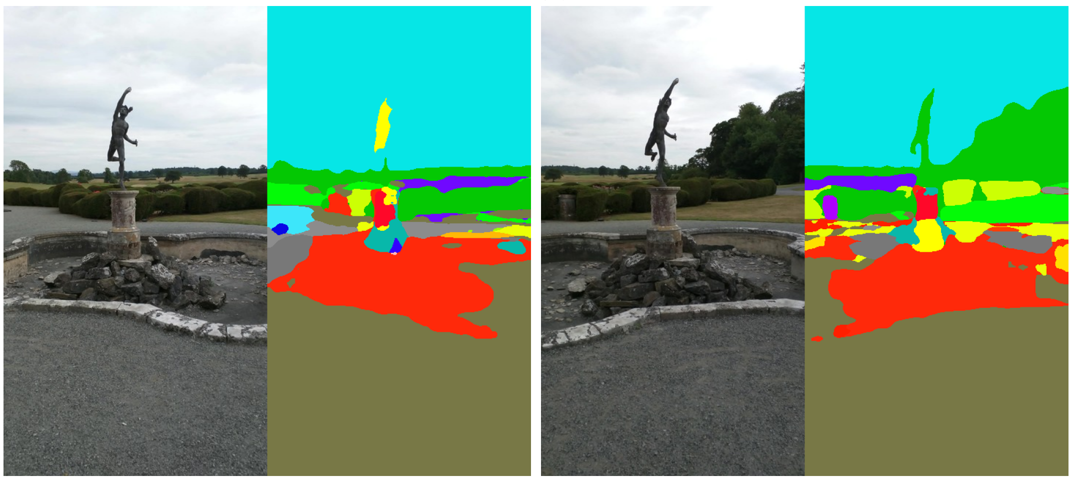


(b) Reference camera at 45°

Figure 4.6: The maximum baseline angle correctly estimated as a function of the number of keypoints on all the feature configurations, averaged over all the datasets and for two reference cameras.

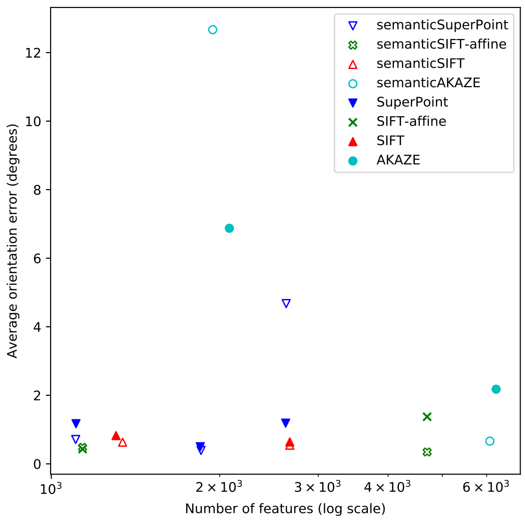


(a) *Wall*

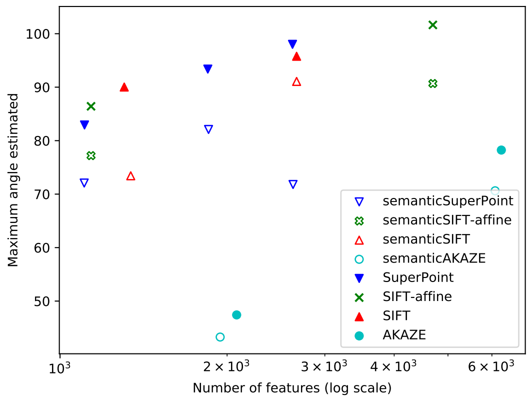


(b) *Statue*

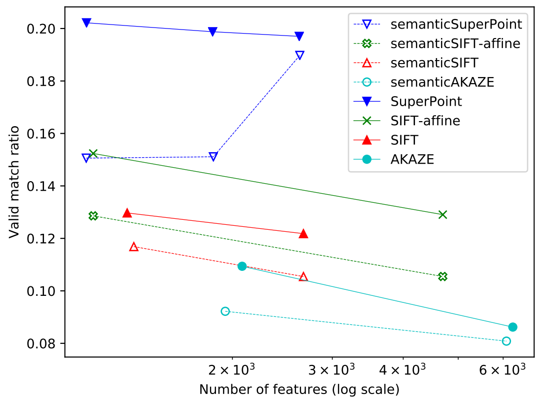
Figure 4.7: Examples of the semantic segmentation of two images from two of the four datasets. The classes are represented by colours. The *Wall* images have a stable segmentation with only two main classes whereas the *Statue* images have up to a dozen classes and present many classification errors. Here, the statue of the person is associated with two different classes.



(a) Orientation error



(b) Maximum baseline



(c) Match ratio

Figure 4.8: The effect of performing semantic classification of keypoints before the matching averaged on all the datasets. The hollow markers shows the results with the semantic classification.

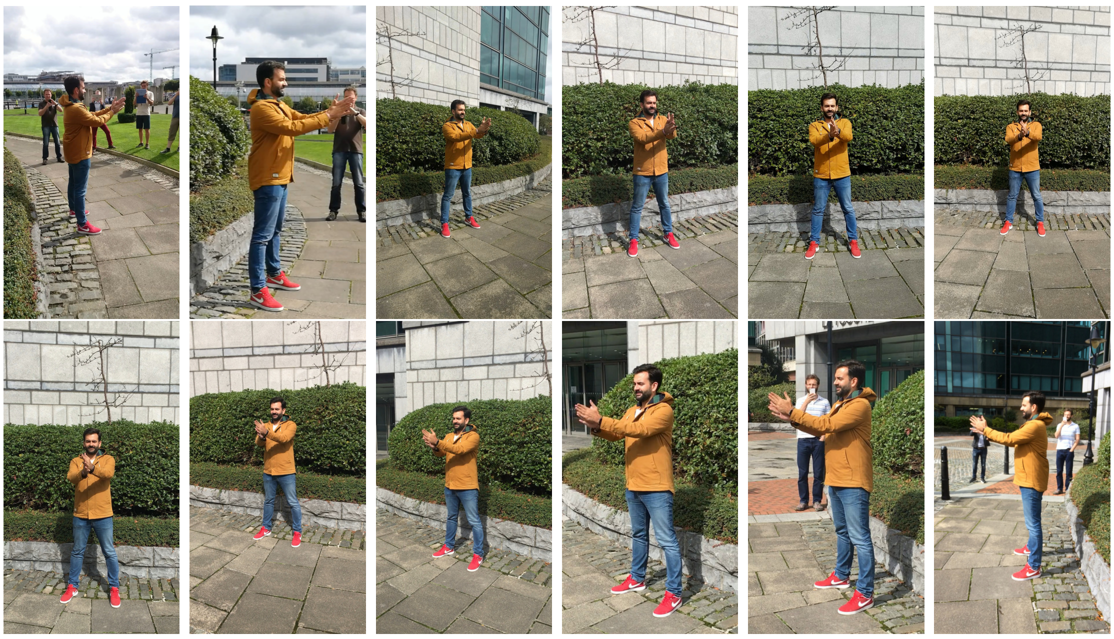


Figure 4.9: The first frames of the twelve hand-held smartphone videos from the FVV dataset *Rafa Dance* ordered from the most right-hand side of the subject to the most left-hand side. The cameras 2, 10 and 11 were taken in landscape orientation and are cropped here for easier presentation.

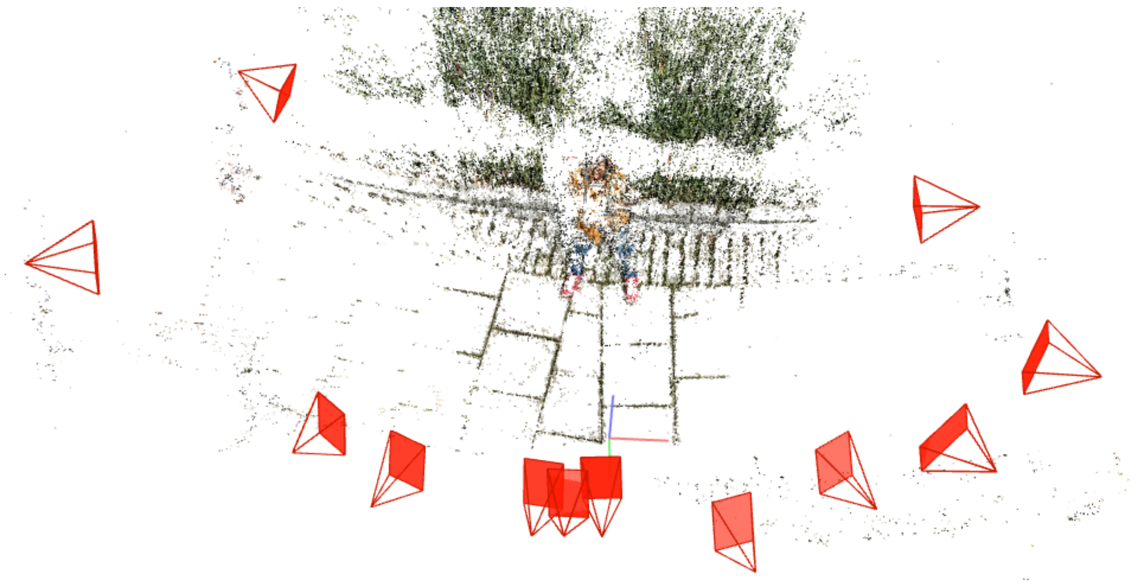
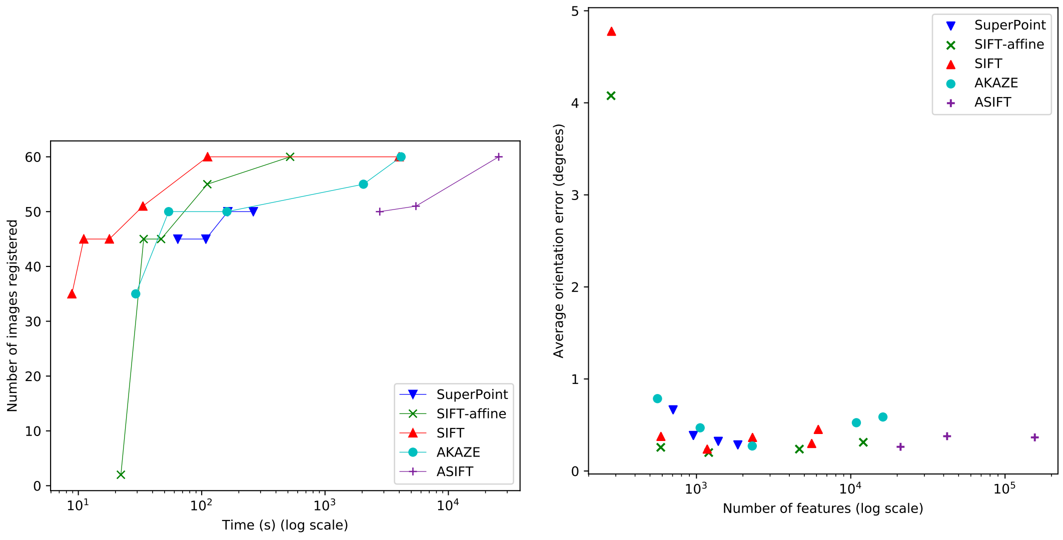


Figure 4.10: The sparse 3D reconstruction of the five first frames of the *Rafa Dance* dataset. The cameras span over more than 180° around the target. While most of the cameras form are grouped in a packed area, three of them are quite isolated.



(a) Number of images registered (b) Average orientation error

Figure 4.11: The pose estimation results on the FVV dataset *Rafa Dance*. The number of images whose pose are correctly estimated is displayed as a function of the total processing time, from feature extraction to pose estimation (a). The average orientation error compared to the reference provided by Volograms are display as a function of the number of keypoints (b).

Chapter 5

Limitations and future work

In this chapter, we present some limitations of our work and discuss the results. Finally, we detail possible future work.

5.1 Limitations

5.1.1 Control of the number of keypoints

As we have seen in the previous Chapter, the implementations of the features detectors that we have used in this study do not allow fine control of the number of keypoints extracted from images. Image sharpness, exposition and texture impact the number of keypoints extracted for each algorithm differently. While some algorithm present different algorithms to control the number of features, other need to have their thresholds tuned manually. Moreover, for the SuperPoint feature, the number of features extracted and the accuracy of their localisation is dependent on the input image resolution that was limited in this evaluation by the GPU memory.

As a result, some features might have presented a lower number of matches or reduced location accuracy in some of the configurations evaluated. This problem could be solved by improving the implementation of the feature detectors to be more adaptive to the image at the cost of increased computation time.

5.1.2 Orientation error

In the SfM algorithm implemented in COLMAP that we use for the pose estimation, a set of parameters determine whether a pose is correctly estimated or not. We used the default values for the maximum reprojection error and the minimum number of inliers to filter the images. Also, images that have no match with the reference image are eliminated.

As a result, the average error is computed on a different number of images for the different configurations tested. Defining a metric per image could have solved this issue.

5.1.3 Camera parameters

In the targeted application of FVV, the scene is acquired by different cameras for each viewpoint. Each smartphone camera has its own set of intrinsic parameters that need to be estimated during the pose estimation. As each camera is not moving much during the recording, the radial distortion might be difficult to estimate as the 3D configuration might not constraint it.

In the dataset that we use in the evaluation, all the images share the same intrinsic parameters which simplify the pose estimation and improves the accuracy.

One way of improving the datasets, without having to use tens of different smartphones, could have been to acquire higher resolution images, apply random distortions that match those of smartphones and then resize the images to match the final video resolution.

5.2 Discussion

We have evaluated multiple hand-crafted and learned features over a wide range of test datasets and real FVV smartphone videos to answer our research question. In all the scenarios, the hand-crafted algorithms, namely SIFT and SIFT-Affine, provide the best performances, both in term of pose accuracy, maximum baseline and speed. The hand-crafted features have been refined over the years to provide optimum robustness when put in production in a wide range of situations. In contrast, the learned features have been developed more recently, shows some limitations, either in the number of features or in computing time, and do not provide enough performances to allow direct use in production for FVV.

5.3 Future work

We have seen that current learned features have not yet reached the level of the best hand-crafted features. However, in the field of camera relocalisation (see Section 2.4), recent work has pushed the boundaries of the hand-crafted features and has allowed learned CNN to reach new standards in term of pose accuracy. While this work requires to train a network for each situation, it provides good insight on the possibilities of CNN for pose estimation.

Future work could use these research directions to design a new type of learned feature in the hope to overcome the current limitations. The new feature could combine ideas from the following research:

- Allow easy unsupervised training of the feature using geometric consistency [73],
- Create a hierarchical representation of features with increasing resolution to refine the matching accuracy [87],
- Take advantage of auxiliary learning of the semantic information to improve the generalisation and improve the features description [71, 72].

Chapter 6

Conclusion

In this research, we have compared the capacities for hand-crafted and learned features to estimate the relative pose of cameras from images spanning a circle around a target. We evaluated whether learned features could beat traditional hand-crafted features in term of position accuracy, maximum relative baseline angle or computing speed.

Over a wide range of specifically acquired datasets, the results showed the superiority of the hand-crafted features on a wide range of numbers of keypoints in term of accuracy, maximum baseline angle or speed of the poses estimation. Learned features have improved over the last years and reach now state-of-the-art results in some configurations. An additional analysis presented the advantages of classifying the features based on the image semantic as it helps filter the keypoints and improves the pose accuracy.

This evaluation helps identify the best feature configuration depending on the speed, accuracy or camera baseline used in an application like free-viewpoint video creation. Surprisingly, the SIFT feature, introduced in 2004, still outperforms the more recent ones in many scenarios.

The feature configurations studied in this research were limited by the computing resources and the current implementations of the algorithms which reduced the span of evaluation for some features. Besides, the evaluation metrics were averaged to simplify the presentation of the numerous data points which resulted in some bias when the poses were filtered during the bundle adjustment step.

This research shows that learned features still need some work before exceeding the traditional hand-crafted ones. Recent studies on camera relocalisation using deep learning give good examples on how to combine image semantic and self-supervised learning to create a new type of learned feature that could be easily trained to create hierarchical descriptors invariant in wide-baseline scenarios.

Bibliography

[1] J. L. Schönberger and J. M. Frahm, "Structure-from-Motion Revisited," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 4104–4113.

[2] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[3] C. Wang, Y. Zhang, and X. Zhou, "Robust Image Watermarking Algorithm Based on ASIFT against Geometric Attacks," *Applied Sciences*, vol. 8, p. 410, Mar. 2018.

[4] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua, "LIFT: Learned Invariant Feature Transform," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 467–483.

[5] D. DeTone, T. Malisiewicz, and A. Rabinovich, "SuperPoint: Self-Supervised Interest Point Detection and Description," in *CVPR Deep Learning for Visual SLAM Workshop*, 2018.

[6] C. Strecha, W. v. Hansen, L. V. Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2008, pp. 1–8.

[7] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, and S. Sullivan, "High-quality Streamable Free-viewpoint Video," *ACM Trans. Graph.*, vol. 34, no. 4, pp. 69:1–69:13, Jul. 2015.

[8] R. Pagés, K. Amplianitis, D. Monaghan, J. Ondřej, and A. Smolić, "Affordable content creation for free-viewpoint video and VR/AR applications," *Journal of Visual Communication and Image Representation*, vol. 53, pp. 192–201, May 2018.

[9] H. M. Briceño, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe, "Geometry Videos: A New Representation for 3d Animations," in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 136–146.

[10] P. K. Tsung, P. C. Lin, K. Y. Chen, T. D. Chuang, H. J. Yang, S. Y. Chien, L. F. Ding, W. Y. Chen, C. C. Cheng, T. C. Chen, and L. G. Chen, "A 216fps 4096x2160p 3dtv set-top box SoC for free-viewpoint 3dtv applications," in *2011 IEEE International Solid-State Circuits Conference*, Feb. 2011, pp. 124–126.

[11] Canon Europe, "Canon announces development of the Free Viewpoint Video System virtual camera system that creates an immersive viewing experience - Canon Press Centre," Sep. 2017. [Online]. Available: https://www.canon-europe.com/press-centre/press-releases/2017/09/free-viewpoint-video-system/

[12] M. Tanimoto, "Free-Viewpoint Television," in *Image and Geometry Processing for 3-D Cinematography*, ser. Geometry and Computing. Springer, Berlin, Heidelberg, 2010, pp. 53–76.

[13] A. Smolic, "3d video and free viewpoint video—From capture to display," *Pattern Recognition*, vol. 44, no. 9, pp. 1958 – 1968, 2011.

[14] R. Silva, "TV Makers End 3d TV - What You Need To Know," May 2018. [Online]. Available: https://www.lifewire.com/why-3d-tv-died-4126776

[15] G. A. Thomas, "Real-Time Camera Pose Estimation for Augmenting Sports Scenes," in *The 3rd European Conference on Visual Media Production (CVMP 2006) - Part of the 2nd Multimedia Conference 2006*, Nov. 2006, pp. 10–19.

[16] O. Grau, G. A. Thomas, A. Hilton, J. Kilner, and J. Starck, "A Robust Free-Viewpoint Video System for Sport Scenes," in *2007 3DTV Conference*, May 2007, pp. 1–4.

[17] J. Y. Guillemaut, A. Hilton, J. Starck, J. Kilner, and O. Grau, “A Bayesian Framework for Simultaneous Matting and 3d Reconstruction,” in *Sixth International Conference on 3-D Digital Imaging and Modeling (3DIM 2007)*, Aug. 2007, pp. 167–176.

[18] A. Zarean and S. Kasaei, “Human body 3d reconstruction in multiview soccer scenes by depth optimization,” in *2016 24th Iranian Conference on Electrical Engineering (ICEE)*, May 2016, pp. 1591–1596.

[19] Y. Ohta and H. Tamura, *Mixed Reality: Merging Real and Virtual Worlds*, 1st ed. Springer Publishing Company, Incorporated, 2014.

[20] R. McHugh, “Irish company brings Virtual Reality technologies to market Technology, news for Ireland, Exporting,Ireland,Technology„” Jun. 2018. [Online]. Available: https://www.businessworld.ie/technology-news/Irish-company-brings-Virtual-Reality-technologies-to-market--570960.html

[21] N. O’Dwyer, N. Johnson, E. Bates, R. Pagés, J. Ondrej, K. Amplianitis, D. Monaghan, and A. Smolic, “Virtual Play in Free-viewpoint Video: Reinterpreting Samuel Beckett for Virtual Reality,” in *16th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE Xplore digital library, Oct. 2017.

[22] I. Kitahara, R. Sakamoto, M. Satomi, K. Tanaka, and K. Kogure, “Cinematized reality: Cinematographic camera controlling 3d free-viewpoint video,” in *The 2nd IEE European Conference on Visual Media Production, 2005. CVMP 2005*, Nov. 2005, pp. 154–161.

[23] S. Würmlin, E. Lamboray, O. G. Staadt, and M. H. Gross, “3d Video Recorder: a System for Recording and Playing Free-Viewpoint Video†,” *Computer Graphics Forum*, vol. 22, no. 2, pp. 181–193, 2003.

[24] C. Kuster, T. Popa, C. Zach, C. Gotsman, and M. Gross, “FreeCam: A Hybrid Camera System for Interactive Free-Viewpoint Video,” in *Proceedings of Vision, Modeling, and Visualization (VMV)*, 2011.

[25] M. Pepe, L. Fregonese, and M. Scaioni, “Planning airborne photogrammetry and remote-sensing missions with modern platforms and sensors,” *European Journal of Remote Sensing*, vol. 51, no. 1, pp. 412–435, Jan. 2018.

[26] I. M.-E. Zaragoza, G. Caroti, A. Piemonte, B. Riedel, D. Tengen, and W. Niemeier, “Structure from motion (SfM) processing of UAV images and combination with terrestrial laser scanning, applied for a 3d-documentation in a hazardous situation,” *Geomatics, Natural Hazards and Risk*, vol. 8, no. 2, pp. 1492–1504, Dec. 2017.

[27] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An Accurate O(n) Solution to the PnP Problem,” *International Journal of Computer Vision*, vol. 81, no. 2, p. 155, Jul. 2008.

[28] J. L. Schönberger, M. Pollefeys, A. Geiger, and T. Sattler, “Semantic Visual Localization,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[29] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. V. Gool, “One-Shot Video Object Segmentation,” in *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[30] J. L. Schönberger, E. Zheng, J.-M. Frahm, and M. Pollefeys, “Pixelwise View Selection for Unstructured Multi-View Stereo,” in *Computer Vision – ECCV 2016*, ser. Lecture Notes in Computer Science. Springer, Cham, Oct. 2016, pp. 501–518.

[31] M. Dou, S. Khamis, Y. Degtyarev, P. Davidson, S. R. Fanello, A. Kowdle, S. O. Escolano, C. Rhemann, D. Kim, J. Taylor, P. Kohli, V. Tankovich, and S. Izadi, “Fusion4d: Real-time Performance Capture of Challenging Scenes,” *ACM Trans. Graph.*, vol. 35, no. 4, pp. 114:1–114:13, Jul. 2016.

[32] D. Casas, M. Volino, J. Collomosse, and A. Hilton, “4d video textures for interactive character appearance: 4d video textures for interactive character appearance,” *Computer Graphics Forum*, vol. 33, no. 2, pp. 371–380, May 2014.

[33] R. Pagés, D. Berjón, F. Morán, and N. García, “Seamless, Static Multi-Texturing of 3d Meshes,” *Comput. Graph. Forum*, vol. 34, no. 1, pp. 228–238, Feb. 2015.

[34] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, ISBN: 0521540518, 2004.

[35] D. Nister, "An efficient solution to the five-point relative pose problem," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2, Jun. 2003, pp. II–195–202 vol.2.

[36] M. A. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[37] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle Adjustment — A Modern Synthesis," in *Vision Algorithms: Theory and Practice*, B. Triggs, A. Zisserman, and R. Szeliski, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 298–372.

[38] C. Zach, "Robust Bundle Adjustment Revisited," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, vol. 8693, pp. 772–787.

[39] J. L. Schonberger, H. Hardmeier, T. Sattler, and M. Pollefeys, "Comparative Evaluation of Hand-Crafted and Learned Local Features," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, HI: IEEE, Jul. 2017, pp. 6959–6968.

[40] L. Zheng, Y. Yang, and Q. Tian, "SIFT Meets CNN: A Decade Survey of Instance Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1224–1244, May 2018.

[41] G. Yu and J.-M. Morel, "A fully affine invariant image comparison method," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, Apr. 2009, pp. 1597–1600.

[42] ——, "ASIFT: An Algorithm for Fully Affine Invariant Comparison," *Image Processing On Line*, vol. 1, pp. 11–38, Feb. 2011.

[43] K. Mikolajczyk and C. Schmid, “An Affine Invariant Interest Point Detector,” in *Proceedings of the 7th European Conference on Computer Vision-Part I*, ser. ECCV ’02. London, UK, UK: Springer-Verlag, 2002, pp. 128–142. [Online]. Available: http://dl.acm.org/citation.cfm?id=645315.649184

[44] ——, “Scale and affine invariant interest point detectors,” *International Journal of Computer Vision*, vol. 1, no. 60, pp. 63–86, 2004.

[45] J. Matas, O. Chum, M. Urban, and T. Pajdla, “Robust wide-baseline stereo from maximally stable extremal regions,” *Image and Vision Computing*, vol. 22, no. 10, pp. 761–767, Sep. 2004.

[46] P. Musé, F. Sur, F. Cao, Y. Gousseau, and J.-M. Morel, “An A Contrario Decision Method for Shape Element Recognition,” *International Journal of Computer Vision*, vol. 69, no. 3, pp. 295–315, Sep. 2006.

[47] C. Harris and M. Stephens, “A combined corner and edge detector,” in *In Proc. of Fourth Alvey Vision Conference*, 1988, pp. 147–151.

[48] A. Vedaldi and B. Fulkerson, “VLFeat: An Open and Portable Library of Computer Vision Algorithms,” 2008. [Online]. Available: http://www.vlfeat.org/

[49] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, “KAZE Features,” in *Eur. Conf. on Computer Vision (ECCV)*, Fiorenze, Italy, 2012.

[50] P. F. Alcantarilla, J. Nuevo, and A. Bartoli, “Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces,” in *British Machine Vision Conf. (BMVC)*, Bristol, UK, 2013.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[52] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *arXiv:1409.1556 [cs]*, Sep. 2014.

[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778.

[54] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 2980–2988.

[55] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene Parsing through ADE20k Dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[56] K. M. Yi, Y. Verdie, P. Fua, and V. Lepetit, "Learning to Assign Orientations to Feature Points," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 107–116.

[57] S. Zagoruyko and N. Komodakis, "Learning to Compare Image Patches via Convolutional Neural Networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[58] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "MatchNet: Unifying feature and metric learning for patch-based matching," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*. IEEE Computer Society, Oct. 2015, pp. 3279–3286.

[59] Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "TILDE: A Temporally Invariant Learned DEtector," in *Proceedings of the Computer Vision and Pattern Recognition*, Boston, Massachusetts, USA, 2015.

[60] H. Aanæs, A. L. Dahl, and K. S. Pedersen, "Interesting Interest Points," *International Journal of Computer Vision*, vol. 97, no. 1, pp. 18–35, Mar. 2012.

[61] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu,

and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[62] K. Wilson and N. Snavely, “Robust Global Translations with 1dsfm,” in *Computer Vision – ECCV 2014*, ser. Lecture Notes in Computer Science. Springer, Cham, Sep. 2014, pp. 61–75.

[63] A. Kendall, M. Grimes, and R. Cipolla, “PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization,” in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 2938–2946.

[64] R. Li, Q. Liu, J. Gui, D. Gu, and H. Hu, “Night-time indoor relocalization using depth image with Convolutional Neural Networks,” in *2016 22nd International Conference on Automation and Computing (ICAC)*, Sep. 2016, pp. 261–266.

[65] F. Walch, C. Hazirbas, L. Leal-Taixé, T. Sattler, S. Hilsenbeck, and D. Cremers, “Image-Based Localization Using LSTMs for Structured Feature Correlation,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 627–637.

[66] P. Purkait, C. Zhao, and C. Zach, “SPP-Net: Deep Absolute Pose Regression with Synthetic Views,” *arXiv:1712.03452 [cs]*, Dec. 2017.

[67] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, “VidLoc: A Deep Spatio-Temporal Model for 6-DoF Video-Clip Relocalization,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2652–2660.

[68] A. Kendall and R. Cipolla, “Modelling uncertainty in deep learning for camera relocalization,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 4762–4769.

[69] ——, “Geometric Loss Functions for Camera Pose Regression with Deep Learning,” in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6555–6564.

[70] E. Brachmann and C. Rother, "Learning less is more - 6d camera localization via 3d surface regression," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[71] N. R. Abhinav Valada and W. Burgard, "Deep Auxiliary Learning For Visual Localization And Odometry," in *Proceedings Of The IEEE International Conference On Robotics And Automation (ICRA)*, May 2018.

[72] N. Radwan, A. Valada, and W. Burgard, "VLocNet++: Deep Multitask Learning for Semantic Visual Localization and Odometry," *arXiv:1804.08366 [cs]*, Apr. 2018.

[73] C. Godard, O. M. Aodha, and G. J. Brostow, "Unsupervised Monocular Depth Estimation with Left-Right Consistency," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 6602–6611.

[74] P.-H. Huang, K. Matzen, J. Kopf, N. Ahuja, and J.-B. Huang, "DeepMVS: Learning Multi-View Stereopsis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[75] S. Wang, R. Clark, H. Wen, and N. Trigoni, "End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 513–542, Apr. 2018.

[76] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "CodeSLAM — Learning a Compact, Optimisable Representation for Dense Visual SLAM," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2018.

[77] Itseez, "Open Source Computer Vision Library," 2015. [Online]. Available: https://github.com/itseez/opencv

[78] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09).* INSTICC Press, 2009, pp. 331–340.

[79] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning Hierarchical Features for Scene Labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[80] P. Pinheiro and R. Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling," in *International Conference on Machine Learning*, Jan. 2014, pp. 82–90.

[81] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 3431–3440.

[82] H. Noh, S. Hong, and B. Han, "Learning Deconvolution Network for Semantic Segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 1520–1528.

[83] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3d semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4628–4635.

[84] C. Zhao, L. Sun, and R. Stolkin, "A fully end-to-end deep learning approach for real-time simultaneous 3d reconstruction and material recognition," in *2017 18th International Conference on Advanced Robotics (ICAR)*, Jul. 2017, pp. 75–82.

[85] A. Mustafa and A. Hilton, "Semantically Coherent Co-Segmentation and Reconstruction of Dynamic Scenes," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 5583–5592.

[86] A. Baumberg, "Reliable feature matching across widely separated views," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 1, 2000, pp. 774–781 vol.1.

[87] A. Resindra Widya, A. Torii, and M. Okutomi, "Structure from motion using dense CNN features with keypoint relocalization," *IPSJ Transactions on Computer Vision and Applications*, vol. 10, May 2018.