Monocular Visual Odometry: Deep Learning vs Classical Approaches

by

Venkatesh Vaidyanathan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2022 by the
Graduate Supervisory Committee:

Hemanth Venkateswara, Co-Chair
Troy McDaniel, Co-Chair
Katina Michael

ARIZONA STATE UNIVERSITY

August 2022

ABSTRACT

Visual Odometry is one of the key aspects of robotic localization and mapping. Visual Odometry consists of many geometric-based approaches that convert visual data (images) into pose estimates of where the robot is in space. The classical geometric methods have shown promising results; they are carefully crafted and built explicitly for these tasks. However, such geometric methods require extreme fine-tuning and extensive prior knowledge to set up these systems for different scenarios. Classical Geometric approaches also require significant post-processing and optimization to minimize the error between the estimated pose and the global truth.

In this body of work, the deep learning model was formed by combining SuperPoint and SuperGlue. The resulting model does not require any prior fine-tuning. It has been trained to enable both outdoor and indoor settings. The proposed deep learning model is applied to the Karlsruhe Institute of Technology and Toyota Technological Institute dataset along with other classical geometric visual odometry models. The proposed deep learning model has not been trained on the Karlsruhe Institute of Technology and Toyota Technological Institute dataset. It is only during experimentation that the deep learning model is first introduced to the Karlsruhe Institute of Technology and Toyota Technological Institute dataset. Using the monocular grayscale images from the visual odometer files of the Karlsruhe Institute of Technology and Toyota Technological Institute dataset, through the experiment to test the viability of the models for different sequences. The experiment has been performed on eight different sequences and has obtained the Absolute Trajectory Error and the time taken for each sequence to finish the computation. From the obtained results, there are inferences drawn from the classical and deep learning approaches.

DEDICATION

I would like to dedicate this to my parents and grandparents.

ACKNOWLEDGMENTS

I would first like to extend my utmost gratitude to Dr. Hemanth Venkateswara, who has been a pillar of support in the journey of publishing this body of work. He has been very supportive in helping me develop my technical understanding and has provided immense personal support. I also want to thank Dr. Troy McDaniel for allowing me to work in the CUbiC lab. And he has always been a kind and endearing personality. And all this would not have been possible without Dr. Katina Michael who was the person responsible for introducing me to the CUbiC lab. My industrial mentor from Intel, Dr. Rita Chattopadhyay, has also provided invaluable technical support and feedback on my work. I would also like to thank Intel Corporation for helping me carry out my work by providing me the necessary financial support.

In my personal sphere, I would like to thank my mother and father for providing me with tremendous emotional and financial support. I would also like to thank Angie Lizarraga and my two pets, Mischief and Maya, for their invaluable emotional support throughout my masters. Finally, I would like to thank god for helping me through my journey in obtaining my master's degree.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

Visual Odometry can be defined as incrementally estimating a robot's pose by examining the changes that motion induces on the images captured by the onboard cameras[5]. The prominence of Visual Odometry began to rise when compared to Wheel Odometry; it was shown to be unaffected by wheel slip in uneven terrain and adverse conditions. With respect to trajectory estimates, Visual Odometry only produces a relative error of 0.1 percent to 2 percent compared to Wheel Odometry. Apart from its robust standalone qualities, it also compliments other pose estimation methods. These methods include Wheel Odometry, Global Position Systems(GPS), Inertial Measurement Units(IMU), and Laser Odometry. Visual Odometry truly shines in robotic localization and mapping because it can solely be used in GPS denied environments such as aerial and underwater mapping.

One of the main distinctions that need to be drawn is the distinctness between Visual Odometry and Visual SLAM. In fact, Visual Odometry serves as what is known as the front end of Visual SLAM. The primary goal of Visual Odometry is to ensure that the change in camera poses are recorded and extracted to draw the current position of the robot in time, or it can be stated that Visual Odometry is concerned more with local consistency. Visual SLAM, on the other hand, is a larger subset that contains Visual Odometry. It is imperative to not only maintain the local consistency but also to optimize the local consistency. Apart from wanting an optimized local consistency of the robot, Visual SLAM also addresses the problem of loop closure. Loop closure is a process in which the robot is able to identify that the robot has already passed through a point in the world. This point is not treated as a new point

but is able to identify, and the loop is detected and updated on the map. Thus it can be stated that Visual SLAM is more concerned with an optimized local consistency accompanied by the ability to perform loop closure.

In this body of work, the aim is to analyze two significant Visual Odometry methods. The first one is the classical geometric approach, and the second one is the deep learning approach. Over the last three decades, classical-geometric has seen a great deal of progress. Nevertheless, when dealing with the robustness of the world, they still suffer in areas such as settings in which there is a change of illumination and places where there is little distinction in the texture, to name but a few. Apart from these challenges, the classical geometric approaches require a great deal of personalization on their low-level features, making it difficult to adapt to robust real-world applications. Also, a great deal of prior knowledge is required to fine-tune these systems for real-world applications. One of the fundamental assumptions in which a wrench can be thrown in the classical geometric approach is that it assumes the world to be predominantly static.

These are the issues that deep learning Visual Odometer hopes to address. The deep learning approach to Visual Odometry is based on learning from high-level appearance present in the environment. Deep learning models can be trained on large, diverse, and robust datasets, and these models can learn and make inferences with significantly less low-level design.

1.1 Problem Formulation

The proecess of defining the Visual Odometry problem by beginning with an agent which moves through an environment capturing images from a rigidly attached camera at discrete times k [29]. When there is the use of a monocular system the

sequence of images can be represented as,

$$I_{0:n} = (I_0, ..., I_n). \tag{1.1}$$

In the above equation, the term $I_{0:n}$ encapsulate all the images present in a particular sequence of images. The camera coordinates through the temporal instance of k can be related with the previous instance $k-1$ using the rigid body transformation formula of the following form:

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}. \tag{1.2}$$

In the above equation, the term $T_{k,k-1}$ is known as the essential matrix, the term k and k-1 denote the current image and previous image from which this relation is obtained. R is termed as the rotation matrix of the rigid body camera and t is defined as the transnational vector. Another important term is the set of camera poses which can be defined as the set of camera poses given by $C_{0:n} = C_0, ..., C_n$. $C_{9:n}$ is the position of the camera beginning from the global reference to the latest image present in the sequence. The global reference is nothing but the camera's coordinates for the first image present in the sequence. All subsequent camera positions are calculated in reference to the translational and rotational changes of the camera with respect to the global reference.

These camera poses contain the set of transformation of the camera with respect to the starting coordinate frame of the camera in the beginning of the sequence. If the agent is in motion, then the current pose at a temporal instance k of the camera can be calculated from concatenating all the transformation the camera has undergone until the previous temporal instance of k-1, and can be expressed in the following manner:

$$C_k = C_{k-1} * T_n \tag{1.3}$$

In the above equation, C_k is defined as the current camera coordinate. It is calculated as a result of the product of the coordinate of the camera from the previous image frame denoted by C_{k-1} and the essential matrix denoted by T_n. The primary goal of Visual Odometry is that we are able to discern the relative transformation of the images present in a temporal manner and that we are able to concatenate all the trajectories of the camera to estimate the full trajectory that the camera has taken from k=0 to n.

1.2 Assumptions

A few assumptions that Visual Odometry makes which need to be elaborated. These are:

- There is sufficient illumination in the environment. This to ensure that distinct features can be descended from one sequence to another.
- There is a dominance of static scenes over moving objects.
- The texture of the images present in the sequence are well defined. This is to ensure that the apparent motion between sequences is extracted.
- There is a sufficient overlap between consecutive sequences.

1.3 Organization of the Thesis

The first chapter gives an introduction to Visual Odometry, the process by which the problem is formulated and the assumptions that Visual Odometry makes. The second chapter goes into the classical Visual Odometry primarily focusing on the feature extractors and the feature matchers. The feature extraction methods we will be looking into will be ORB and SIFT, and the feature matching methods we

will be looking into will be will the Brute Force Matcher and the Fast Library for Approximate Nearest Neighbors.

The third chapter delves into a novel deep learning model released by the company Magic Leap which is the combination of the feature extractor SuperPoint and the feature matched SuperGlue. In the fourth chapter we will look into the Karlsruhe Institute of Technology and Toyota Technological Institute dataset in detail, and explain the experimental setup in which we prove using monocular grey scale images that the deep learning model is able to perfume better than the classical geometric methods. In the fifth chapter make inferences about the results obtained and also chalk up the future work that is possible.

1.4 Contributions

The following contributions have been made in this body of work:

- An inferential analysis has been performed on the SuperPoint [10] model to obtain the throughput and the latency.
- Extensive analysis is performed on the Optimal Transport problem and the log-based Sinkhorn algorithm to showcase why the deep learning model takes significantly more compute time compared to the classical geometric models.

Chapter 2

CLASSICAL METHODS FOR VISUAL ODOMETRY

The classical geometric approach is a structured pipeline that begins with the input data. In this body of work, this will be the monocular sequences of grayscale images. The input images are first fed to the feature detector when the features of interest are detected and extracted. These points are then passed on to the feature matchers, which give us the number of matching features between two images. Finally, we feed the change in features to a pose estimation method that gives us the robot's position in space.

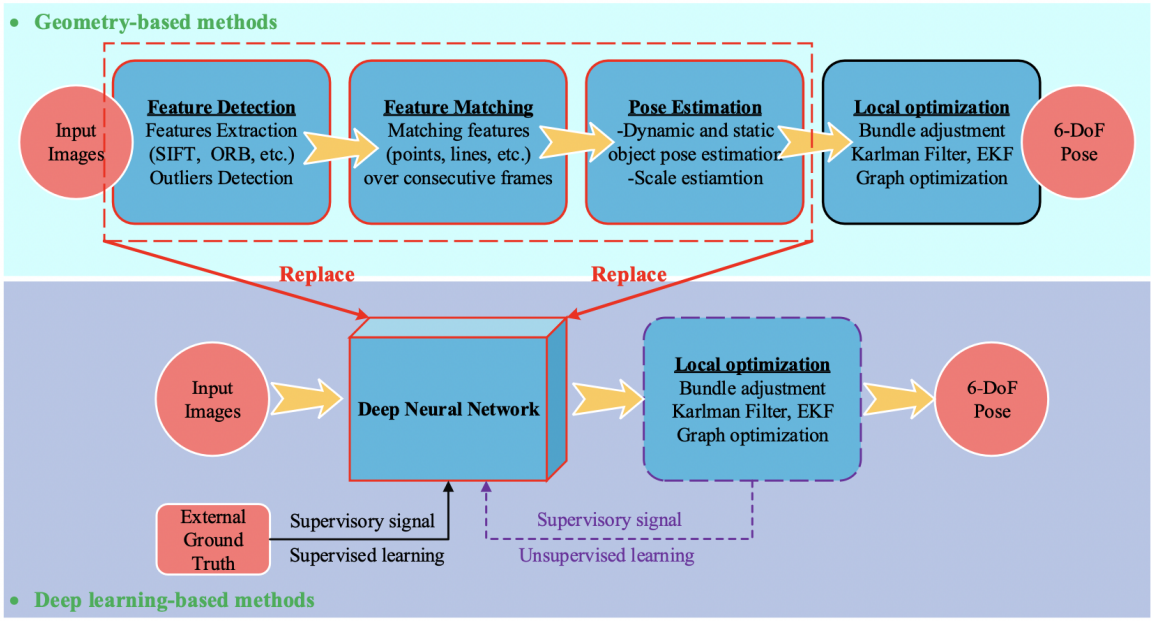

Figure 2.1: The above pipeline gives the classical geometric approach. The latter pipeline gives us the formulation of the deep learning approach to visual odometry.The Feature Detection, Feature Matching and Feature Matching are all performed by the deep learning model. Figure Source [33].

.

It begins with defining a feature; a feature is a token of data that can be used to

solve a specific computational task. Features in images are primarily two types known as key point features and edges. Key point features are localized points of interest or a group of pixels surrounding a point of primary interest. Edges may be defined as images' features that can be matched based on their orientation and edge profile. Edges enable us to distinguish the distinctness of boundaries in objects present in an image.

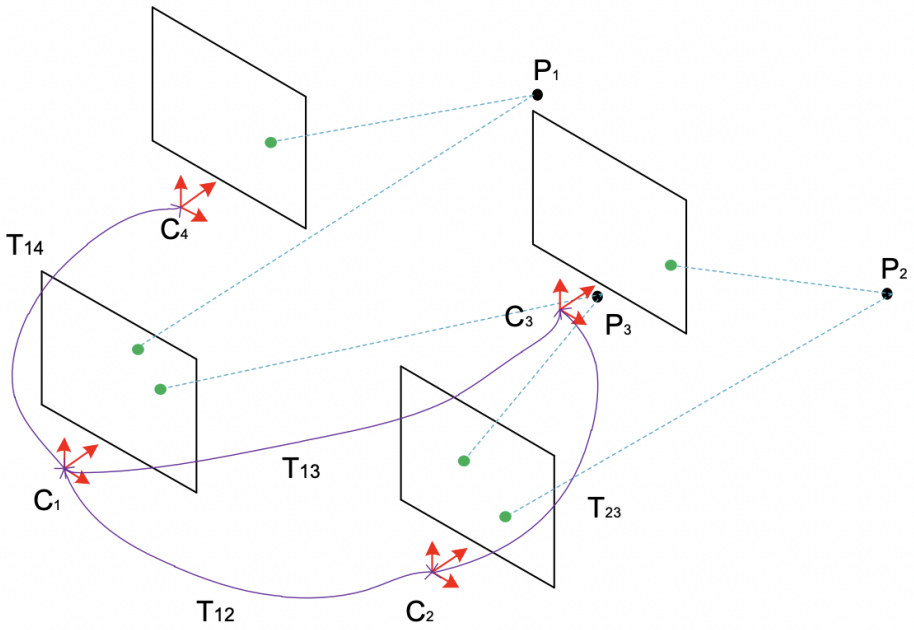

Figure 2.2: Figure showcases the monocular visual odometry process. Images are concatenated to obtain the essential matrix, which provides rotational and transnational information for the agent. Figure source [34].

Feature detection is the ability of an algorithm to extract key points of interest. A key point of interest is a point in an image that possesses a unique texture. For a point in an image to be a key point of interest, it must be well localized in the image. The key point of interest must also be stable under local and global perturbations. This means that even if there is a change in illumination in the image domain, the key point of interest must still possess the ability to be reliably computed with a high degree of repeatability. Another significant output of feature detection is the feature

descriptors. Feature descriptors help store unique and interesting information about an image in the form of a vector. Feature descriptors are essential because even if the image were to undergo a transformation, the information contained within the feature is invariant to transformations. Therefore we can find a specific feature in an image even after an image has been transformed. This body of work analyzes two classical geometric feature matchers, SIFT(Scale Invariant Feature Transform) and ORB(Orient Fast Rotate Brief).

Feature matching is the ability of an algorithm to establish a relationship between two images that either contains the same object or scene. The process involves matching the key point of interest between images in conjunction with information from the respective image descriptors. The feature matchers we have chosen for our experimental analysis are the Brute Force Matcher and FLANN (Fast Library for Approximate Nearest Neighbors).

2.1 Scale Invariant Feature Transform

The Scale Invariant Feature Transform or SIFT [18] is a proprietary algorithm invented by David Lowe in 1999. The SIFT algorithm is a feature matching algorithm that enables the detection of, describe and match local features in an image. The features extracted by SIFT are not affected by any changes made by scaling and rotating an image. It is also partially able to handle changes in illumination. The features extracted do not contain much noise and occlusion and are well distributed in the image domain. The features extracted by SIFT fall under four levels of computation.

2.1.1 Scale Space Extrema Detection

Scale Space Extreme Detection involves the identification of well-defined key points.The process of scale-space detection begins with the definition of the scale-

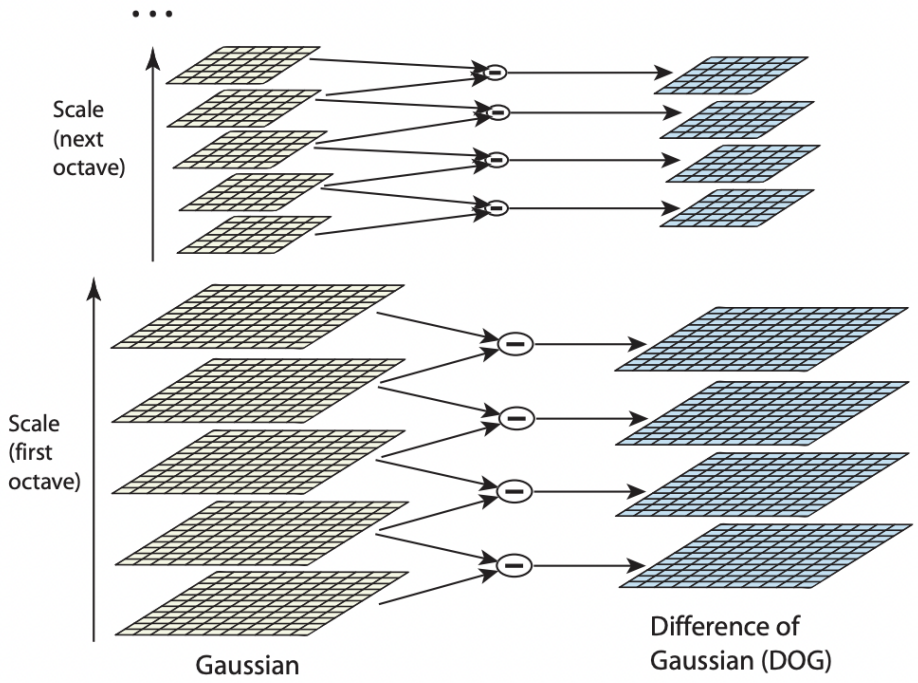

Figure 2.3: The scale space images which are present on the left are obtained by convoluting images using Gaussian process. The Gaussian space which are next to each are taken and their difference is how we obtain the Difference of Gaussian(DoG). Figure Source [19].

space of an image denoted by $L(x, y, \sigma)$.The scale space is obtained by convoluting the input image with a variable-scale Gaussian by the following equation:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y). \tag{2.1}$$

The variable Gaussian can be further be defined as:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2/2\sigma^2)}. \tag{2.2}$$

After this, the image is convoluted or warped using Gaussian filters, and the difference of successive Gaussian-image blurs is calculated. This process is known as the Difference of Gaussians(DoG). The following formula defines the DoG of an image:

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma). \tag{2.3}$$

The Gaussian blurred is applied to an image at different scales. After the DoG images are obtained, the key points are gathered by taking the maxima and minima of the DoG images across different scales. A candidate set of key points are finally extracted at the end of this step. A candidate key point has defined a keypoint whose pixel value is either equal to the minimum or the maximum of all the pixels present in that image.

2.1.2 Keypoint Localization

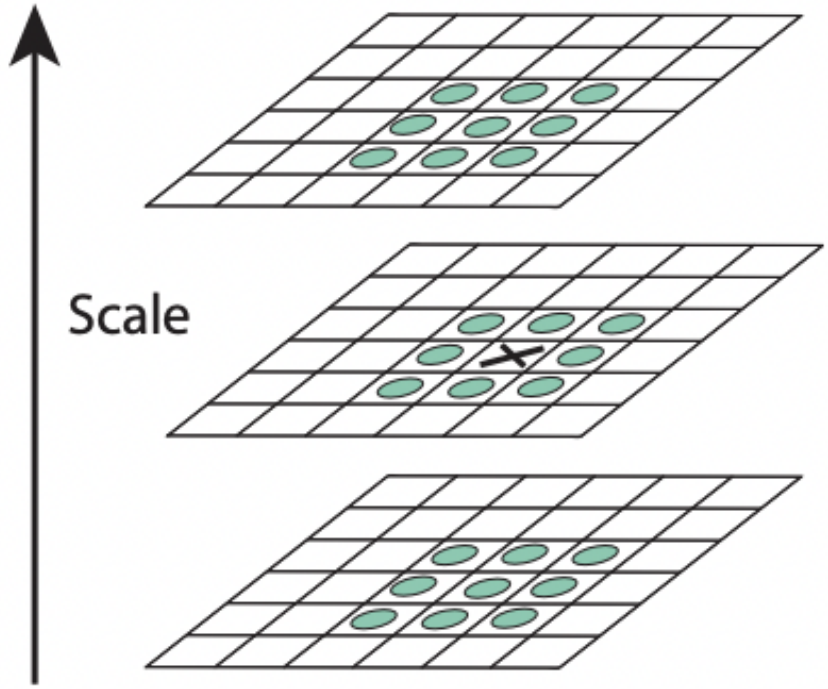

Figure 2.4: The value which pertain to the maxima and minima of the difference of gaussian's is obtained by taking a 3x3 region in which 26 neighbors are compared around the selected key point.Figure Source [19].

In the previous step, we were able to find the candidate key points in a group of pixels. In this step, we would like to precisely fit the nearby data to obtain the scale,

ratio of principal curvatures, and location. This process helps us reject key points which are poorly localized or are sensitive to noise.

Interpolation is done upon each candidate keypoint by using the nearby data to accurately localize the keypoint. The interpolation is performed by using a quadratic Taylor expression of the DoG, keeping the origin the same as the candidate keypoint. The Taylor expansion is as follows:

$$D(x) = D + \frac{\partial D^T}{\partial X} + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2} X. \tag{2.4}$$

Key points with low contrast are discarded, this is done if the value of the second-order Taylor's series expansion of is computed at the extremum. If the contrast value falls below the threshold of 0.03, the candidate key point is no longer considered.

Although rejection of low contrast does improve stability, further improvements can be made to improve the stability. The Difference of Gaussian tends to pick up on edges that are not well defined, leading to an introduction of a small amount of noise. This noise can be explained due to the nature of the Difference of Gaussian to have a high degree of affinity for edges. This leads the Difference of Gaussian to have a degree of curvature along a poorly defined edge. This large principal curvature for the poorly defined edge can be defined with the help of a Hessian Matrix as follows:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}. \tag{2.5}$$

In the above equation D represents the principal curvature of an undefined edge. In the above equation, the eigenvalues of the matrix are proportional to the value of D. To determine the trace and determinant of the matrix; we choose an arbitrary value that is assumed to be the largest and smallest value in the matrix. The largest value is assigned a variable L and the smallest value S; it is also arbitrarily taken

that D_{yy} is the largest value and D_{xx} is the smallest value. Then the trace and the determinant are calculated as follows:

$$det(H) = D_{xx} * D_{yy} - D_{xy}^2 = L * S, \tag{2.6}$$

$$Tr(H) = D_{xx} + D_{yy} = L + S. \tag{2.7}$$

To ensure efficient computation we ensure that the ratio of principal curvature is less than a defined threshold. If the threshold were to be defined by t. Then the condition to ensure that efficient computation occurs is as follows:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}. \tag{2.8}$$

2.1.3 Orientation Assignment

In this step of the algorithm, the key points that remain are given an orientation depending on local image gradient directions. This is where the key point descriptor can be given in relation to its relative orientation; by doing so, the feature descriptor is mutually exclusive to any rotation of the image. The computation begins in a scale-invariant manner by taking the Gaussian smoothed image defined as L(x,y,σ). The magnituxde of the gradient and the orientation are found using the following formulas:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \tag{2.9}$$

$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1)/L(x+1,y) - L(x-1,y)). \tag{2.10}$$

The above operations of magnitude and direction of the gradient are performed on every pixel, which is considered as a neighbor to the key point in consideration. These calculations lead to constructing an orientation histogram consisting of 36 bins, where each bin denotes a 10-degree angle. The more dominant an orientation, the higher the corresponding peak of that angle bin is.

Each peak in the histogram is symbolic of one specific direction the local gradient can take. Once the selection of the highest peak is made, all other peaks which are in the range of 80 percent of the highest are given the same orientation as the orientation of the highest peak. If there is more than one peak with the same magnitude as the highest peak, then more than one key point is generated with different orientations. About 3/20th of the points are provided with more than one orientation. This is to ensure stability during the feature matching process. The final step in the orientation assignment is to fit a parabola. The parabola is fitted onto the top three highest values of the histogram to ensure the peak position is assessed with the best possible accuracy.

2.1.4 Keypoint Descriptor

In the final step of the algorithm, it is ensured that each key point is computed so that it maintains a high degree of distinctiveness and, to a certain degree, is mutually exclusive to changes in illumination. This procedure is carried out to the image nearest in scale to the selected key point.

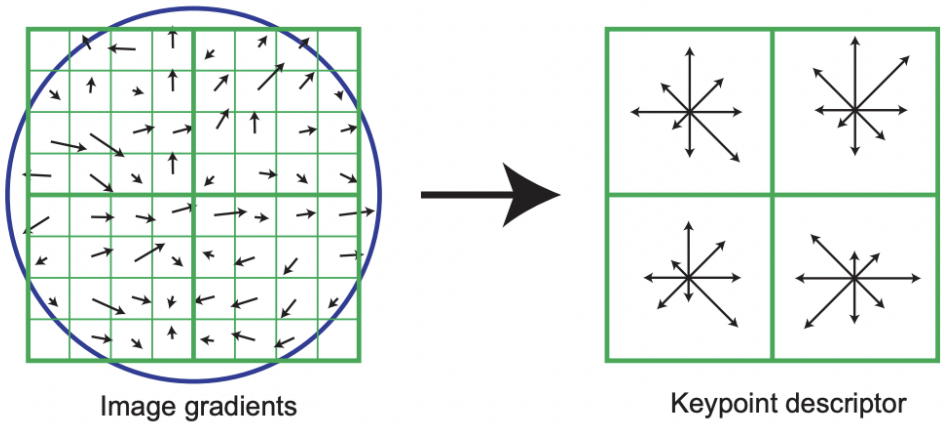

Figure 2.5: Formation of a keypoint descriptor in the SIFT algorithm. Figure Source [19].

The operations are performed on the 4x4 neighborhood of the key point. These neighboring points can be further classified into eight bins given their respective orientation histograms. The histogram is generated by obtaining the orientation samples of the neighboring 16x16 pixels of the key point. The magnitudes of the histogram are further weighted by a Gaussian function with σ equal to one-half the width of the descriptor window. The combined values of these histograms are obtained, and a final descriptor vector is obtained.

The final step in generating the keypoint descriptor is to reduce any effects which might be brought about by changes in illumination. The process begins with the normalization of the feature vector to a unit length. The normalization of the feature vector helps in preventing changes that may occur when the contrast of an image changes. The contrast of an image changes when the value of a group of pixels is changed when the group of pixels is multiplied by a constant. It also helps the feature change when brightness changes occur. Brightness change occurs when a group of pixels has a constant added to them.There is another kind of illumination change known as non-linear illumination change; this occurs during camera saturation n or when there is a change to the 3-D surface. To combat the effects of non-linear illumination change, the feature vector is normalized to no more than 20 percent the length of the normalized unit length.

2.2 Oriented Fast and Rotated Brief

Oriented FAST and Rotated BRIEF or popularly known as ORB[27], was created by Ethan et al.; at OpenCV labs. ORB was conceived as an open-source alternative, as SIFT is a patented algorithm. The ORB algorithm can be segmented into three different methods: feature extraction, generating feature point descriptors, and feature point matching.

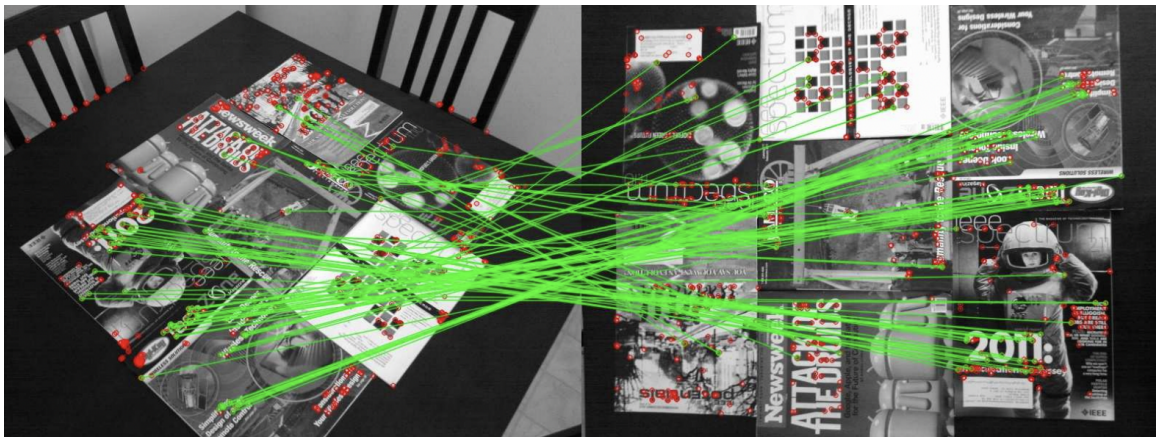

Figure 2.6: Visualization of ORB matching process when there is a change in orientation to the image.The green lines represent correct matches and the red spots are keypoints extracted which are matched. Figure Source [20].

2.2.1 Keypoint Extraction

The ORB algorithm employs the FAST algorithm or features from accelerated segment test algorithm to detect key points. The fundamental idea behind this algorithm is that if a selected pixel showcases different properties from its neighboring pixels, then that selected pixel may be considered a corner point. The process of

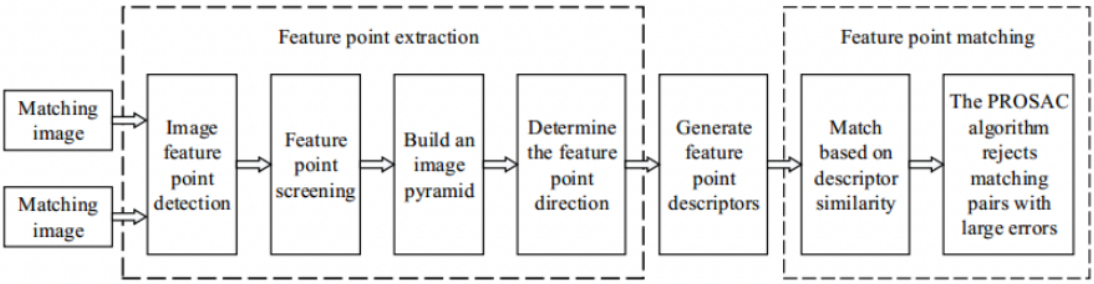

Figure 2.7: ORB pipeline. Figure Source [20].

detecting a key point begins by selecting a single pixel on an image. Let us name the selected pixel as m, the brightness of the image may be defined by I_m and the brightness threshold of the image is defined by a value T. We then consider the selected pixel m as the center and go on to the select the neighboring 16 pixels to the center m.The criteria for selecting these pixels is that we consider the pixel m

as the center and draw a circle around it encapsulating the neighboring 16 pixels. Then these surrounding 16 pixels would be encapsulated within this circle. After this process we compare the gray-scale value of the center pixel m and the neighboring pixels present within the circle. If the brightness of the selected 16 pixels is greater than the value of $I_m + T$ or less than the value of $I_m - T$, the selected pixel m is considered as a key point.

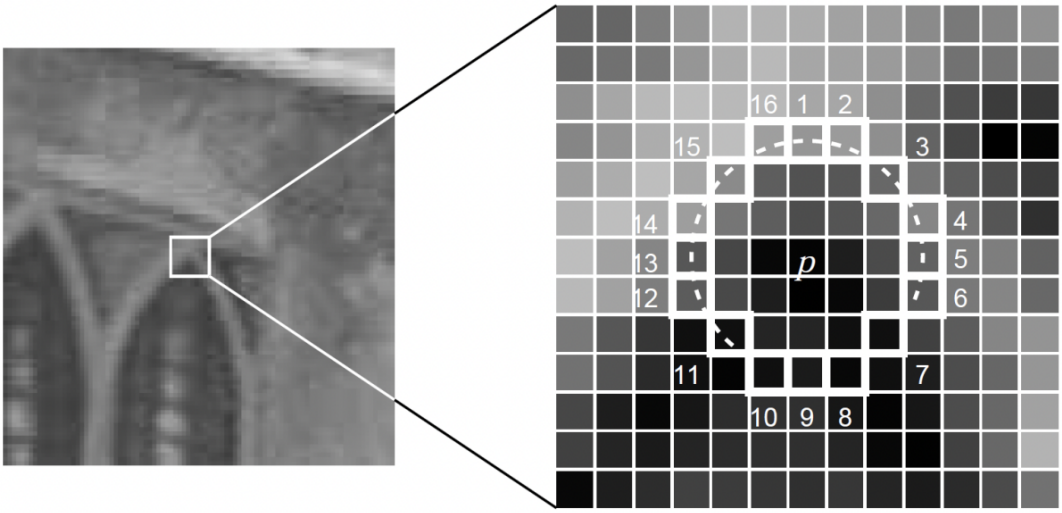

Figure 2.8: Corner detection performed using the FAST algorithm on the key point p.The pixels around the point p have a brightness greater than or less than by a certain threshold.Figure Source [26].

In the previous step the FAST[32] algorithm provides a large number of key points and there is no directional information which provided by the FAST algorithm. Therefore an improvement is made to the FAST algorithm by calculating the Harris response values which sorts the key point according to their gray-scale values and select only the first-N sorted points. The Harris response value is calculated using the following formula:

$$R = det(M) - k(trace(M))^2, \tag{2.11}$$

$$M = \Sigma w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}. \tag{2.12}$$

In the above equations R is the Harris response values, $w(x, y)$ is the image window function and M is a 2x2 matrix. The variable I_x gives us the horizontal component of the key point and I_y gives us the vertical direction of the key point.

The next step is to construct image scale pyramids and are sampled on each layer to extract FAST feature key points and ensure an addition of scale invariant to these key points. After the identification of the primary key point we would like to add the component of direction to the key points. The direction component is introduced to the key point with the help of the Intensity Centroid Method. The process begins with the selection of a small block in an image.

Let us name this image block as X. The moment the image block is defined using the following equation:

$$m_{pq} = \sum_{(x,y)\in B} x^p y^q I(x, y); \quad p, q = 0, 1. \tag{2.13}$$

In the above equation x and y are pixel coordinates in the image $I(x, y)$. $I(x, y)$ is nothing but the gray scale value corresponding to the selected pixel. The centroid of the image block X is chosen as,

$$C = \begin{pmatrix} \frac{m_{10}}{m_{00}} & \frac{m_{01}}{m_{00}} \end{pmatrix}. \tag{2.14}$$

The term m_{00} is the first instance of the image block and the first moment of the centroid of the image block is given by (m_{10}, m_{01}). In the final step the geometric center of the block O is connected to the centroid X and we obtain a vector $\vec{OX}$, and the direction of the key point is defined by:

$$\theta = arctan \left(\frac{m_{01}}{m_{10}} \right). \tag{2.15}$$

By obtaining the values of C and θ the keypoints extracted by ORB have become scale invariant and rotational invariant in nature.

2.2.2 Generating Keypoint Descriptors

Once the keypoints are obtained from the FAST algorithm, we employ an improved version of BRIEF[21] to assign a descriptor to each keypoint. The BRIEF algorithm is a binary vector descriptor, and is of the following form:

$$\tau(p; x, y) = \begin{cases} 1 & p(x) < p(y) \\ 0 & \text{otherwise} \end{cases}. \tag{2.16}$$

In the above equation $p(x)$ signifies the gray scale value along the x-axis of the keypoint and $p(y)$ is the gray scale value along the y-axis of the keypoint. To minimize the effect of any noise, the image is run through a Gaussian filter. Now if we were to take the chosen keypoint as the center m, and we take a neighborhood window of pixels of size MxM. From this neighborhood window we select a random number of pixels N(the default value of N is usually 256). After the selection of the N keypoints are made we check the brightness value according to the following equation and perform the binary assignment:

$$\theta = arctan\left(\frac{m_{01}}{m_{10}}\right). \tag{2.17}$$

In the end the algorithm by obtaining an N-dimensional vector consisting of N-binary strings and is as follows:

$$f_N(p) = \Sigma_{1 \le i \le N} 2^{i-1} \tau(p; x_i y_i). \tag{2.18}$$

One of the crucial issues with the above problem is that the above output is not rotation invariant. This will lead to loss in information if the rotated. We now employ an improved version known as Steer BRIEF to obtain a directional dimension to each key point. This is done by introducing a rotation matrix R_θ:

$$R_\theta = \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix}. \tag{2.19}$$

Matrix R_θ for each set of keypoints gives us a new matrix Q:

$$Q = \begin{bmatrix} x_1, x_2, ..., x_N \\ y_1, y_2, ..., y_N \end{bmatrix}. \tag{2.20}$$

The final directional descriptor is obtained in the following form:

$$g_{N(p,\theta)} = f_N(p)|(x_i, y_i) \in Q_\theta. \tag{2.21}$$

2.3 Feature Matching

The two matchers considered for ORB and SIFT are the Brute-Force matcher[15] and Fast Library Approximation for Nearest Neighbors[23]. The Brute-Force Matcher pairs well with ORB as it is a feature extractor that produces binary descriptors. Moreover, the reason to pair FLANN with SIFT is similar lines in reasoning as FLANN handles floating-point descriptors more easily. The Brute-Force matcher is straightforward in its approach, such that it takes the feature descriptor from one image and tries to match it to all the possible feature descriptors in the concurrent image. The Brute-Force Matcher uses what is known as the Hamming distance for string-based feature descriptors. There are two kinds of Brute-Force Matchers: the regular match and the k-nearest neighbor match. The advantage of the k-nearest neighbor variant of the brute-force matcher is the number of matches specified by the user, and the k-nearest neighbor has been used in this body of work. Fast Library Approximation for Nearest Neighbor is also popularly known by its popular acronym FLANN. FLANN, unlike Brute-Force matcher, is not a single algorithm but is a collection of algorithms to perform a fast search on the nearest neighbors. The

advantage of FLANN over Brute-Force matcher is that it performs well on features that have high dimensionality and large datasets. FLANN has two parameters that need to be specified to get a specific algorithm from the FLANN package. The first parameter is a dictionary IndexParams which specifies the algorithm used in the feature matching process from the FLANN library. A second parameter is a dictionary known as SearchParams. The parameter SearchParams indicates to the algorithm the number of traversals that need to be made recursively. The higher the value assigned to SearchParams, the higher degree of precision is obtained, but this is achieved at the cost of an increase in computational time.

Chapter 3

DEEP LEARNING APPROACH FOR VISUAL ODOMETRY

Deep learning employed in Visual Odometry has shown a great deal of promise in the last few years. It is able to infer high-level representations in a more robust and dynamic manner, with very little fine-tuning. Most deep learning models are trained on large datasets with a lot of visual information, enabling them to be readily employed in diverse environments. The model presented in this body of work combines two models, Super point and SuperPoint and SuperGlue. The SuperPoint[10] model helps in keypoint detection, and the SuperGlue[28] model enables us to match these key points over consecutive sequences.

3.1 Superpoint

The primary step of any Visual odometry task is the extraction of keypoint. It has been shown repeatedly that convolutional neural networks are inherently better than the hand-tuned classical geometric methods. The SuperPoint algorithm is constructed such that there is no human supervision and is a completely self-supervised solution with the help of self-training. The self-supervised model is trained on a large synthetically generated ground truth interest point dataset rather than using a conventional dataset which is human annotated. The self-supervised training begins by letting the initial untrained detector be trained on a dataset of synthetic 2d-images. The task of the untrained detector is to locate key points in simple 2D images. The resulting trained detector after this task is called the MagicPoint detector. When the MagicPoint detector is tested for real-world images, it suffers from domain adaptation problems. To bridge the gap, a multi transform technique is applied, which is known

as homographic adaptation.

3.1.1 *Homographic Adaptation*

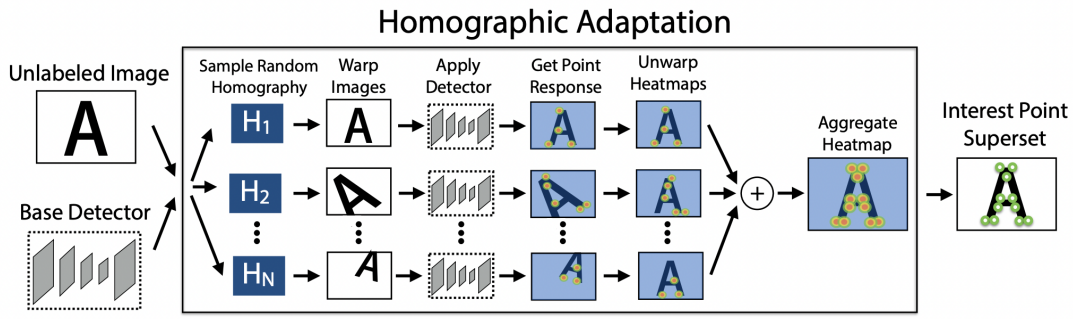

Figure 3.1: Process of homographic adaption.Figure Source [10].

Homographies are applied to images to provide a slight rotational variance to an image. Such a transformation helps the model view an image from different viewpoints just from a single image. The significant advantage of applying homographies is that they do not require any spatial information and can almost be applied to any image. This is why homographies form the core of self-supervised learning. The selection of the homography matrix needs to be carefully done. A well-selected homography must represent a plausible camera orientation in the real world.

To improve the ability of SuperPoint to work well on 3D objects, more than one homographic warp is required. The number of homographic warps for an image is considered a hyperparameter which can be denoted by the term N_h.The first homographic warp performed is not considered as performing any homographic adaptation and is considered as the zeroth point of the homographic adaptation process. To determine the optimal number of homographic adaptations, an experiment was conducted with small, medium, and large numbers of homographic warps applied. A small number of homographic warps equal to 10, medium equal to 100, and high which is equal to 1000.It was observed that 100 was the optimal number of homographic warps needed

to obtain optimal results. Above 100 warps, it was observed that it was diminishing in returns.

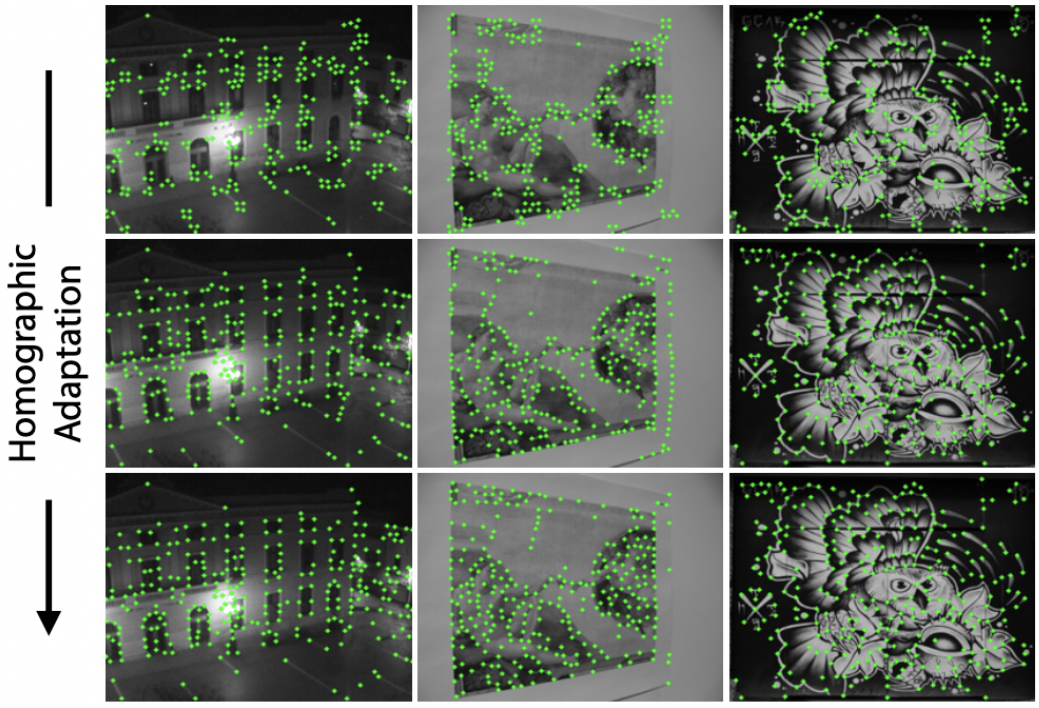

Figure 3.2: The figure shows improvement in keypoint detection with iterative homographic adaptation.The first row is before any homographic adaptation, the middle and last row shows an improvement with iterative homographic adaptation.Figure Source [10].

3.1.2 Problem Formulation

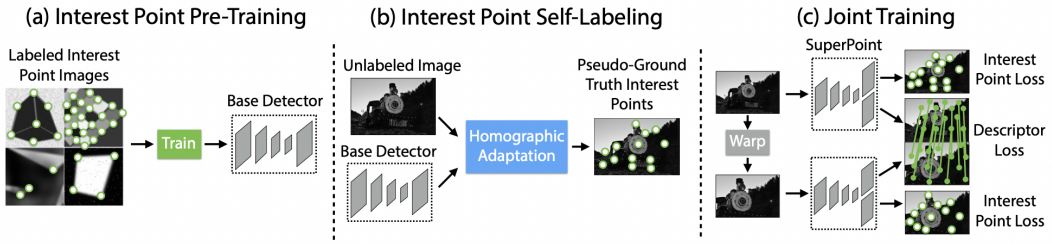

Figure 3.3: Figure represents SuperPoint self-supervised learning. a) MagicPoint training b) Homographic adaptation performed on MagicPoint to obtain SuperPoint c) Training of SuperPoint.Figure Source [10].

We define the keypoint of interest as $f_\theta()$, I as the input image, x the resulting keypoints and H a random photography such that:

$$x = f_\theta(I). \tag{3.1}$$

An idea keypoint in an image should be one unaffected by any homographic transformations. The function $f_\theta()$ must be covariant to as H follows:

$$Hx = f_\theta(H(I)). \tag{3.2}$$

The primary idea of homographic adaptation is to perform an empirical sum that is over a sufficiently large sample of random H. The resulting aggregation is what we improved with as the new and improved SuperPoint detector:

$$\hat{F}(I, f_\theta) = \frac{1}{N_h} \sum_{i=1}^{N_h} H_i^{-1} f_\theta(H_i(I)). \tag{3.3}$$

3.1.3 Architecture

The SuperPoint algorithm was designed to take images of full size and produce crucial key points and descriptors of fixed length in a single forward pass. The model has one shared encoder and two decoders. Each decoder is responsible for performing one function. One of the decoders is responsible for finding the key points and the other for producing the respective descriptor for the generated key point. This parallel process is quite different from the traditional approach as; first, the key points are detected, and then the descriptor is defined.

The functionality of the shared encoder is to reduce the dimensionality of the input image, which is passed into the model completely. The shared encoder comprises of spatial downsampling, convolutional layers and non-linear activation functions. The shared encoder reduces the dimensionality of the image of the dimensions of HxW as $H_c = H/8$ and $W_c = W/8$. The pixels we obtain from the resulting output are

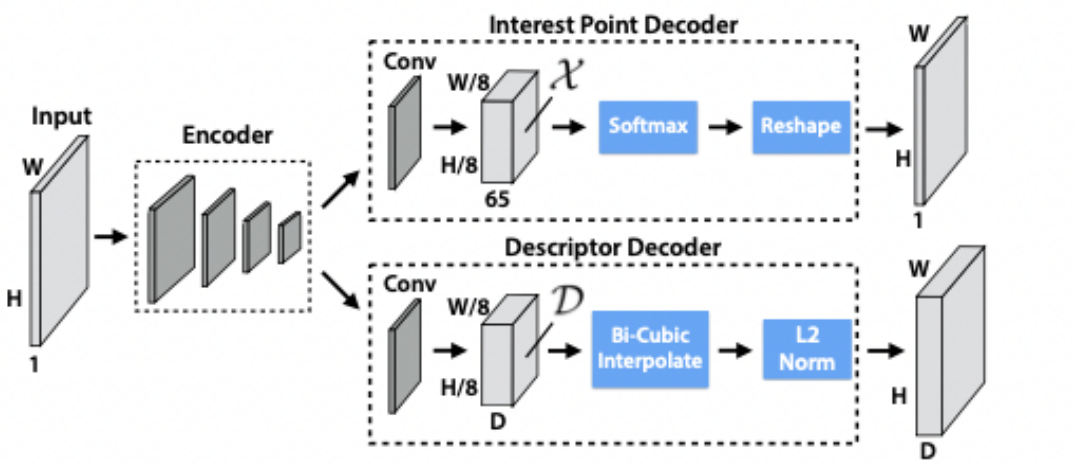

Figure 3.4: Architecture of the SuperPoint Algorithm. The model consists of one shared encoder, The two descriptors present are the keypoint descriptor and feature descriptor.Figure Source [10].

known as cells. The encoded images $I \in R^{H \times W}$ to a tensor $B \in R^{H_c \times W_c \times F}$ off a smaller dimension and greater channel depth which implies that $H_c < H$, $W_c < W$ and $F > 1$.

The Key Point decoder was designed with the intent to keep the computational cost at a minimum. To reduce the computational cost, to mitigate this an explicit decoder is constructed and it also avoids any unsmiling of layers. The detector is constructed such that it computes $X \in R^{H_c \times W_c \times 65}$ to give us an output tensor of the form $R^{H \times W}$.The number 65 corresponds to the number of channels. These channels correspond to an 8x8 grid which is non-overlapping in nature, and it also contains a dustbin dimensions where the points not considered keypoints are discarded. After each channel has a softmax performed on it, the dustbin dimension is discarded. The final reshaping is of the form$R^{H_c \times W_c \times 64} \Longrightarrow R^{H \times W}$.

The descriptor decoder computers a vector $V \in R^{H_c \times W_c \times D}$. These vectors undergoes two important processes before it is outputted the first one is bi-cubic interpolation and the second is L2-normalization. Bi-cubic interpolation is a mathematical operations which assists in the interpolation of data points in a matrix. The purpose

of the bi-cubic interpolation is to perform image scaling to obtain difference scales of the same image. The L2-normalization is performed as a normalization technique that transforms the matrix such that sum of the squares of the row is equal to 1.

The SuperPoint algorithm is trained on two datasets. Initially a base detector is trained on the Synthetic shape dataset to obtain the MagicPoint algorithm. The MagicPoint algorithm is further trained on the MS-COCO [17] dataset with homographic adaptations to the images to obtain the SuperPoint algorithm. The difference between the MagicPoint and the SuperPoint algorithms is only the descriptor decoder head which is present in SuperPoint and not in MagicPoint.

3.1.4 Model Loss

The loss of the SuperPoint model is the combination of the key point detector and the key point descriptor head losses. The loss of the key point detector is denoted by L_p. The loss of the key point descriptor is denoted by L_d. The final loss is defined as follows:

$$L(X, X', D, D'; Y, Y', S) = L_p(X, Y) + L_p(X', Y') + \lambda L_d(D, D', S). \tag{3.4}$$

In order to understand the above equation two images A and B can be taken. This is because the model is able to take an input of two images. The X,Y correspond to the coordinates of a key point in image A. The notation D represents the descriptor of that key point in image A. Similarly the X' and Y' are the coordinates of the key point in image B and the notation D' represents the descriptor information for the chosen key point in image B. The value λ multiplied to L_d is nothing but a constant factor present to balance out the final loss.

The loss of the key point detector can be calculated using the following equation

:

$$L_p(X,Y) = \frac{1}{H_c W_c} \sum_{h=1;w=1}^{H_c W_c} l_p(x_{hw}, y_{hw}). \tag{3.5}$$

In the above equation H_c and W_c are the modified dimensions of the image when they are passed through the shared encoder. The term x_{hw} belongs to X. The $y_h w$ belongs to Y which is the ground truth. The term l_p can be further be calcualted as follows:

$$l_p(x_{hw}, y) = -log \frac{\exp(x_{hwy})}{\sum_{k=1}^{65} \exp(x_{hw})}. \tag{3.6}$$

The descriptor loss is given by the following equation:

$$L(D, D', S) = \frac{1}{(H_c W_c)^2} \sum_{h=1;w=1}^{H_c, W_c} \sum_{h'=1;w'=1}^{H_c, W_c} l_d(d_{hw}, d'_{h'w'}; s_{hwh'w'}). \tag{3.7}$$

In the above equation, the terms d_{hw} and $d'_{h'w'}$ are the descriptor loss for the images A and B taken by the algorithm. The term S signifies the complete sequence of images that will pass through the algorithm. The term l_d in the above equation can be further be expanded as follows:

$$l_d(d, d', s) = \lambda_d . s . \max(0, m_p - d^T d') + (1 - s) . \max(0, d^T d' - m_n). \tag{3.8}$$

3.2 Superglue

Similar to how the classical geometric methods consisted of a keypoint detector and a key point, the keypoint matcher for the proposed deep learning model is SuperGlue created by the company Magic Leap. SuperGlue is essential to the localization and mapping approach for keypoint matching. There is a key point detector in the usual localization and mapping methodology that encompasses the front end. There is a backend that involves using an optimization methodology like bundle adjustment. The algorithm is currently designed to work with a set of image pairs and strives to take a learning-based approach to the features of the pair of images.

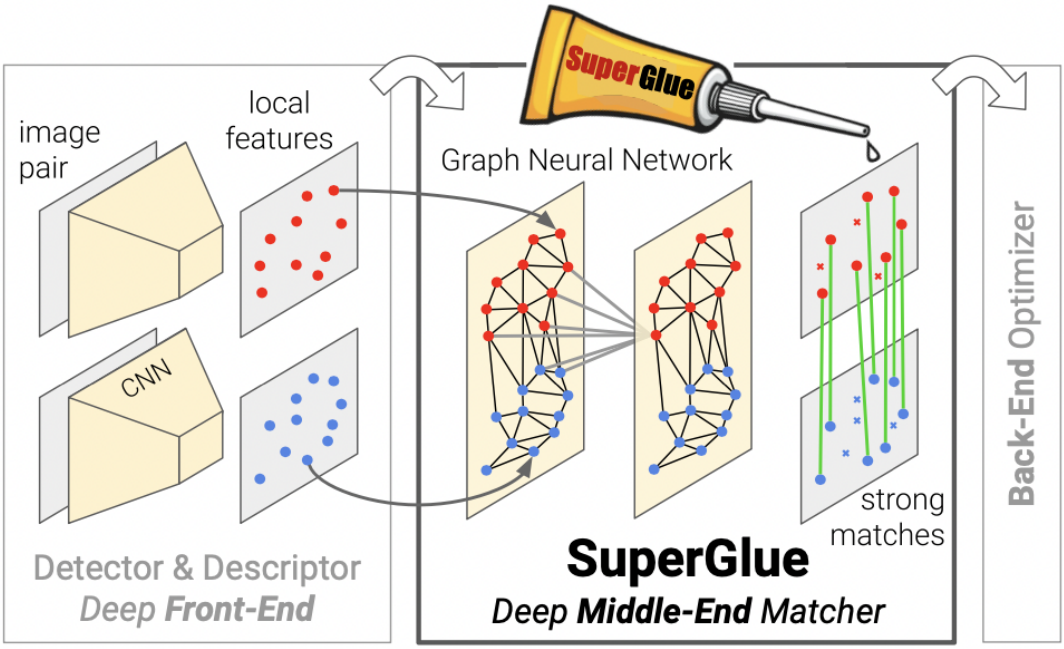

Figure 3.5: SuperGlue acts as middle end.It provides a multi-headed graph based attention mechanism and an optimal matching layer.Figure Source [28].

To formally define the problem, let us take two images, X and Y; the input information we would obtain from these two images are the key points and the descriptors. We can define the key points as k and the descriptor as m; these two values form the local features (k, m). The key points contain three main sources of information the x coordinate and y coordinate of the image and the level of confidence defined by c. The confidence provides us with information regarding the chosen key point. All this information put together represents the keypoint k, $k = (x, y, c)$. If the reprojection error is stated to be less than 1 pixel, the keypoint is considered stable. If the reprojection error is greater than 1 pixel but less than 5 pixels, it is said to be considered unstable. The algorithm is split into two modules, the Attentional Graph Neural Network and the optimal matching layer.

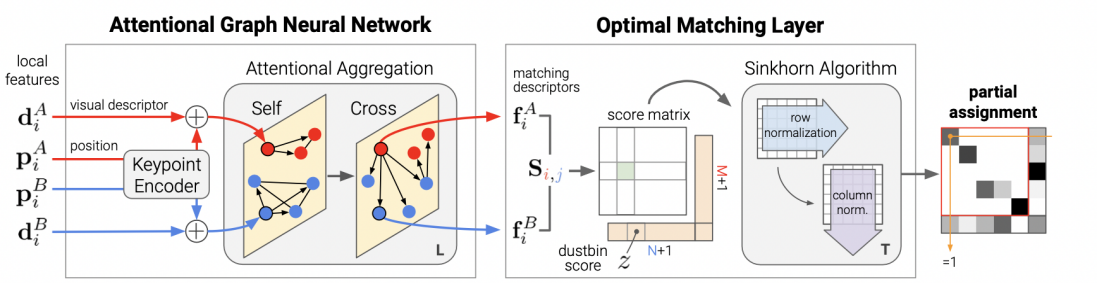

Figure 3.6: First block is the Attentional Graph Neural Network and the second block is the Optimal Matching Layer. The feature descriptors obtained from the Attentional Graph Neural Network is conveted to a soft partial matrix by the Optimal Matching Layer.Figure Source [28].

3.2.1 Attentional Graph Neural Network

The output of the SuperPoint is taken as the input for the SuperGlue network. The information that is obtained from the SuperPoint algorithm is the key point information and the key point descriptor information. The first block of the SuperGlue network, which is the Attentional Graph Neural network attempts to extract the degree of similarity between local descriptors.

Each key point is given a preliminary representation in the form of p_i^0. The key point is then passed through a key point encoder module. This key point encoder module is nothing but a Multi-Layer Perceptron (MLP). This multi-layer perceptron encodes this initial key point into a high-dimensional vector and the initial vector representation we obtain is as follows:

$$v_i^0 = d_i + MLP_{encoder}(p_i^0). \tag{3.9}$$

3.2.2 Multiplex Graph Neural Network

A multiplex graph[22, 24] is a graph in which the nodes of the graph are represented in different layers, and each layer consists of an unweighted connection between itself and other nodes. The nodes present in this graph are the key points that are

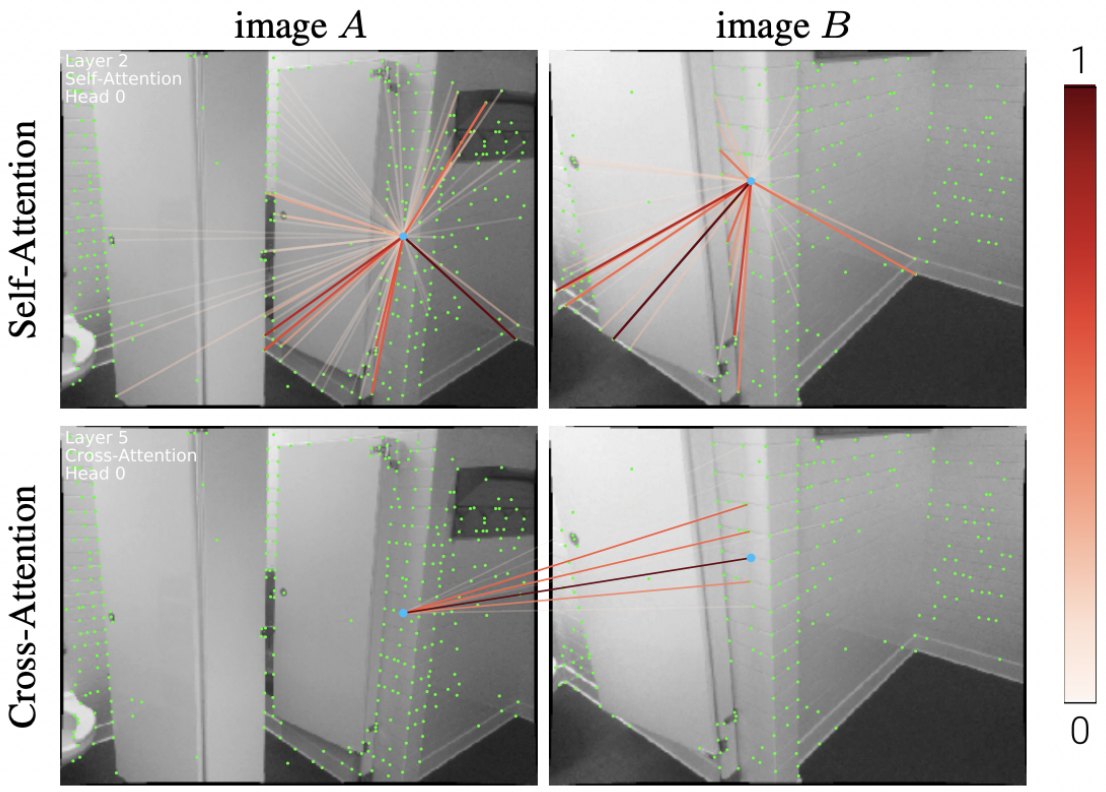

Figure 3.7: An illustrative example of self and cross attention.Figure Source [28].

common in both the input images. The multiplex graph generated has two kinds of unweighted edges. The first one is a self-directed edge E_{self} which connects a selected keypoint in an image to the respective edges continued within that image. The other type of edge is the cross-directed edge E_{cross}. These connect a specific key point to key points present in another image. The information between the two kinds of edges is connected with the help of a message-passing formula[7, 14]. Using this message passing formula, the initial states of these key points are updated as more and more key points are added from subsequent layers. The message passing formula(mpg) is key component in traversing between layers for a select image(img). If the layer is in an odd sequential layer then the edges of that layer are considered self and if even they are considered to be cross edges. The formula is as follows:

$$^{layer+1}x_i^{img} =^{layer} x_i^{img} + \left(\left[|^{layer} x_i^{img} || mpf_{\epsilon \to i} | \right] \right). \tag{3.10}$$

The message passing formula is computed with the help of an attention mechanism. The attention mechanism for E_{self} is self-attention [31] and for the E_{cross} it is cross-attention. The term α_{ij} is the attentional weight. The message passing formula is the sum of the weighted average as follows:

$$mpf_{\epsilon \to i} = \sum_{j:(i,j) \in Edge} \alpha_{ij} v_j. \tag{3.11}$$

In the above equation α_{ij} is considered as the weight of the attention mechanism. The weight factor α_{ij} is calculated with the use of a softmax function.The three main values to calcualte are the key, query and value. The inputs of the softmax function can be considered as a key and query. It is calculated as follows:

$$\alpha_{ij} = Softmax_j(q_i^T k_j). \tag{3.12}$$

A linear porjection of the feautres of the graph network helps in calulating the three main values. Each layer present in the graph network has its own unique set of key, value and query values. The following methods are the way in which the query, key and value are calculated:

$$q_i = W_1^{(l)} x_i^Q + b_1. \tag{3.13}$$

$$\begin{bmatrix} k_j \\ v_j \end{bmatrix} = \begin{bmatrix} W_2 \\ W_3 \end{bmatrix}^{(l)} + x_j^S + \begin{bmatrix} b_2 \\ b_3 \end{bmatrix}. \tag{3.14}$$

The above process softmax databse retrival helps in improving the expressivity of the network. This process also aids the network in learning specific features from a group of key points. Information can be retrieved based on either the features or key point locations which have embedded into x_i by the multi-layer perceptron. It also

helps in retrieving information from nearby key points or the relative positions of the nearby key points. The final descriptor vector for the selected image is represented in the form of a linear projection as follows:

$$f_i^A = W_{\cdot}^{(L)} x_i^{img} + b. \tag{3.15}$$

The above calculation is perfumed for all key point of the selected image and is done for all the images present in the sequence.

3.2.3 Optimal Matching Problem

After obtaining the final feature descriptors, it is converted into a score matrix. The new score matrix is from the combination of the feature descriptors and is sparse in nature. The S_{ij} is obtained by taking the Frobenius inner product of the obtained final feature descriptor:

$$\sum_{i,j} S_{ij} C_{ij}, \qquad C \in [0,1]^{M \times N}, \tag{3.16}$$

$$C\mathbf{1}_N \leq \mathbf{1}_M, \tag{3.17}$$

$$C^T \mathbf{1}_M \leq \mathbf{1}_N. \tag{3.18}$$

After the formulation of the score matrix, we would like to distill some key points out of it to prevent any occlusion. This is by a process where the rows and columns of the score matrix are augmented. Occlusion is when one key point in an image matches more than one key point in the subsequent image. Key points that do not have a match are also put into the dustbin. This augmented portion of the matrix is known as the dustbin to take into key point values that are not of use. The score matrix after the dustbin augmentation is as follows:

$$\bar{S}_{i,N+1} = \bar{S}_{M+1,j} = \bar{S}_{M+1,N+1}. \tag{3.19}$$

The final step is the optimization of the above equation. The formulation of the optimization problem is similar to the optimal transport problem[25]. The method used in solving this is an entropic regularization algorithm known as the Sinkhorn algorithm[8]. The entropic regularization process converts the sparse matrix with low entropy into a soft partial matrix of high entropy.

3.2.4 Model Loss

The loss of the superglue model is as follows:

$$L = -\sum_{(i,j)\in M} \log \bar{P}_{(i,j)} - \sum \log \bar{P}_{i,N+1} - \sum \log \bar{P}_{M+1,j}. \tag{3.20}$$

In the above equation the first term is the normal coupling matrix. The second term is when the column is augmented and the third term is when the row is augmented by the dustbin augmentation.The matching of key point is performed in the image domains. The minimization of the log-likelihood helps achieve an optimal value for the loss function, in a convex optimization sense. If there are key points which are not able to be matched then they are removed. The matching process is done using a re-projection method of key point present in the locality. The SuperGlue algorithm is a trained in a supervised manner to ensure there is a high degree of precision and recall. The supervised training process is done by using the ground truth. The primary aim of this loss function is to reduce the negative log value of P.

Chapter 4

EXPERIMENT

4.1 Karlsruhe Institute of Technology and Toyota Technological Institute Dataset

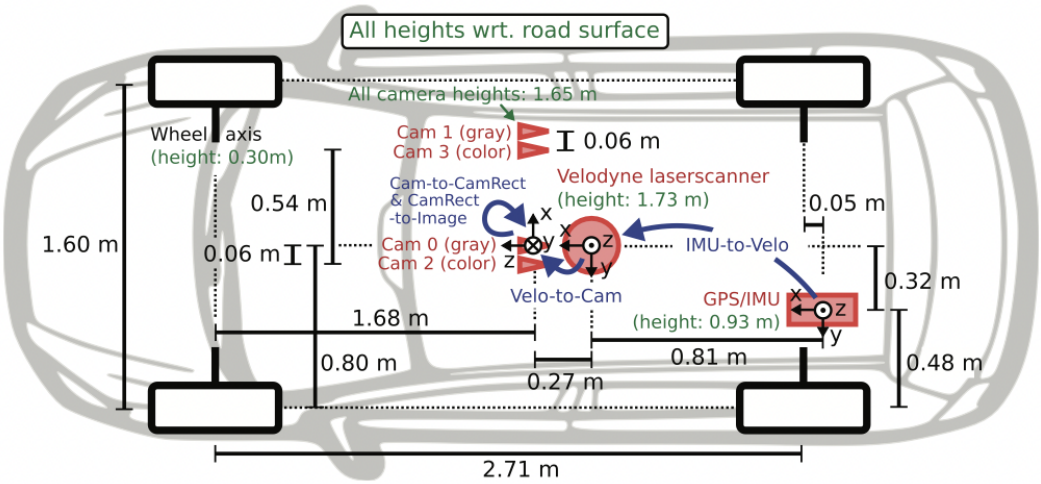

Figure 4.1: There are four cameras present. Camera 0 and Camera 1 gray scale images. Camera 1 and Camera 2 color images. Velodyne laser present on the center. The GPS is present on the rear side of the car.Figure Source [13].

The Karlsruhe Institute of Technology and Toyota Technological Institute dataset[13] is one of the most prolific benchmarks for Visual Odometry and SLAM. The dataset was created by capturing Visual Odometry data from the city of Karlsruhe in Germany. The curation of the dataset has been done using a Volkswagen Passat station wagon. This car has been fitted with many crucial sensors that give rich sensor data about some regions of the city of Karlsruhe in Germany. The car is fitted with four cameras—two cameras to capture grayscale video and two cameras to capture the RGB-graded images. The car is fitted with a Velodyne Laser. This laser helps us create point cloud images and give us a more 3-dimensional perspective of the cars surrounding. There is also a Global Positioning System on the car to give accurate

results of where the car is present. The dataset has been created to ensure diversity in data, and the sequences created have been from roadways, residential areas, cities, campuses, and even first-person data.

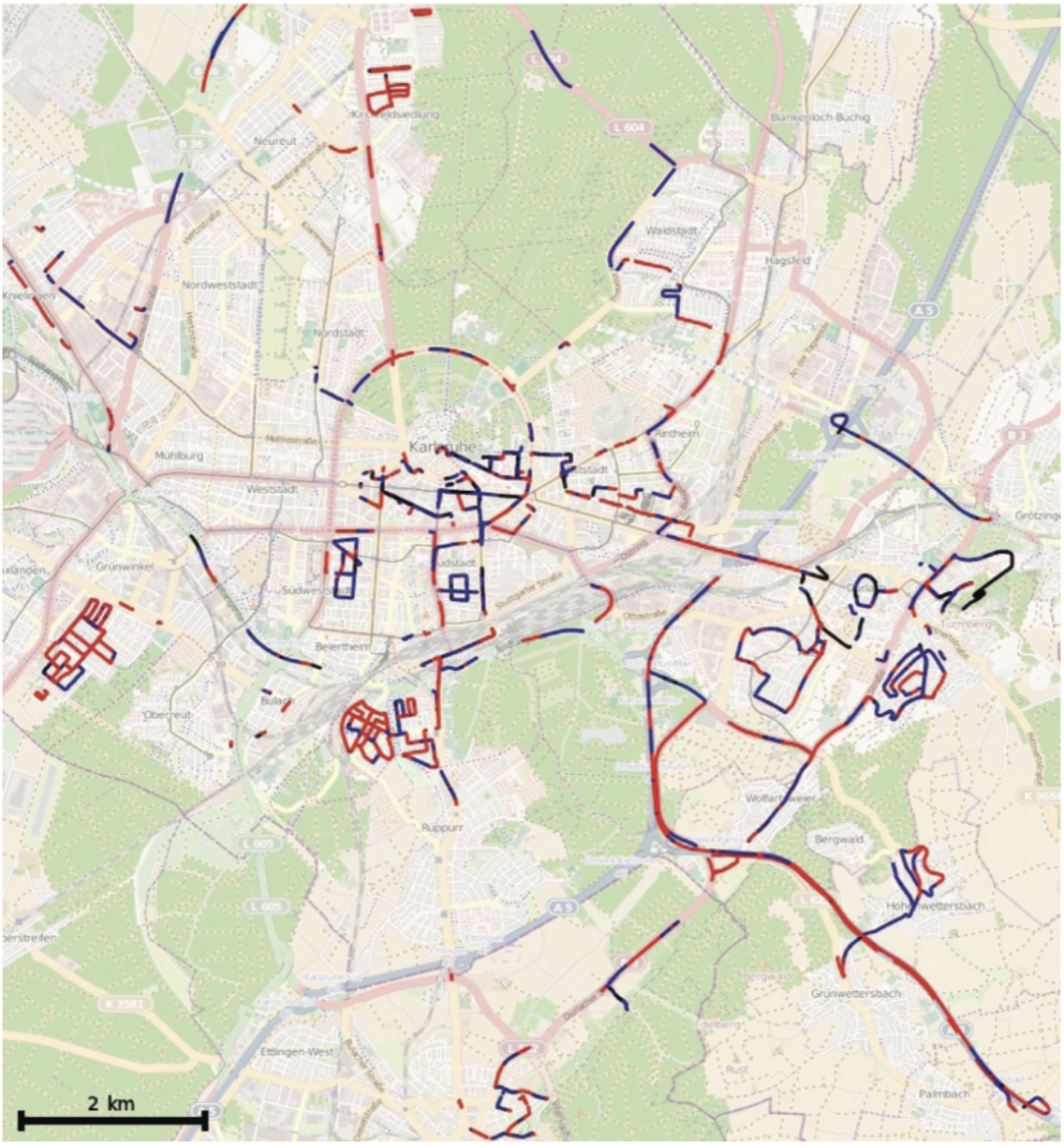

Figure 4.2: Regions in the city of Karlsruhe added in the dataset in red, black and blue. The colors represent GPS strength. Red for highest accuracy, blue having no corrections. The parts not considered are in black.Figure Source [13].

The visual odometry section of the data is the data is that will be used in the

experiment. This section consists of the two main files, namely sequence and poses. The sequence folder consists of various sequence folders present in it. Each sequence is a video that has been sliced into images. It is from these images that the motion is extracted. The second folder is the poses folder; this folder provides what is known as the ground truth information. The ground truth information helps give the true position of the car. It also gives a reference value to which the estimated path of our methods can be compared to the real value using the ground truth value.

4.2 Formulation

The experiments were performed on a system with a GTX1080 card and an intel i7 processor. The system also contained 8GB of Visual Random Access Memory capacity and 32GB of system Random Access Memory capacity. The performance metrics obtained are for this system configuration. The number of Sinkhorn iterations were kept at 400, for the SuperGlue algorithm.

Each pair of images in a sequence folder are first passed through the keypoint detector. After the key points have been extracted for both the images, they are then passed on to the key-pint matcher, which extracts the similarity and differences in the image. After this, we obtain the essential matrix. The essential matrix consists of both the estimated rotational and translational motion of the camera. This information is used to plot the estimated pose by the respective keypoint detector and matcher. The average deviation distance obtained between the estimated pose and ground truth pose is what we obtain as the Absolute Trajectory Error[30]. The Absolute Trajectory Error is an essential metric for comparing how different models stack up in performance to one another.

For testing, the models have been tested on eight different sequences. These sequences are divided into three types based on the nature of the track. The three

Absolute Trajectory Error					
Sequence Number	Seq Length	Sequence Type	ORB+BFM	SIFT+FNN	SP+SG
00	4540	DB	252.0834m	30.4049m	6.9047m
02	4660	UDB	407.8469m	37.966m	16.1529m
03	800	S	28.857m	2.2497m	2.8293m
04	270	S	23.2341m	1.0637m	1.0090m
05	2760	DB	125.2752m	12.3997m	10.7522m
06	1100	S	213.7643m	3.6854m	7.3692m
07	1100	UDB	77.3247m	11.9247m	12.2798m
08	4070	DB	243.1933m	17.5910m	11.8398m

Table 4.1: The Absolute Trajectory is a measure of how much the estimated pose of the agent has defined from the actual ground truth.

different types are straight(ST), defined bend (DB), and undefined bend (UDB). Straight sequences are those which have little to no angular bends to them. Defined bends are those sequences that have bends to them at a defined angle. Undefined bends are those sequences that do not seem to have any well-defined geometric pattern and are very skewed in their orientation.

From the results obtained, it can be observed that the difference between classical geometric methods and the deep learning models in the straight sequence is very minimal, and in some sequences, classical geometric models perform slightly better than the deep learning model. In defined bends, the deep learning model tends to be better than the classical geometric models. It is a similar case in undefined bends where the deep learning model performs better than the classical geometric model.

From the Absolute Trajectory Error table [3], it can be inferred that the SuperPoint and SuperGlue model can perform better than the classical geometric methods.

The model time log table					
Sequence Number	Seq Length	Sequence Type	ORB+BFM	SIFT+FNN	SP+SG
00	4540	DB	382.210s	784.340s	1321.062s
02	4660	UDB	423.637s	728.681s	1361.031s
03	800	S	69.522s	138.859s	233.532s
04	270	S	21.592s	46.176s	79.990s
05	2760	DB	229.439s	428.060s	801.014s
06	1100	S	95s	187.676s	321.219s
07	1100	UDB	89.824s	170.539s	363.146s
08	4070	DB	351.255s	651.527s	1210.541s

Table 4.2: The Model Time Log table provides the amount of time a specific model takes to run a sequence and is measured in seconds.

However, in sequences that are straight in nature, the difference between SIFT and the deep learning model is relatively minimal. In a specific instance, particularly in straight tracks, the classical geometric model performs better than the deep learning model.

The time taken for each model to compute a sequence has also been calculated. This gives the run-time efficiency of the model. From the table obtained, it is quite evident that the deep learning model has a substantially higher run time for every sequence in comparison to the classical geometric approaches. There is also a clear explanation as to why the deep learning model takes more time which is analyzed in-depth in the next section.

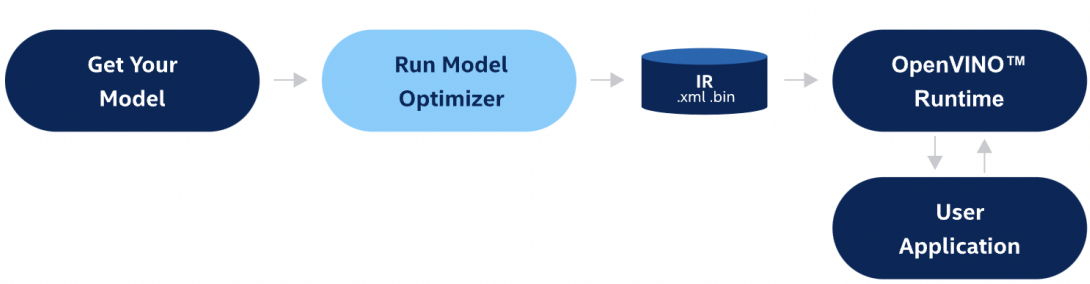

Figure 4.3: Workflow of OpenVINO toolkit. [1]

4.3 Superpoint Analysis

Inferential analysis is performed on the SuperPoint algorithm with the aid of the OpenVINO toolkit. OpenVINO is a toolkit that aids in the optimization and deployment of an Artificial Intelligence model onto various intel devices. The process of inferential analysis of the SuperPoint algorithm begins with obtaining the ONNX version of the algorithm. ONNX stands for Open Neural Network Exchange. The ONNX framework provides a computational graph model of the algorithm, which is embedded with the appropriate representation of the operations present in the model and the kind of data the model performs computation.

After obtaining the ONNX representation, it is passed through the first module of the OpenVINO toolkit, the model optimizer. The model optimizer is a multi-platform transition tool. It ensures the model is optimally converted to a format that can be easily deployed into multiple kinds of hardware. Once the model is passed through the model optimizer, the output we obtain is known as the Intermediate Representation. The intermediate representation provides two kinds of output files: the .xml file and a .bin file. The XML files provide the topology of the processed model, and the bin file provides the weights of the model. The model optimizer cuts certain parts of the model to optimize it for the target intel hardware. The model optimizer also provides the option to give a custom input shape that could be different from the original

model.

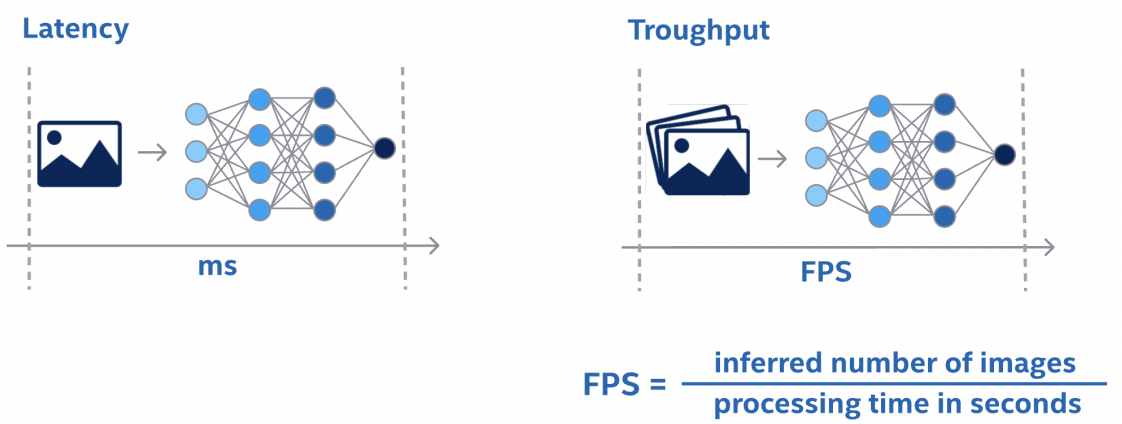

Figure 4.4: Pictorial representation of latency and throughput of a neural network model. [2]

The Intermediate Representation of the model is an input to the inference engine. The role of the inference engine is to provide the appropriate plug-in for different intel hardware such as CPU, GPU, and VPU. The plug-in used for the inferential analysis of this experiment is the Intel-Xeon CPU. Inferential performance for the Xeon processor can be obtained using the benchmark app. The two critical metrics for a neural network model are latency and throughput. Latency can be defined as the time it takes for a single image to pass through a model. Throughput can be defined as the time a model takes to infer a sequence of images.

4.4 Optimal Transport Overview

The optimal has become a significant cornerstone in the field of machine learning. It is a method that provides the ability to transmute one probability distribution to another. In the field of machine learning in domain adaptation, color transfer, and feature matching, to name a few of the tremendous applications of Optimal transport problems. Optimal transport also has two very desirable qualities, which are adept to machine learning problems. The first is the triangle inequality property which states

Inferential Results of SuperPoint	
Parameter	Output
Model Read Time	14.96ms
Model Compile Time	48.92ms
Average Latency	15.79ms
Minimum Latency	14.14ms
Maximum Latency	45.82ms
Throughput	252.56fps

Table 4.3: The above table provides the latency and throughput of the SuperPoint model.

that the sum of two sides of a triangle must be greater than that of the third side. The second property is that it provides symmetric functions; a function is said to be symmetric if the order of its argument does not affect the output of the function.

One of the first distance functions for the optimal transport problem is known as the Kullback-Leibler divergence. It was one of the first equations to give the notion of distance between two probability distributions. The Kullback-Leibler divergence can be mathematically expressed as follows:

$$D_{KL}(P||Q) = \int p(x) log \frac{p(x)}{q(x)} dx. \tag{4.1}$$

But the drwaback with the Kullback-Leibler divergence there are certain distributions which when mapped are close to one another, but according to the Kullback-Leibler divergence it equates the distance of the two probability distributions to be infinitely distant form one another. This clearly violates the triangle inequality property. To overcome this, another distance function was considered known as the Wasserstein distance. The Wasserstein distance can be calculated with a simple example if there is a certain amount of soil, which is an analogy to the first probability distribution. Then the cost to transfer this mass of soil to another pile at a distance is known as

the transportation matrix. There might be multiple transportation matrices that can fit the need, but there are constraints to finding the optimal transport matrix. The constraints for the optimal transport matrix are that the rows of the transport matrix must equate to the original soil mass, and the sum of the columns of the transport matrix must equal the cost of the target soil mass, this is where the alternative name of the Wasserstein distance is also known as the earht moving distance.

4.5 Superglue Sinkhorn Analysis

When the different modules of the SuperGlue module were analyzed, the module that took the most execution time and computational resources was the optimal matching module. The primary algorithm present in the optimal matching module is the Sinkhorn algorithm, specifically the log-based Sinkhorn algorithm.The analysis of the Sinkhorn algorithm begins with the formulation of the Optimal Transport Problem. The optimal transport problem involves moving one probability distribution to another. The optimal transport to be concerned with for this case is discrete in nature. The amount of mass of one probability distribution is moved to another probability distribution is defined as the transport matrix or, in the case of the SuperGlue model, the score matrix. The amount of effort needed to move this mass from one probability distribution to another. A parallel can be drawn here between probability distribution and the final feature descriptors, which are input to the optimal matching module in SuperGlue.

As stated, the goal of the Optimal transport problem is to find the most optimal form of transportation or the score matrix. The problem can be formulated in the following form:

$$L_c = min < S, P >, \tag{4.2}$$

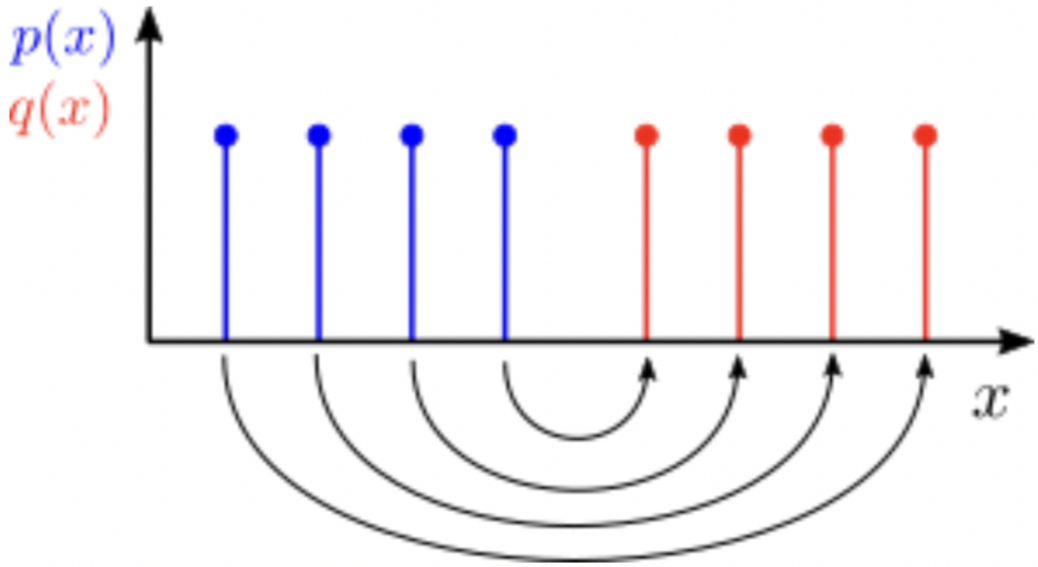

Figure 4.5: A visual representation of the Optimal Transport Problem. The discrete probability distribution p(x) is converted into another discrete probability distribution q(x)

$$P\mathbf{1}_N = a, \tag{4.3}$$

$$P^T\mathbf{1}_M = b. \tag{4.4}$$

The problem to be solved now takes the form of a convex optimization problem. The problem is formulated such that we maximize the transport matrix while keeping the costs minimum using what is known as the Wasserstein distance. The constraints for the optimal score matrix are that the rows of the matrix must sum up to the original probability distribution, and the columns of the matrix must sum up to the goal probability distribution. The above formulation is similar to how the optimal matching layer module optimizes the score matrix. Linear Programming can solve this, but as the size of the score matrix increases, the computational cost also drastically increases. This can be mitigated with the help of approximation methods which provide an answer which is very close to the optimal score matrix. The approximation method used in the optimization process is entropic regularization. Entropic

regularization ensures that a matrix either has a low entropy or a high entropy. A low entropy matrix is one that is sparse in nature, and a matrix with high entropy is said to have a matrix with an evenly distributed matrix. The Sinkhorn-Knopp, which was the first version of the Sinkhorn, is as follows:

Algorithm 1 Computation of $\mathbf{d} = [d_M^\lambda(r, c_1), \cdots, d_M^\lambda(r, c_N)]$, using Matlab syntax.

Input $M, \lambda, r, C := [c_1, \cdots, c_N]$.
$I = (r > 0); r = r(I); M = M(I, :); K = \exp(-\lambda M)$
$u = \texttt{ones}(\texttt{length}(r), N)/\texttt{length}(r)$;
$\tilde{K} = \texttt{bsxfun}(\texttt{@rdivide}, K, r)$ % equivalent to $\tilde{K} = \textbf{diag}(1./r)K$
while u changes or any other relevant stopping criterion **do**
  $u = 1./(\tilde{K}(C./(K'u)))$
end while
$v = C./(K'u)$
$\mathbf{d} = \texttt{sum}(u. * ((K. * M)v)$

Figure 4.6: Psudo-Algorithm of the Sinkhorn Algorithm.Figure Source[8]

The inputs to the Sinkhorn-Knopp algorithm are the desired Score matrix upon which the transformations need to be performed. λ also known as the entropic regularization factor; the lower the value of λ, the sparser the matrix is, the higher the value of λ the more evenly distributed the value of the score matrix. r and c are the original probability distribution and the target probability distribution. The algorithm normalizes the rows and columns in an iterative fashion. In the above algorithm the term K refers to the kernel of the cost matrix. The kernel of a matrix gives us a map of the null vector. It can be calcualted using the following equation:

$$K_{ij} = \exp\Big(- \frac{C_{ij}}{\epsilon}\Big). \tag{4.5}$$

Although the above algorithm provided great strides in the efficient computation of the optimal score matrix, there are some disadvantages to the Sinkhorn-Knopp algorithm. One disadvantage is for higher values of λ, the values for initializing the initial score matrix become very small. This cause two known issues which are numerical overflow and underflow. The overflow problem occurs when the matrix

requires more memory than the stack can provide. The underflow problem occurs when the values of the matrix are smaller than the smallest value the computer is designated to computationally handle.

The SuperGlue algorithm employs a variation of it known as the log-based Sinkhorn. The log-domain Sinkhorn converts the values to the log domain where large values of λ can be used and provide numerical stability. After the operations are performed similarly to Sinkhorn-Knopp, the values are converted back from the log-based domain. The disadvantage of the log-based Sinkhorn is that it consumes a lot of computational resources due to the log() and exp() function used, which causes a dramatic increase in computation time.

Chapter 5

CONCLUSION

5.1 Conclusion

In comparison to classical geometric methods, the proposed deep learning model can perform much better in almost all tracks of the KITTI dataset. The deep learning model can perform better than the classical geometric models for tracks that contain tracks with defined angular bends and tracks with undefined bends. A critical point to note is that the deep learning model has not been trained on the KITTI dataset. The SuperPoint model was trained on the Synthetic Shapes and MS-COCO datasets. The SuperGlue algorithm was trained on the ScanNet dataset [9], a relatively large dataset of indoor monocular sequences to obtain the indoor weights, and the PhotoTourism dataset [6] to obtain the outdoor weights. This clearly shows the ability of the model to learn high-level interpretations from large and diverse datasets and apply them to specific applications. From the Absolute Trajectory Error, it is evident that the deep learning model requires much less post-process optimization than the classical geometric models to get an estimated path closer to the ground truth.

But the proposed deep learning model also has some observed limitations. In tracks that are straight in nature, the classical geometric models tend to have slightly better accuracy than the proposed deep learning model. The proposed deep learning model also takes significant computation time, almost taking twice the time of the classical geometric model. The proposed deep learning model requires significant computational resources. Most of the computations performed by the deep learning model run on the GPU, making it unsuitable for edge devices such as robots that

have limited computational power.

5.2 Future Work

There are a number of avenues for future work. There are a plethora of optimal matching techniques present in the current literature. One of the methods which can prove to give a faster convergence rate to the optimal solution is the Adaptive Primal Dual Accelerated Gradient Descent [11]. There are also other versions of the Sinkhorn Algorithm which can be further looked into such as the Greenkhorn [16] and Screening Sinkhorn [4].

When it comes to the computational cost of deep learning models, it can be very high. More pertinent work can be done to make the models lightweight in order to be able to deploy them on edge devices. The current deep learning models can work on mitigating high computational costs incurred due to heavy matrix operations, which require significant parallel processing power. Since this body of work focuses on an outdoor dataset, future work may include the study to be conducted on an indoor dataset. One of the datasets known for its good monocular visual data is the TUM monocular visual odometry dataset [12].

REFERENCES

[1] Convert model with model optimizer. `https://docs.openvino.ai/latest/openvino_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html`, last accessed on 06/08/2022.

[2] Introduction to performance optimization. `https://https://docs.openvino.ai/latest/openvino_docs_optimization_guide_dldt_optimization_guide.html`, last accessed on 06/08/2022.

[3] Reference for absolute trajectory error code. `https://github.com/Shiaoming/Python-VO`, last accessed on 06/08/2022.

[4] Mokhtar Z Alaya, Maxime Berar, Gilles Gasso, and Alain Rakotomamonjy. Screening sinkhorn algorithm for regularized optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

[5] Mohammad OA Aqel, Mohammad H Marhaban, M Iqbal Saripan, and Napsiah Bt Ismail. Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, 5(1):1–26, 2016.

[6] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5173–5182, 2017.

[7] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[8] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[9] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[10] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[11] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.

[12] J. Engel, V. Usenko, and D. Cremers. A photometrically calibrated benchmark for monocular visual odometry. In *arXiv:1607.02555*, July 2016.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[14] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[15] Amila Jakubović and Jasmin Velagić. Image feature matching and object detection using brute-force matchers. In *2018 International Symposium ELMAR*, pages 83–86. IEEE, 2018.

[16] Tianyi Lin, Nhat Ho, and Michael Jordan. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *International Conference on Machine Learning*, pages 3982–3991. PMLR, 2019.

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

[18] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[20] Chuan Luo, Wei Yang, Panling Huang, and Jun Zhou. Overview of image matching based on orb algorithm. In *Journal of Physics: Conference Series*, volume 1237, page 032020. IOP Publishing, 2019.

[21] Suraya Mohammad and Tim Morris. Binary robust independent elementary feature features for texture segmentation. *Advanced Science Letters*, 23(6):5178–5182, 2017.

[22] Peter J Mucha, Thomas Richardson, Kevin Macon, Mason A Porter, and Jukka-Pekka Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *science*, 328(5980):876–878, 2010.

[23] Marius Muja and David Lowe. Flann-fast library for approximate nearest neighbors user manual. *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, 5, 2009.

[24] Vincenzo Nicosia, Ginestra Bianconi, Vito Latora, and Marc Barthelemy. Growing multiplex networks. *Physical review letters*, 111(5):058701, 2013.

[25] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Center for Research in Economics and Statistics Working Papers*, (2017-86), 2017.

[26] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[27] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[28] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[29] Davide Scaramuzza and Friedrich Fraundorfer. Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92, 2011.

[30] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[32] Deepak Geetha Viswanathan. Features from accelerated segment test (fast). In *Proceedings of the 10th workshop on image analysis for multimedia interactive services, London, UK*, pages 6–8, 2009.

[33] Ke Wang, Sai Ma, Junlan Chen, Fan Ren, and Jianbo Lu. Approaches challenges and applications for deep visual odometry toward to complicated and emerging areas. *IEEE Transactions on Cognitive and Developmental Systems*, 2020.

[34] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. An overview to visual odometry and visual slam: Applications to mobile robotics. *Intelligent Industrial Systems*, 1(4):289–311, 2015.