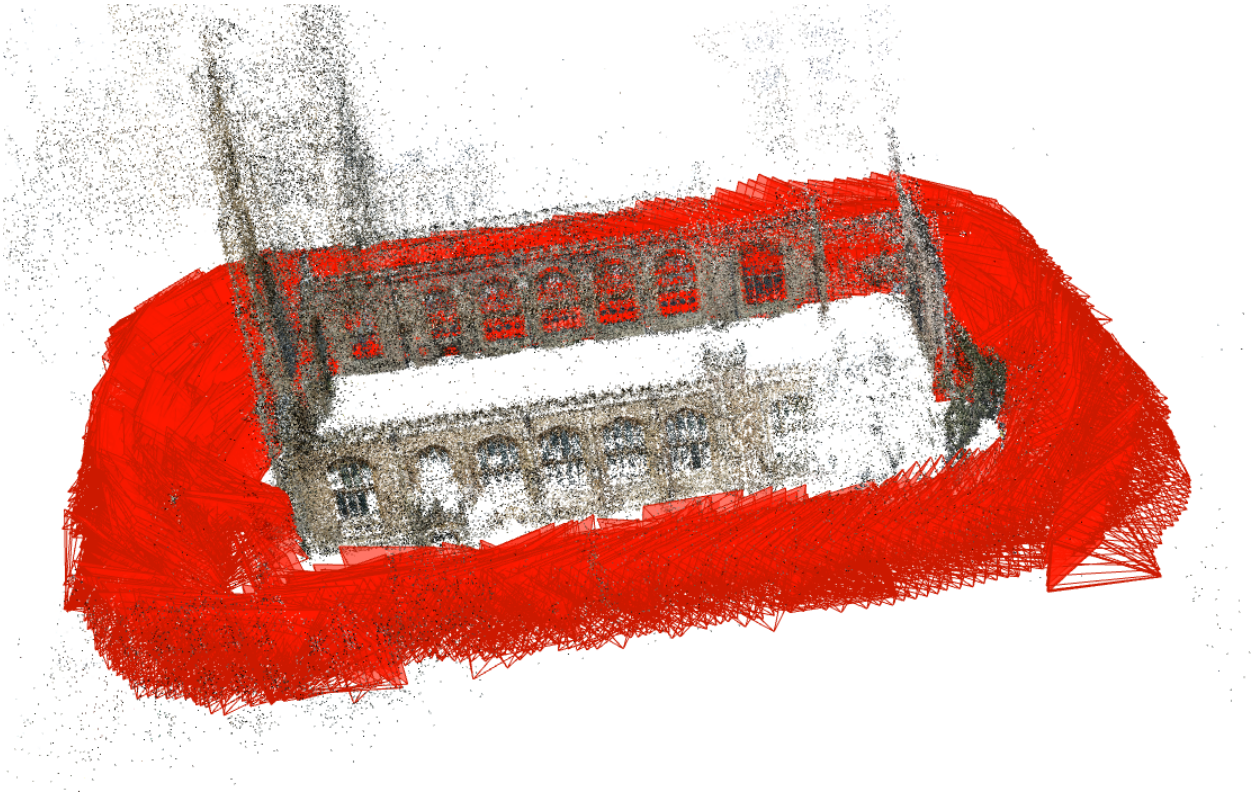
CHALMERS
UNIVERSITY OF TECHNOLOGY



Prioritized 2D-3D Matching for Visual Localization Revisited

Master's thesis in Complex Adaptive Systems

Earl Fernando Panimayam Fernando

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's thesis 2019:NN

Prioritized 2D-3D Matching for Visual Localization Revisited

Earl Fernando Panimayam Fernando

Department of Electrical Engineering
Computer Vision Group
Chalmers University of Technology
Gothenburg, Sweden 2019

Prioritized 2D-3D Matching for Visual Localization Revisited
Earl Fernando Panimayam Fernando

© Earl Fernando Panimayam Fernando, 2019.

Examiner: Torsten Sattler, Electrical Engineering

Master's Thesis 2019:NN
Department of Electrical Engineering
Computer Vision Group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A 3D model reconstruction of the Saint Mary's College landmark in Cambridge developed using the dataset provided by [26].

Typeset in LaTeX
Gothenburg, Sweden 2017

Prioritized 2D-3D Matching for Visual Localization Revisited
Earl Fernando Panimayam Fernando
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Determining the position and orientation of the camera from which an image was taken with respect to a scene representation both accurately and time efficiently is a bottleneck for many applications of Computer Vision like Augmented Reality and Autonomous Driving. Recent developments in Structure from Motion help us to reconstruct large scale 3D models which could be used for the camera pose estimation of an image, by matching its 2D image features with the 3D points in the model. Matching these 2D image features with all the 3D points is time consuming. For a good camera pose estimation only a few number of matches are required. The time taken for 2D-3D matching could be reduced by prioritizing the 2D image features for 2D-3D matching and once a certain number of matches are found, the camera pose could be estimated using the matched 2D-3D features. This thesis proposes a novel method to prioritize the 2D features from the image in a effective manner for 2D-3D matching by using the probabilities of these 2D image features to be in 3D model and the search costs for matching these 2D image features with the 3D points. The probabilities of the 2D image features are computed using a random forest classifier and the search costs of the 2D image features are the number of 3D points features which are having similar visual appearance to the query image feature. Through extensive evaluation with different datasets, I show that the proposed prioritization functions could be used for a time efficient visual localization.

Keywords: Prioritized feature matching, Random forest, Structure for Motion, K-means clustering, Knapsack.

Acknowledgements

I would like to thank my examiner Mr.Torsten Sattler for giving me the opportunity to work with this thesis and for guiding me throughout this thesis work.

Earl Fernando Panimaym Fernando, Gothenburg, August 2019

Contents

List of Figures

1
Introduction

One of the classic topics in the field of Computer Vision is Visual Localization, which is the problem of determining the camera position and orientation of an image taken with respect to some scene representation. Visual localization is the fundamental requirement behind many applications like Robotics [49], Autonomous driving [48], Augmented Virtual Reality [50], Navigation systems [51] and Location recognition systems [52].

One of the methods used for estimating the camera pose for a given image is by using image retrieval techniques to match similar images in the database and use the GPS locations of the matched images in the database to estimate the camera position of the query image [8, 9, 10, 11]. Though image retrieval based methods are fast, the estimated camera pose is not accurate which is not viable for applications like Autonomous Driving and Robotics. Another alternative method for visual localization is by using deep convolutional neural networks [58] to regress the camera pose [12] or the 3D scene coordinates [53, 54]. Though the methods proposed by [53, 54] gives a reasonably accurate camera pose estimation, they are limited to the scale of the scene considered for localization.

Most of the state-of-the-art techniques for large-scale visual localization reconstruct the 3D representation of a scene using Structure from Motion techniques [2, 28, 3] to estimate the camera pose of the query image. In general, 3D Structure-based visual localization techniques [4, 5, 6, 7] match the 2D local features extracted from the query image with all the features associated with the 3D points. The camera pose of the query image is later computed using RANSAC or DSAC [55, 54, 53] using the found 2D-3D matches. The time taken for matching all the 2D feature in the image and all the features corresponding to the 3D model becomes a bottleneck for 3D structure based visual localization algorithms as the size of the 3D model increases. To reduce the time taken for matching the 2D points with the 3D points and camera pose estimation, it would be ideal to reduce the number of features used for matching and use only the features which would have a higher chance of matching with a 3D point. [13] proposes to use only the query image features which are predicted by a classifier to be matchable during 2D-2D feature matching. This approach resulted in reducing the number of image features used for matching. [14] proposes to arrange the features based on the search costs in a ascending order and when N features are matched, the camera pose was estimated. [14] clustered the 2D image features of all the 3D points in descriptor feature space where each cluster is called as a visual word and the collection of all the visual words is called as a visual

vocabulary. [14] assumed that the search cost for every query image feature is the number of 3D points features in the visual which it had the shortest Euclidean distance with. Though this method reduces the time taken for matching, there might be some image features that might not have a corresponding 3D point but these features would be used for 2D-3D matching because of their low search cost.

Motivated by the approaches proposed by [13] and [14] to reduce the time taken for 2D-3D feature matching and to improve accuracy of the camera pose. In this thesis, I present a novel method to prioritize the 2D query image features to reduce the time taken for 2D-3D feature matching while maximizing the number of 2D-3D feature matches found for camera pose estimation. To determine the prioritization order of the 2D features in the query image, I estimate the probabilities for the 2D image features to have a corresponding 3D point, using a random forest classifier. The computed probabilities along with the cost for searching these 2D image features in the 3D model, computed by using the method proposed by [14] is used for prioritizing the 2D image features for matching.

1.1 Related Work

In this section, I explain about the various research papers that are closely related with the visual localization problem and the motivation for this thesis. I also discuss about the various researches on the techniques adopted for this thesis.

1.1.1 Visual Localization

One of the earliest localization methods was proposed by [40]. The authors of [40] proposed a robot navigation system which uses the scale invariant features [19] detected from the images taken from a stereo mounted camera on the robot to simultaneously reconstruct a 3D reconstruction of the real-world environment. The robot used this 3D reconstruction to determine its position. In contrast to the earlier method [35] proposed to retrieve the location of a query image related to a set of facades that were registered in a city map. Similar to this approach [36] used SIFT features [19] for matching with the image dataset with known GPS coordinates and triangulated the pose using the best two matches. [37] developed city scale visual localization using image retrieval using a vocabulary tree as suggested by [38]. [39] clustered the images with similar features to scene maps thereby reducing the number of images that were used for feature matching. But in most of all the above mentioned methods, the accuracy of the images were on a GPS coordinate system level, where as most of the real-world problems require more accurate camera positions.

To determine a higher quality of the camera position, [41] proposed to develop a 3D model of the scene using structure for motion [28]. They created new synthetic views of the 3D model and increased the size of their database. This database was later compressed into a set of original and synthetic views which efficiently covered all the 3D points that were relevant for camera pose estimation. They

adopted a vocabulary tree based approach a proposed by [37] to retrieve images which are similar to the query image. The features associated with the 3D points in the retrieved images were used for 2D-3D feature matching. The matched 2D-3D correspondences were later used to determine the camera position using a 3-point pose estimation algorithm [20]. The paper by [34] proposed to use a regression forest to classify every pixel in the query image to its corresponding 3D point which are later used to estimate the camera pose of the query image. [34] used RGB-D images for training their classifier. The estimated camera pose was not as accurate as the state-of-the-art 3D structure based localization pipelines. [26] proposed to train a convolution neural network to regress the camera pose of the input query image. This method still had a few shortcomings in terms of accuracy of the camera pose estimated when compared with the 3D structure based localization technique. [54] proposed to use a convolutional neural network to predict the 3D points associated with 2D query image features, which where later used to estimate the camera pose. [54] implementation of a end-to-end pipeline using convolutional neural networks to estimate camera pose without 3D construction had better camera pose estimation compared to traditional 3D Structure based localization when the scene used for localization was small but this method does not scale well with large-scale 3D models.

1.1.2 Prioritized 3D Structure based localization

To reduce the time taken for 2D-3D feature matching in 3D structure based localization, the 2D image features used for matching needs prioritized. [13] proposed to develop a classifier that predicts if a 2D image point would be matchable during 2D-2D feature matching or not. The 2D image features which were predicted by the classifier to be matchable were only used for 2D-2D feature matching. This ensures the probability of finding a match, thereby reducing the time taken for matching when compared with the traditional 3D structure based localization methods. Later, [14] proposed an appearance based strategy to prioritize the image features. They clustered the 2D image features belonging to all the 3D points in the descriptor feature space. Each cluster was called as a visual word. The search costs for an query image feature is proportional to the number of 3D points features in the visual word with which the query image feature had the shortest Euclidean distance. They prioritized the 2D image descriptors based on the search cost in an ascending order for 2D-3D feature matching. While all the above methods of searching corresponded to 2D-3D searches, [43] proposes to search 3D-2D in the reverse manner. They prioritize the 3D points used for matching depending on the number of images that were used for its reconstruction thereby covering the entirety of 3D model space. Once a matching 3D point is found, they propose to increase the priorities of the 3D points found in the same image along with the matched 3D point during 3D reconstruction. They stopped their search once N number of 2D-3D matches were found and the camera poses are estimated. [4] proposes to combine the techniques of prioritization based on appearance [14] and co-visibility [43] of the 3D points once a 3D match was established. [4] also proposes the strategy of actively searching for co-visible 3D points during 3D reconstruction and prioritizing then 2D image feature proposals for 3D matches and the 3D matches for 2D image feature matching together.

The idea of this thesis is to develop a prioritization function to reduce the time taken for matching 2D features with the all the features associated with the 3D points while the number of 2D-3D feature matches found is maximized. In this thesis, I propose to use the techniques used by [13] but instead of only using the 2D image points which are predicted to be matchable during 2D-2D feature matching, I propose to train a classifier to predict the probabilities of each 2D query image features to having a corresponding 3D point. These probabilities along with the appearance strategy of [14] is used to prioritize the 2D features for 3D matching.

1.2 Thesis Outline

The report for this thesis is divided into five chapters. The second chapter **Theory** gives an overall view of the problem behind the thesis and the main concepts used to attest this issue. In the next chapter **Methods**, a description of the strategies adopted to reduce the time take 2D-3D feature matching in 3D structure based localization while maximizing the number of 2D-3D matches found. The **Results** section consists of the experimental setup, evaluation methods and a discussion on the results. In the **Conclusion** an overall outcome of the thesis is given along with suggestions for future work.

2

Theory

Image-based local visualization can be defined as the problem of determining the position and orientation of the camera for a given query image. One of the most traditional strategies adopted to solve this problem is by creating a 3D scene using Structure from Motion. Using this 3D reconstruction, one could try to match all the 2D image features associated with the 3D points in the 3D scene with all the 2D image features in the input query image. But the problem with this method is the time taken for matching all the 2D image features with all the features associated with the 3D points. This time taken could be reduced if the 2D images features are prioritized using a prioritization function which maximizes the number of 2D-3D matches found while minimizing the time taken for 2D-3D feature matching and once N matches are found, they could be used for camera pose estimation. In this thesis, I propose to predict the probabilities of the 2D features in the query image to have corresponding 3D point using a random forest classifier [13]. These probabilities along with the search costs of the 2D image features [14] in the 3D model for matching, could be used to prioritize the 2D query image features. In this section, I discuss in detail about the concepts used for 3D construction using Structure from Motion, predicting the probabilities of the 2D image features to have a 3D point and calculating the search costs of the 2D image features.

2.1 Structure From Motion

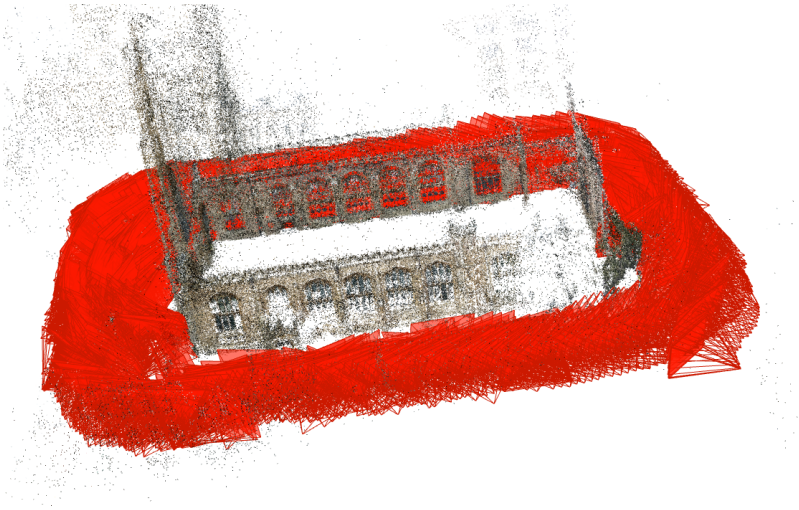


Figure 2.1: An example of 3D construction using Structure from Motion. (This is a 3D reconstruction of the St.Mary's College in Cambridge. The model was developed using the dataset provided in [26].)

To estimate the camera pose of the input query image in 3D-Structure based localization method, it necessary to develop the 3D representation of the scene. Structure from motion [3] is a method used to re-construct a 3D scene from a set of un-ordered images. In general, the following steps are involved in constructing a 3D scene using Structure from motion.

1. Feature Extraction
2. Feature Matching
3. Geometric Verification
4. Initialization for a reconstruction
5. Incremental SFM
6. Bundle Adjustment

2.1.1 Feature Extraction

To develop a 3D model from a bunch of un-ordered images, it is necessary to establish the relationship between the images. These relationships could be established by extracting different features in the images. The features extracted from the image should be invariant to geometric and to radiometric changes, so it would be possible to uniquely detect these features from other images which might be in a different orientation and position. Though there are many types of feature extractors like SURF [17] and learned descriptors [18], SIFT [19] and its derivatives [16] are few of the most widely used methods for feature extraction.

2.1.1.1 Scale Invariant Feature Transform

The interesting features detected in an image have to be invariant to scale and rotation, so that we could detect the same features in other images for feature matching. The matched features are later used for 3D scene construction. To obtain these interesting features which are supposed to be scale invariant, a Laplace of Gaussian is applied for different values of the scale parameter σ in the Gaussian Blur operator for the same image in different sizes. This Laplace of Gaussian acts as a blob detector to detect the local maximas and minimas across the scale and space. Applying a Laplace of Gaussian is computationally expensive. So [19] proposes to determine the Difference in Gaussian for a image which is an approximation for the Laplace of Gaussian, in different scales and spaces to detect the local maximas and minimas (a visual representation of this is shown in figure 2.2). The difference between the Gaussian blurring of an image with two different scale parameters gives the Difference of Gaussians. The detected local extremas in the images over scale and spaces are defined as the key locations for keypoints.

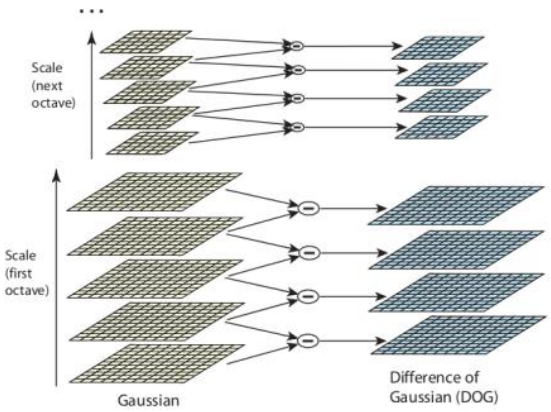


Figure 2.2: For each octave of the scale space, the images are convolved to give a set Gaussian blurred images for different values of σ. Images next to each other are subtracted to give a Difference of Gaussians. These Gaussian images are downsized by a factor and the same process is repeated. This image was taken from [19].

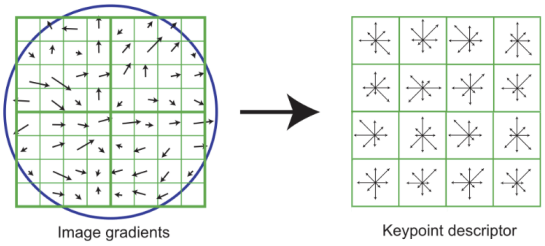


Figure 2.3: The image in the left shows the direction and magnitude of a gradients around an interesting point. The image in the right shows the accumulated orientation histogram of the gradients from the image in the left. This image was taken from [19].

To improve the accuracy of the keypoints locations [19] proposes to use a Taylor

series expansion of the scale space. The extremas were rejected if the intensity at that point was less than a threshold value of 0.03. Since the DoG have high response to edges, [19] proposes to use a concept similar to Harris corner detector to eliminate the edges. [19] proposes to give an orientation to every keypoints to achieve an invariance to rotation of the image. [19] also proposes to compute the gradients around the keypoints based on its scale. An orientation histogram of 36 bins for 360 degrees is used to determine its highest peak and any peak above 80% of the highest peak. These peaks are used to compute the orientation of the keypoints. The description needs to be robust to changes in appearance. So [19] proposes to create keypoint descriptor by first computing the gradient magnitude and orientation at each image sample point in a 16 $\times$ 16 neighbourhood around the keypoint location. The gradient magnitudes are then down weighted using a Gaussian. The 16 $\times$ 16 neighbourhood then is divided in 16 sub-blocks of size 4 $\times$ 4 and for each sub-block a 8 bin histogram is created. There are a total of 128 bins to represent each SIFT features.

2.1.2 Feature Matching

To represent a real-world scene in a 3D model, the features representing a point in the real world should also represent the same point in the 3D model. Therefore all the features representing a point in the 3D model should be matched to develop a 3D model. But matching all the SIFT features in the images with each other is time consuming. So [19] proposes to use a variation of the nearest neighbour algorithm [57] to determine the closest neighbour in the database of keypoints. But the problem with this method is that there might be a large number of points with almost comparable distances, so the 2D points might not be matched correctly. [19] suggests to find the top two best matches d_1, d_2 for a feature. If the ratio of distances $\frac{d_1}{d_2} > 0.8$, the candidate will be rejected. The output of feature matching will be a set of image pairs with each image pair containing a set of feature matches. Since the matched features represents the same scene in every image, these matched features would represent the same point in 3D model. The matched features will be later used for the estimation of the camera pose in the 3D model construction. The figure 2.4 is an example of the feature matching process.

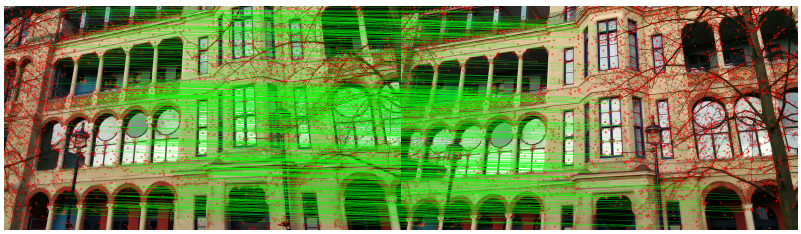


Figure 2.4: This figure shows feature matching between two images of the same building with different camera positions. The red dots in two overlapping images show their corresponding SIFT features and the green lines highlight the matches. The images used in this figure were taken from the Cambridge landmark dataset [26].

2.1.3 Geometric Verification and Pose Estimation

The matched image features in the previous section might not be always correct, so these matches needs to be verified. If a 3D point has two matched 2D features in two different images, these points could be geometrically transformed using projective geometry [2]. If sufficient number of features could be mapped using projective geometry, then the image pairs are overlapping [31, 32]. A Random Sample Consensus type sampling algorithm [20] is used to eliminate outlier correspondences while matching and to determine the geometric transformation. The estimated geometric transformations (Fundamental matrix) is used to estimate the Essential matrices. And the camera poses of the images can be computed from the essential matrices as proposed in [2] and by performing the Cheirality condition for triangulation.

2.1.4 Incremental Structure From Motion

Once the camera positions of the images and features matches are obtained. It is possible to construct the 3D model by triangulating all the matched features in the images using the camera poses of the images. All the matched 2D features should represent the same 3D point in the 3D model. To reconstruct the scenes in the images to a 3D model, [3] uses the pairs of images that were obtained from feature matching. The images chosen for 3D structure initialization are very carefully selected from a set of images which have many overlapping matched features [30]. The robustness, performance and the accuracy of the model depends on this initialization [3]. New images are registered to the model and their camera parameters are calculated using Perspective n point problem on the new image 2D features and the triangulated 3D points. A new 3D point is added to the list of already know 3D points once a pair of images having the same 3D correspondences is registered. The newly registered 3D points and the camera poses have high level of uncertainties as they are highly correlated. To improve the quality of the newly registered points and the camera poses of the images, [3] proposes to perform bundle adjustment on these newly registered 3D points and camera pose estimates.

2.2 3D structure based localization

For a query image similar to all the images that were used for 3D reconstruction, there might be few 2D features in the image which would have a corresponding 3D points. By establishing these set of 2D-3D correspondences, the camera pose of the given query image can be found using RANSAC [20]. To obtain these 2D-3D feature correspondences, the most naive approach that could be followed is by matching all the 2D image features with all the features that were used to triangulate a 3D point. If the viewpoints of the query image is similar to the viewpoints of the images used to build the SFM model and if the size of the 3D model is small, it is be possible to find the 2D-3D matches. But as the size of the model increases, the time taken for matching also increases. To overcome this bottleneck, it is necessary to reduce the number of 3D points used for matching. In general to overcome this bottleneck, a vocabulary tree based search approach is used to retrieve the 3D points that are

closely related with all the input image features. These 3D points are later used for 2D-3D feature matching [41]. [13] proposes another method to reduce the number of input features that are used for feature matching. They propose to use a classifier which predicts whether a 2D query image features would be used for 2D-2D feature matching or not and use only the features which were predicted as matchable for 2D-2D feature matching. Alternatively, [14] clustered the descriptor space based on the visual appearance of the 2D image features in the descriptor space. These clusters are called visual words. The input images features are then prioritized for 2D-3D feature matching depending on the number of 3D point features in the visual word having the shortest euclidean distance with the 2D input image feature. They propose to stop the 2D-3D feature matching once N number of matches were found. Motivated by [13] and [14], I propose an improvised way of prioritizing the features using the probabilities of the the 2D image features to be in a 3D model and the costs for searching these 2D features in the 3D model for matching [14] and stop the matching once N matches are found.

2.3 Predicting Probabilities

The probability of finding a 3D point for a given 2D query image feature can be computed using by training a classifier to predict if the 2D query image feature would have a corresponding 3D point or not.

2.3.1 Decision Trees

A decision tree is a classifier which works on tree like decision graph with a root node, internal nodes, branches and final end nodes or leaf nodes. Unlike trees, a decision tree classifier grows from the root node to the leaf nodes. The leaf node represents all the classes of the classifier. At every node, except the leaf node a decision is taken by randomly testing an attribute. Based on the decision, the nodes are split into branches and at the end of the branches there is either another internal node or a leaf node to determine the class. The figure 2.5 shows an example of how a decision tree works.

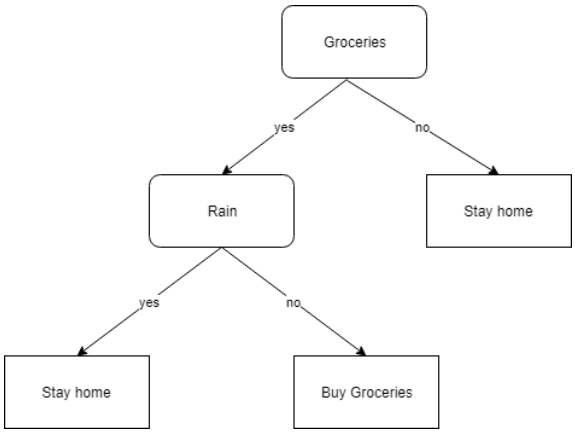


Figure 2.5: The image shows an example of how decision tree works. At every node a random attribute is test to determine whether to go for grocery shopping or not.

2.3.2 Decision Tree Learning

The decision tree algorithm uses the strategy of decision trees to predict the outcome of an observation [21]. A decision tree learning algorithm selects the best variable for splitting of the items at each node using many types of metrics [22]. The two commonly used type of metrics are Gini Index and Information Gain.

2.3.2.1 Gini Index

Gini index or Gini impurity measures the probability of an randomly chosen element from a set, to be miss-classified. This is also a measure of the purity of the node that is, if the Gini index at a node is 0, then all the elements in the node belong to the same class. The Gini impurity can be expressed as the sum over the product of the probability (p_i) of an item of the class i and the probability ($1 - p_i$) for miss-classifying the same item belonging to the class i [22]. In general, when there are K number of classes in a set and p_i is the distribution of the class i in the set, the Gini Index can be simplified and expressed as

$$G_i = \sum_{i=1}^{K} p_i \sum_{i=1}^{K} (1 - p_i) = 1 - \sum_{i=1}^{K} p_i^2 \text{ where i} \in \{1, 2, ...K\} \tag{2.1}$$

The best splitting decision variable is the variable with the maximum Gini gain.

$$G(T, a) = G_i(T) - G_i(T, a) \tag{2.2}$$

Here G(T,a) is the Gini gain for splitting the current node using a decision variable a, $G_i(T)$ is the Gini Index of the current node and $G_i(T, a)$ is the weighted sum of the Gini indexes in the child nodes when the splitting decision is based on a decision criteria a. The weight for a child node while calculating $G_i(T, a)$ is the ratio between the number of items in that child node and the total number of items in the node which has been split based on the decision variable a (parent node).

2.3.2.2 Information Gain

Information Gain is based on the concept of the entropy at a particular node and the change in entropy if a decision to divide the node using an attribute or a feature. Entropy H(T) can be defined as

$$H(T) = -\sum_{i=1}^{K} p_i \log_2 p_i \tag{2.3}$$

Here p_i are the percentage of each class i and i is the number of classes$\in \{1, 2, ...K\}$ present in a child node from a branch in the tree. The best splitting decision variable is the decision variable which has the maximum information gain.

$$I_G(T, a) = H(T) - H(T, a) \tag{2.4}$$

Here $I_G(T, a)$ is the information gained for making a decision split using the decision variable a, H(T) is the entropy in the parent node and H(T,a) is the weighted sum of the entropies in the child nodes when the the splitting decision is based on the decision variable a. The weight for a child node while calculating $I_G(T, a)$ is the ratio between the number of items in that child node and the total number of items in the node which has been split based on the decision variable a (parent node).

2.4 Random Forest

A random forest is a ensemble of decision trees and the output of the model is based on the majority of votes for particular class (an example of a random forest model is given in the figure 2.6).

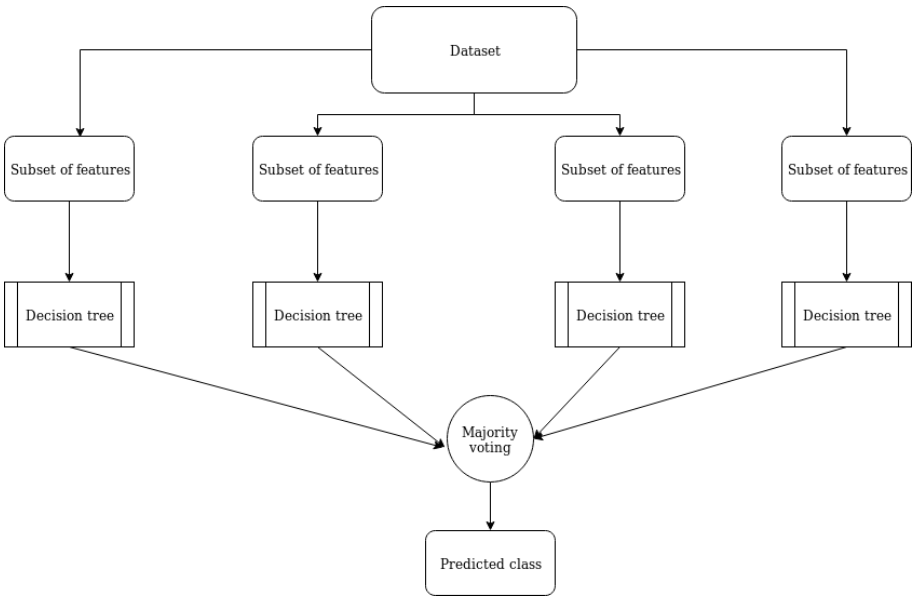


Figure 2.6: Example of how a random forest classifier works.

[24] proposes to grow each decision tree on a set of randomly selected features, whereas [23] proposed to grow each decision tree on set of randomly selected features

and randomly selected observations. This method of training on randomly selected features is called feature bagging. The decision splits at the nodes in this type of forests are along one particular feature. For this thesis, I have used the random forest model as proposed by [23].

2.5 Calculating Search Costs

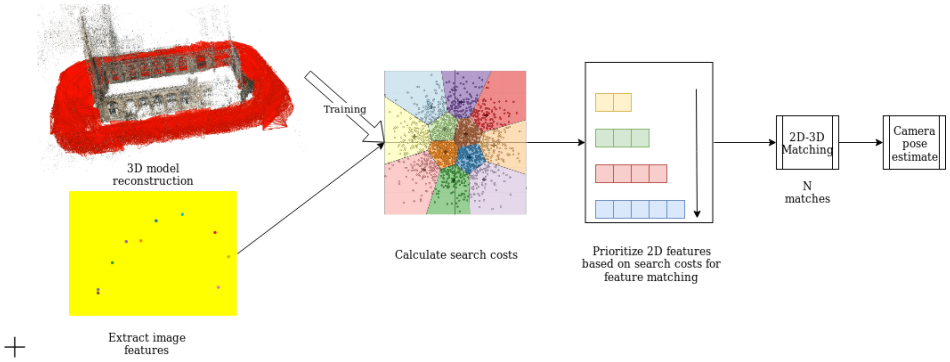


Figure 2.7: Shows the structure of the 3D structure based visual localization pipeline proposed by [14]. At first, a 3D model is reconstructed using a set of images of a location. The descriptor space of the image features used for 3D reconstruction is later clustered into visual words by training a clustering algorithm. For the localization of a given query image, all the interesting images features in the query are extracted first. The search costs of the extracted features are calculated in the next stage based on the number of descriptors in the visual word which has the shortest Euclidean distance with the query image feature. In the next stage, the query image features are prioritized for 2D-3D feature matching based on their search costs. Once N number of 2D-3D features have been found, the camera pose is estimated.

[14] proposes a visual vocabulary based search where they clustered the descriptor feature space containing all the 2D image descriptors belonging to all the 3D points into clusters called visual words. [14] computed the search cost for an query image feature as the number of 3D point features in the visual word with which the query image feature has the shortest Euclidean distance. An example of the 3D structure based visual localization pipeline proposed by [14] is given in the figure 2.7. By this method we can reduce the number features that needs to matched to determine a matching 3D point thereby reducing the time taken for 2D-3D matching.

2.5.1 K-means Clustering

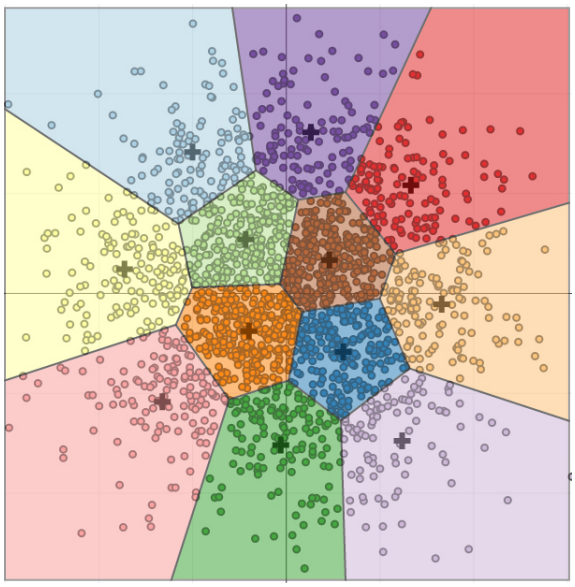


Figure 2.8: The different colours represents the different visual words and the points inside each partition represents the 3D point descriptors, the plus sign indicates the cluster centers of the visual words. This image was taken from `https://plot.ly/~MariaKu/97/voronoi-polygons-and-k-means-clustering/#plot`.

The descriptor space of the 3D points can be clustered different visual words by training a K-means clustering algorithm using the descriptors corresponding to the 3D points. The K-means clustering algorithm is an iterative unsupervised learning algorithm. The idea of K-means clustering algorithm is to divide the n observations $(x_1, x_2, ...x_n)$ having the same dimension, into K clusters C = $C_1, C_2, ..C_K$ by minimizing the sum of squares of the distances between the mean point of the observations in a cluster (cluster center) and the points in the cluster for all the clusters. For our task, the number of clusters K is the number of visual words in the visual vocabulary. The objective function for a K-means clustering problem can be viewed as:

$$\underset{C}{\operatorname{argmin}} \sum_{i=1}^{K} \sum_{x \in Ci} ||x - \nu_i||^2 \tag{2.5}$$

Here ν_i is the cluster center in C_i. The steps involved in a K-means clustering algorithm [25] is explained below :

1. Initially K number of cluster centroids are assigned randomly throughout the input data dimensions.
2. All the input observations are assigned to different cluster based on the distance between observation and the cluster center.
3. The cluster centroids are moved to the new means of the clusters
4. Step 2 and 3 are repeated in a sequence until the cluster centers no longer changes

The figure 2.8 is an example of the clustered descriptor space with visual words and their respective cluster centers. The search cost for an input query image feature

can be determined by using the number of 3D point descriptors in the visual word with the shortest Euclidean distance to the query image feature.

2.6 Prioritization Function

The main aim of this thesis work is to reduce the time taken for matching the 2D query image features with the 3D points for 3D-Structure based localization. My proposal in thesis is to prioritize the 2D query image features using a prioritization function which minimizes the search cost for searching the 3D model under the constraint of maximizing the probability of find a matching 3D point. To understand the approach used in this thesis, it necessary to know about the knapsack problem.

2.6.1 Knapsack

Let us consider if there are n objects of different weights w_i and each item has values v_i. The objective of a knapsack problem is to determine the best combination of items that can be fit into a bag (knapsack) with a weight constraint of L while maximizing the value of the items in the bag. A pictorial representation of the knapsack problem is given in the figure 2.9. The objective function of a knapsack problem could be expressed as :

$$\max \sum_{i=1}^{n} v_i x_i \tag{2.6}$$

Subject to $\sum_{i=1}^{n} w_i x_i \leq L$and $x_i \geq 0$.Here x_i represents the number of copies of the items that were made.
A knapsack problem is called as a 0/1 knapsack problem if there can be no copies of the items can be made. The then the objective function of a 0/1 knapsack problem can be expressed as :

$$\max \sum_{i=1}^{n} v_i x_i \tag{2.7}$$

Subject to $\sum_{i=1}^{n} w_i x_i \leq L$and $x_i \in \{0, 1\}$. Here x_i is either 0 or 1 depending if the the item i is included or not.
The number of combinations that needs to be checked to find the best fit for the bag increases exponentially as the number of items in the problem increases and this problem is considered as an NP-complete problem [60]. The 0/1 knapsack problem is similar to our problem of developing a prioritization function which minimizes the search costs of searching the 3D model and also maximize the probability of find a 3D point match.
In our problem, the weights are the search costs and the probabilities are the values. The time complexity of an algorithm represents the time taken by an algorithm so solve a given problem. The time taken by an recursive algorithm to determine the solution for a knapsack problem is $\mathcal{O}(n2^n)$ [64]. A problem is solvable in polynomial time if the running time of the algorithm is bound by some polynomial of the input size for the problem. Through bottom-up dynamic programming, the the time complexity of a knapsack problem can be reduced to $\mathcal{O}(nL)$ where n is the number

of items and L is the capacity constraint [62]. $\mathcal{O}(nL)$ is also called psuedo-polynomial because when L is small, the algorithm is quick to find a solution but as L increases the time complexity also increases drastically. The search cost constraint for the prioritization problem cannot be greater than nP where P is the maximum search cost of all the n items for prioritization. The time complexity using bottom-up dynamic programming can be expressed as $\mathcal{O}(n^2P)$ [61].

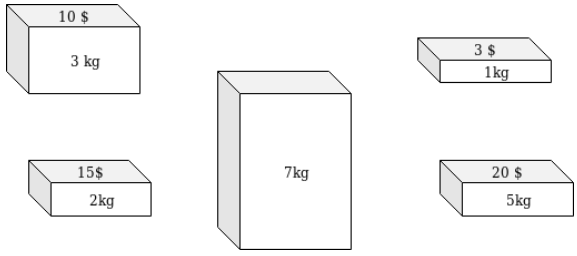


Figure 2.9: The image shows an example of the knapsack problem where the objective is to fit items into a bag of the weight constraint of 7 kilograms while maximizing the value of the items in the bag.

3

Methods

The main objective of this thesis is to prioritize the 2D image features of the query images in an order that minimizes the search costs for searching the 3D model and also maximizes their probability of finding their corresponding 3D points. To fulfill this objective, the following steps were done.

1. Develop a 3D model of the test images using Structure for Motion.
2. Obtain the probabilities of matching the 2D query image features in the 3D model.
3. Obtain the search cost for every 2D query image features in the 3D model.
4. Combine the search cost and the probabilities of the query 2D image features and develop a prioritization function

In this chapter, I discuss about how the above mentioned steps were performed and I also explain about the various methods used to prioritize the 2D query image features.

3.1 Outline

The figure 3.1 gives an overall outline of the method proposed in this thesis. Initially the 3D reconstruction of the scene could be obtained using Structure from Motion. Using the images features in the 3D model and features not in the 3D model, a random forest classifier is trained to predict the probabilities of the 2D query image features for having a 3D point. Along with this a K-means clustering algorithm is trained to cluster the 2D image features of the 3D model to different visual words as proposed by [14]. For a given query image, the probabilities of the query image features having a 3D point and the search costs of query image features for searching the 3D model, can be computed using the random forest classifier and visual vocabulary. Later these probabilities and search costs could be combined using a prioritization function which minimizes the search costs while maximizing the matchability. Once N number of 2D-3D matches have been found, these matches could be used for camera pose estimation.

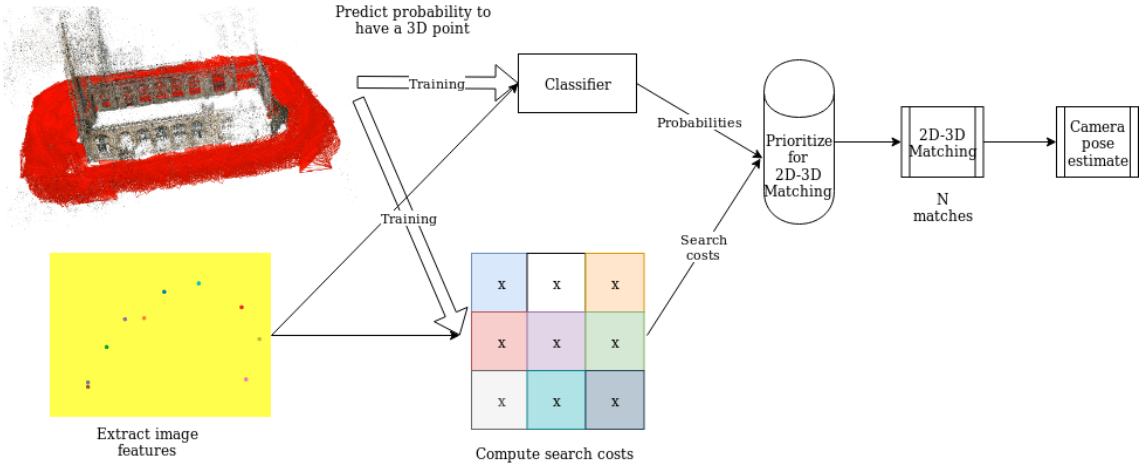


Figure 3.1: This image gives an overview of the method proposed in this thesis for visual localization. At first, a 3D model is reconstructed using a set of images of a location. The descriptor space of the image features used for 3D reconstruction is later clustered into visual words by training a clustering algorithm. A classifier is also trained to predict the probabilities of the query image features to have a 3D point. For the localization of a given query image, all the interesting images features in the query are extracted first. The search costs of the extracted features are calculated in the next stage based on the number of descriptors in the visual word which has the shortest Euclidean distance with the query image feature along with their probabilities to have a 3D point. In the next stage, the query image features are prioritized for 2D-3D feature matching based on their search costs and probabilities . Once N number of 2D-3D features have been found, the camera pose is estimated.

3.2 3D Reconstruction

For 3D structure based localization, it is necessary to develop the 3D model to determine the camera pose of the query image. The first step in the reconstruction of a 3D scene using structure for motion [28] is the extraction of all the interesting features (SIFT features [19]) from all the training images. Once all features are extracted, the images are paired with each other for feature matching based on the camera distances between the images and the rotation between the cameras of the images. With these image pairs, one could reconstruct the 3D structure using [3]. An example of the 3D reconstruction using SFM is shown in the figure 3.2.

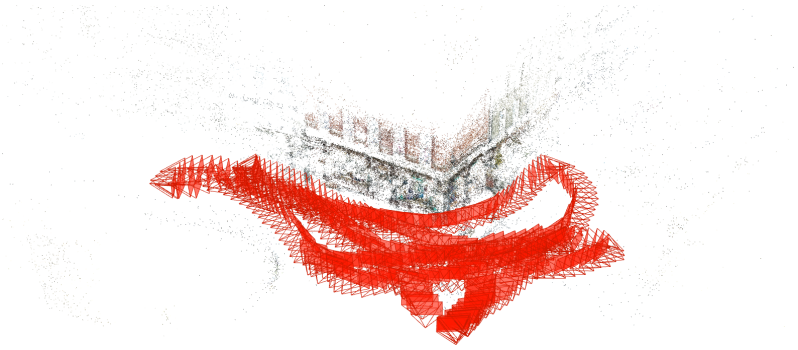


Figure 3.2: 3D Reconstruction of the Shop Facade Dataset [26], the red points represents represents the cameras.

3.3 Computing Probabilities

To determine the camera position of a query image feature, every feature in the 2D query image needs to be matched with all the 3D points. Matching all the 2D query image features with all the 3D points is time consuming. To overcome this bottleneck, [13] proposed to use 2D query image features which were predicted by a classifier to be matchable during 2D-2D feature matching. Similar to [13] a classifier is trained to predict if the 2D query image features would have a 3D point or not. From here on the 2D features which have a 3D point are called as positive class and the 2D features which do not have a 3D point is called a negative class. A random forest model is trained to classify the 2D features in the images those were used construct the 3D model and the 2D features that were not used for 3D model construction. The classifier works based on a probability bases and the class which has the highest probability is predicted as the output of the classifier. From the classifier, I obtained the probabilities of all the features to belong to the positive class for the prioritization function.

3.3.1 Computing Search Costs

[14] clustered the descriptor space containing all the 2D image descriptors used for 3D reconstruction into clusters called visual words. The search cost of a 2D query image feature is the total number of 2D feature descriptors of 3D points that are in the visual word which has the shortest Euclidean distance with 2D query image feature [14]. A K-means clustering algorithm [25] with the number of clusters equal to the number of visual words, is trained to cluster all the 2D descriptors used for 3D reconstruction and the search costs for all the 2D query image features were computed as proposed by [14].

3.4 Prioritization Function

To improve the time taken for 2D-3D feature matching, [13] proposed to use 2D query image features which were classified to have a 3D point for feature matching. [14] proposed to prioritize the 2D images features based on the search costs for searching with a 3D point. In this thesis I propose to combine the probabilities of all the 2D images features to have a 3D point along with the cost for searching the 2D query image feature. By this method, one could minimize the search cost for searching in the 3D model under the constraint of maximizing the probability of 3D point match I propose three different methods for prioritizing the query image features to reduce the time taken for 2D-3D feature matching. As explained in the theory section, the 0/1 knapsack problem is similar to the our problem of developing a prioritization function where the weights are the search costs and the probabilities are the values. But the problem is the time complexity to find the best solution. To overcome this problem, I have adopted three approximation schemes of the knapsack problem to develop a prioritization function for the query image features.

3.4.1 Greedy Approximation Method

This method is an approximation scheme for the knapsack proposed by [59] where the items for the knapsack problem are sorted in a descending order based on their value per weight $\frac{v_i}{w_i}$, here w_i represents the weights and v_i represents the values. This method reduces the computational complexity of the knapsack problem drastically as to $\mathcal{O}(n \log n)$ [61]. The top N items that satisfy the weight constraints is the result for this method of prioritizing the items in a knapsack problem. For our problem of prioritizing the query image features, the search costs are the considered to be the weights and the probabilities are the values. For a query image of i features with $v_1....v_i$ probabilities and $w_1....w_i$ search costs, the prioritization function as be expressed as:

$$G_i = \frac{v_i}{w_i} \tag{3.1}$$

$$G_1 > G_2 > G_3.... > G_i \tag{3.2}$$

The top N image features that satisfies the weight constraint can be later used for feature matching and camera pose estimation.

3.4.2 Average Ranking Method

In this method for approximating a knapsack problem, items are ranked independently based on the weights in an ascending order and values in an descending order. Later the items are prioritized based on the average rank between the weights and the values. Since the items are sorted three times,this method also has a computational complexity of $\mathcal{O}(n \log n)$. The top N items that satisfy the weight constraint is the result for this method of prioritizing the items in a knapsack problem. For our problem of prioritizing the query image feature, the search costs are the considered to be the weights and are arranged in an ascending order. And the probabilities are the values which is arranged in an descending order. The average rank between the

search costs and the probabilities is used to prioritize the query image features. For a query image of i features with $v_1....v_i$ probabilities and $w_1....w_i$ search costs, the prioritization function as be expressed as:

$$v_1 > v_2 > v_3.... > v_i \tag{3.3}$$

$$w_1 < w_2 < w_3.... < w_i \tag{3.4}$$

$$R_i = \frac{w_i + r_i}{2} \tag{3.5}$$

$$R_1 < R_2 < R_3....R_i \tag{3.6}$$

The top N image features that satisfies the weight constraint can be later used for feature matching and camera pose estimation.

3.4.2.1 Fully Polynomial Time Approximation Scheme

An alternative method to overcome the time complexity to solve the Knapsack problem is by using a Fully Polynomial Time Approximation Scheme. [63] proposes to scale profit values v_i by a polynomial bounded in $(\frac{1}{\epsilon})$ and n, where ϵ is the scaling factor. By this method the time complexity of the FTPAS algorithm to solve the knapsack problem could be reduced to polynomial time with respect to n and $\frac{1}{\epsilon}$. Motivated by this method, I have adopted a variation of this method, by dividing the search cost constraint by a factor of epsilon and the search of all the items were approximated by $\lfloor \frac{w_i}{\epsilon} \rfloor$. The time complexity is thus reduced to $\mathcal{O}(n^2 \lfloor \frac{n}{K} \rfloor)$. Here K is $\frac{n\epsilon}{P}$ where P is the maximum search cost. The approximation factor determines the degree of correctness of the solution found using the FPTAS algorithm when compared to the best solution for the knapsack problem. The top N prioritized query image features using the prioritization function could be later used for 2D-3D feature matching and once N matches are found, they can used for camera pose estimation.

4
Results

In this section, a detailed description on the experimental set up, evaluation methods and quality of the results obtained from the random forest classifier and the prioritization function is given.

4.1 Experimental Setup

The images from the Cambridge landmark dataset [26] was used to evaluate the performance of the proposed method. The 3D model was reconstructed using the library colmap `https://github.com/colmap/colmap` [3]. But for the 3D reconstruction of the scene, the images in the dataset have to be matched. In the next section, I explain criterias that were used to match the images for 3D reconstructions.

4.1.1 Image Matching for SFM

The images were paired for feature matching based on the distances between the camera positions of the images and rotational angle between the cameras of the images. The distances between the cameras were measured in the 3D scene coordinate system of the Cambridge Landmark dataset (which is in meters). The maximum distance between the images used for feature matching was 10 meters and the maximum angle of rotation between cameras of the images was set to 30 degrees.

4.1.2 Training Dataset

To develop the training dataset, the training images were used to reconstruct a 3D model. For the training the random forest, all 2D image descriptors that were used for 3D reconstruction were labeled as positive class and the remaining 2D image descriptors of the training images which were not used for 3D reconstruction were labeled to the negative class. The number positive and negative classes used for training the classifier were balanced by randomly adding positive and negative samples. A random set of 100000 descriptors from the positive class were used for training the K-means clustering algorithm.

4.1.3 Testing Dataset

To test the results of the random forest classifier and the prioritization functions, all the training and the testing images were used to reconstruct a 3D model. For

testing the random forest classifier, all the descriptors in the test images which had a 3D point that was triangulated by using two or more image features in the training dataset were classified as positive class. All the remaining 2D test image descriptors were labelled as class negative. The testing dataset for the prioritization function consisted of all the descriptors from the test images.

4.2 Evaluation Methods

In this section, I discuss about the various evaluation methods used to determine the quality of the results from the classifier and the prioritization functions.

4.2.1 Quality of the Random Forest Predictions

We evaluate only the probabilities of the positive class since we are only interested in the probabilities of the 2D features being in the 3D model (positive case). The probabilities of the positive class is evaluated based the number of positive and negative matches found. A plot is made between the number of positive and negative matches found based on the probabilities of the positive class, is used to evaluate the classifier.

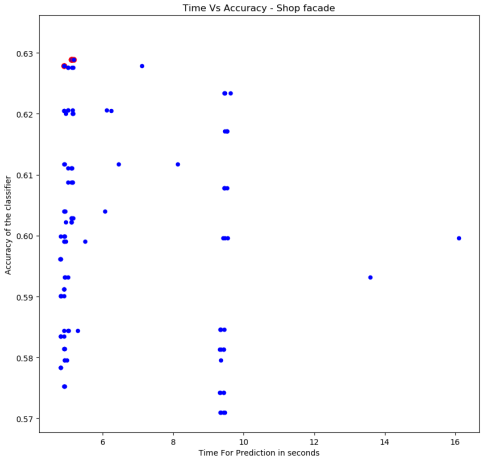
4.2.2 Quality of the Prioritization Function

The idea of this thesis is to minimize the time taken for 2D image feature matching with the features associated with 3D points while maximizing the number of 2D-3D matches for camera pose estimation in a 3D structure based localization. The prioritization function was evaluated based on the time take for matching (Search costs) and the number of 2D-3D feature matches found.
Using the 3D model reconstruction with all the testing and training images, a set of 2D-3D matches could be found. The 2D features in the test images that have a corresponding 3D point is the ground truth i.e., the features that have a corresponding 3D point. The prioritization functions were evaluated based on the number of ground truth features available in the prioritized list. This method of evaluation shows how many prioritized 2D image features would be matched with a 3D point. Another method to evaluate the prioritization function is by computing the search costs for only the top N prioritized features in every image. This method gives an overview on how much would be the overall search cost for a image when only the top N features were considered for matching.

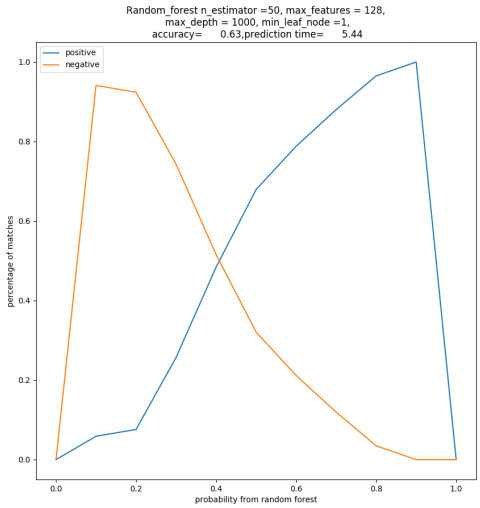
4.3 Classifier Results

I trained the different random forest classifiers by varying the number of trees used to grow the random forest model, the maximum depth of the forest trees and the minimum number of leaf nodes or final nodes in each tree. The number of trees parameter in the random forest was varied between 100, 500, 1000 and 1500 for all the datasets used for evaluation in this thesis except for the Shop Facade dataset.

For the Shop Facade dataset, the number of trees was varied between 10, 50 ,100 and 500. The maximum depth of the random forest was varied between 10, 100, 300 and 1000 and the the minimum number of leaf nodes in the random forest was varied between 1, 3, 10 and 20, for all the datasets used for evaluation in this thesis.

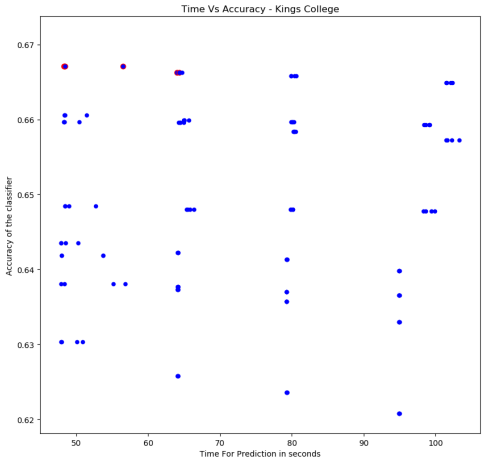


(a) Time vs Accuracy of Different Classifiers for the Shop facade dataset.

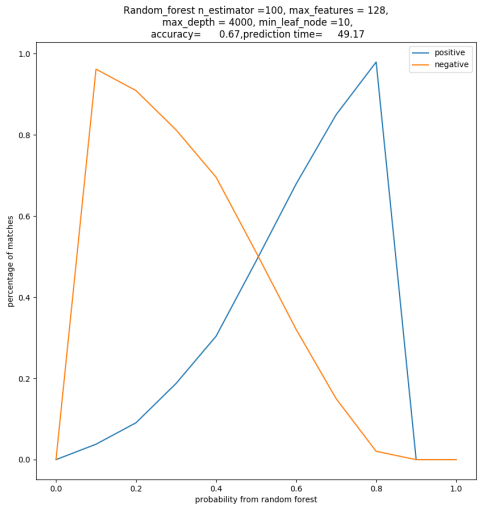


(b) Probability vs Percentage of matches for a classifier with N = 50, maximum depth = 1000 and minimum number of leaves = 1, Number of features = 128.

Figure 4.1: 4.1b shows the classification results for the positive and negative descriptors of the best classifier for the Shop Facade dataset.

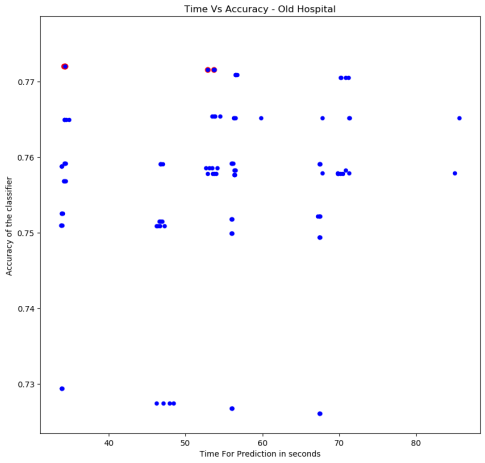


(a) Time vs Accuracy of Different Classifiers for the Kings College dataset.

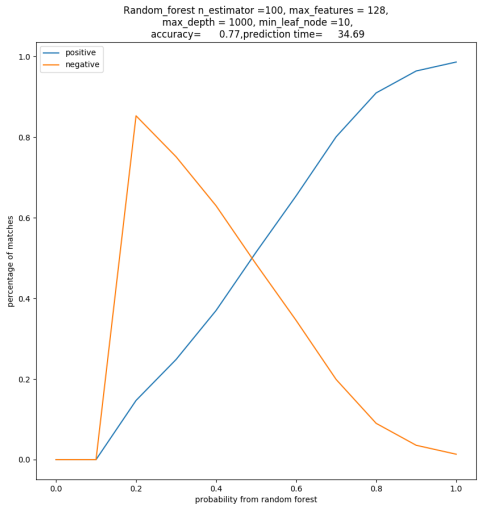


(b) Probability vs Percentage of matches for a classifier with N = 100, maximum depth = 4000 and minimum number of leaves = 10, Number of features = 128.

Figure 4.2: 4.2b shows the classification results for the positive and negative descriptors of the best classifier for the Kings College dataset.

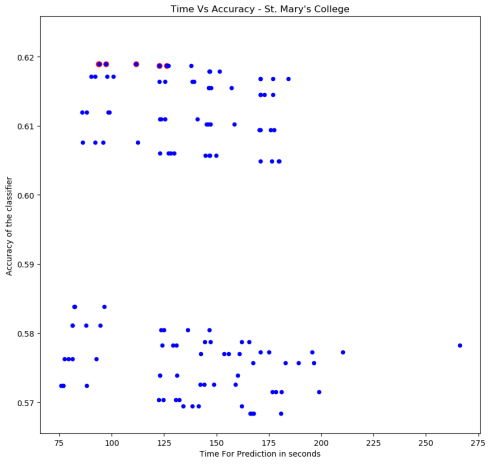


(a) Time vs Accuracy of Different Classifiers for the Old Hospital dataset.

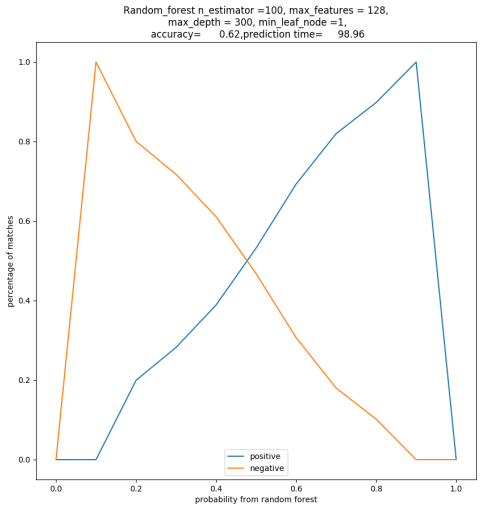


(b) Probability vs Percentage of matches for a classifier with N = 100, maximum depth = 1000 and minimum number of leaves = 10, Number of features = 128.

Figure 4.3: 4.3b shows the classification results for the positive and negative descriptors of the best classifier for the Old Hospital dataset.

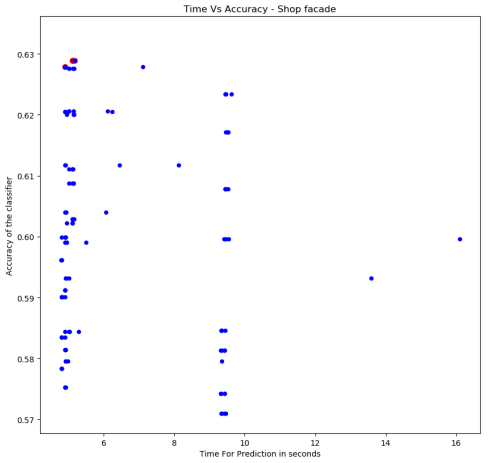


(a) Time vs Accuracy of Different Classifiers for St. Mary's College dataset.

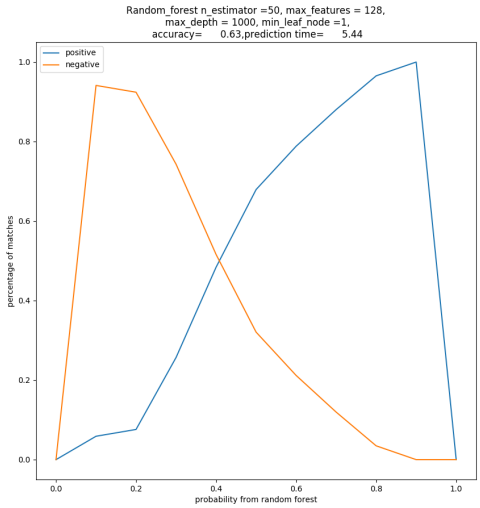


(b) Probability vs Percentage of matches for a classifier with N = 100, maximum depth = 300 and minimum number of leaves = 1, Number of features = 128.

Figure 4.4: 4.4b shows the classification results for the positive and negative descriptors of the best classifier for St. Mary's College dataset.



(a) Time vs Accuracy of Different Classifiers for the Trinity Great Court dataset.



(b) Probability vs Percentage of matches for a classifier with N = 100, maximum depth = 10000 and minimum number of leaves =1 0, Number of features = 128.

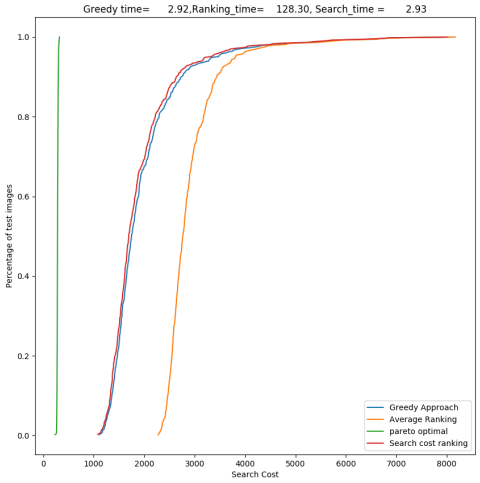
Figure 4.5: 4.5b shows the classification results for the positive and negative descriptors of the best classifier for the Trinity Great Court dataset.

4.4 Prioritization Function Results

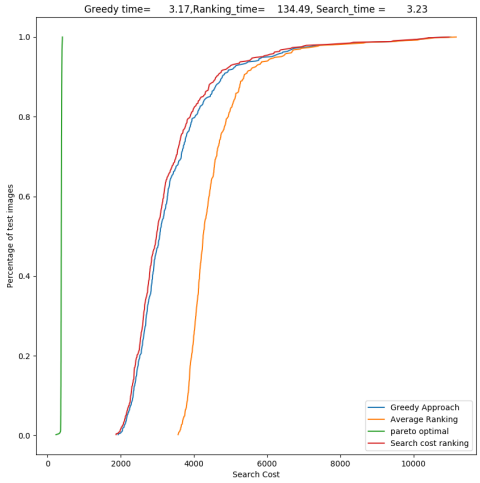
In this section, I show the resulting plots of the prioritization function. The prioritization functions were evaluated based on two methods. The first method was using the resulting search costs when only the top N prioritized features were used to compute the search costs. The second method was evaluated by using the number of descriptors that had a 3D corresponding point in the prioritized features, when the search costs was fixed.

4.4.1 Performance with Top N Fixed Features

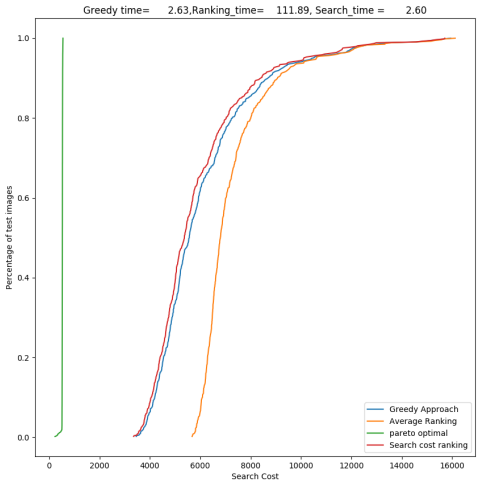
In this section, the plots (figures 4.6, 4.7, 4.8, 4.9 and 4.10) represents the cumulative frequency of the computed search costs in different images when the number of prioritized features was restricted to N. In the frequency plots (figures 4.6, 4.7, 4.8, 4.9 and 4.10) the legend *pareto optimal* represents the optimal solution of the prioritization function, the legend *Average Ranking* represents the average ranking method for prioritizing the features (section 3.4.2), the legend *Search cost ranking* represents the search cost based prioritization function [14] and the legend *Greedy Approach* represents the prioritization function using the greedy approximation method (section 3.4.1).



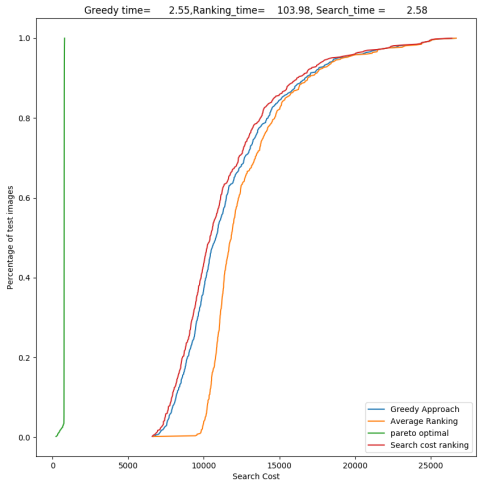
(a) Number of prioritized features = 500.



(b) Number of prioritized features = 700.

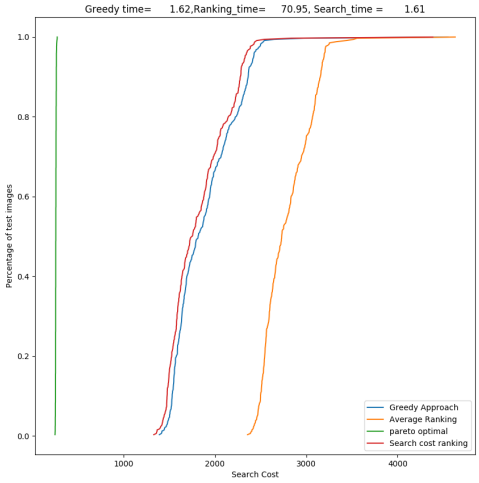


(c) Number of prioritized features = 1000.

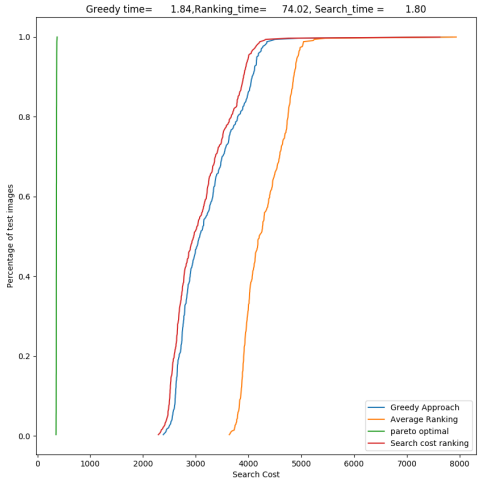


(d) Number of prioritized features = 1500.

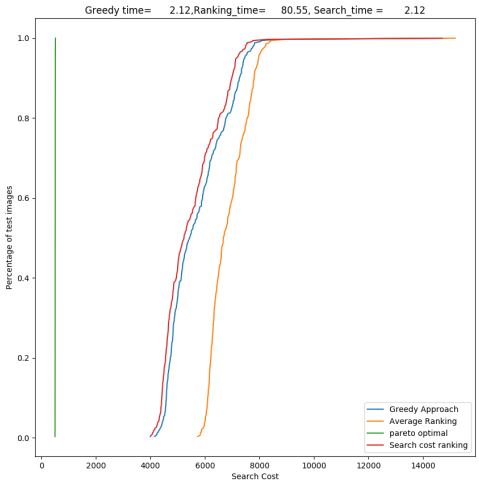
Figure 4.6: Cumulative histogram of Search costs over the percentage of images for the Greedy, Average Ranking prioritization function along with the optimal solution when the number of descriptors is fixed - Shop Facade dataset.



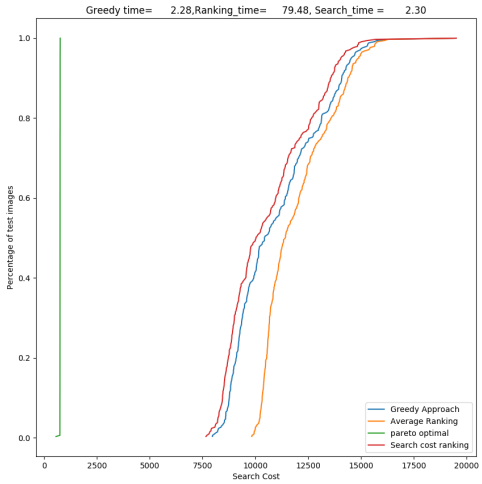
(a) Number of prioritized features = 500.



(b) Number of prioritized features = 700.

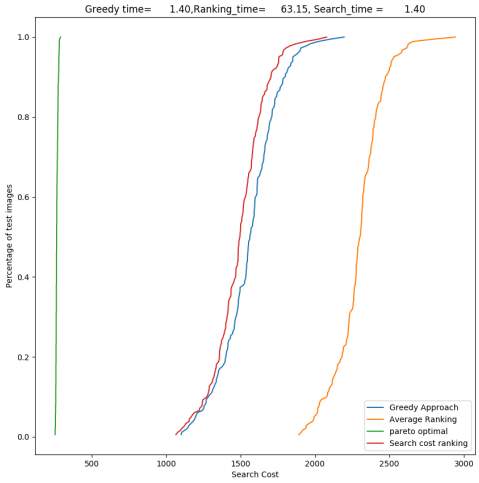


(c) Number of prioritized features = 1000.

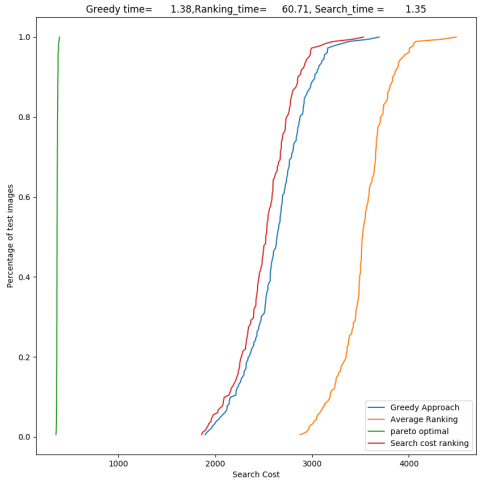


(d) Number of prioritized features = 1500.

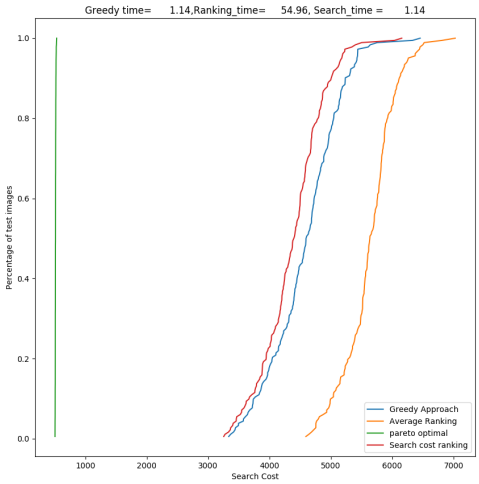
Figure 4.7: Cumulative histogram of Search costs over the percentage of images for the Greedy, Average Ranking prioritization function along with the optimal solution when the number of descriptors is fixed - Kings College dataset.



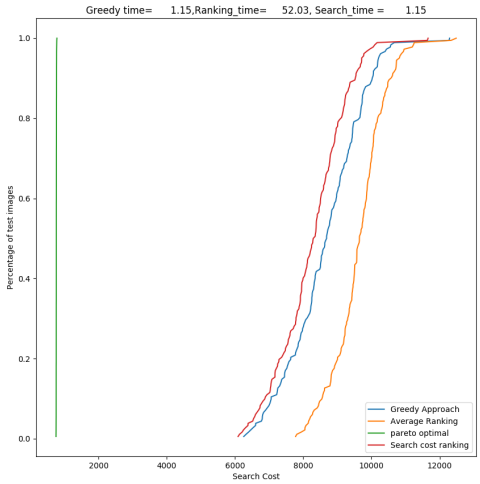
(a) Number of prioritized features = 500.



(b) Number of prioritized features = 700.

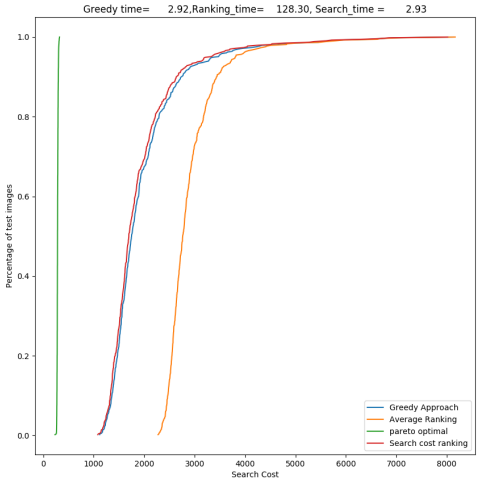


(c) Number of prioritized features = 1000.

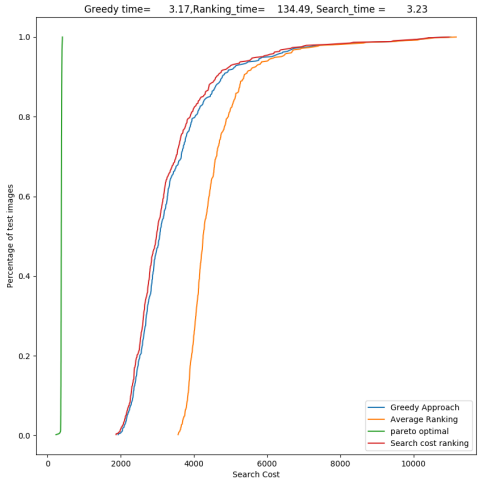


(d) Number of prioritized features = 1500.

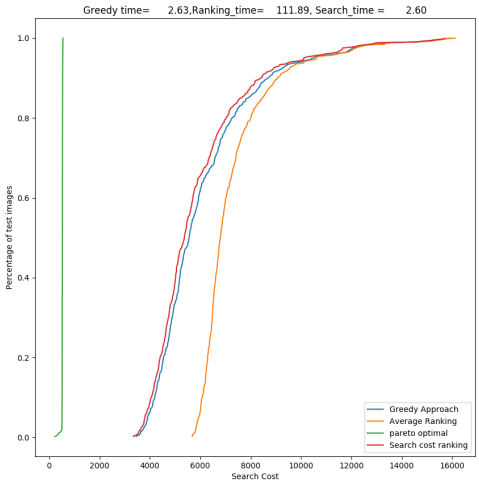
Figure 4.8: Cumulative histogram of Search costs over the percentage of images for the Greedy, Average Ranking prioritization function along with the optimal solution when the number of descriptors is fixed - Old Hospital dataset.



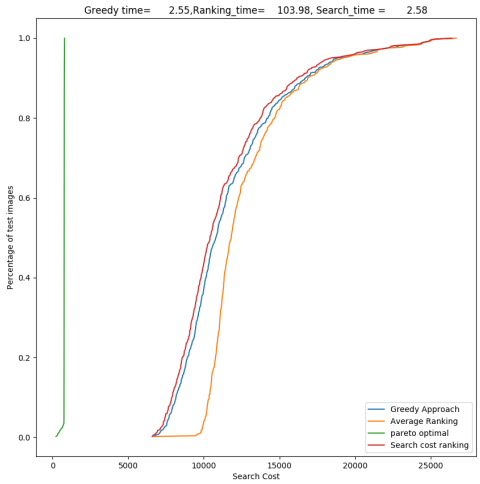
(a) Number of prioritized features = 500.



(b) Number of prioritized features = 700.

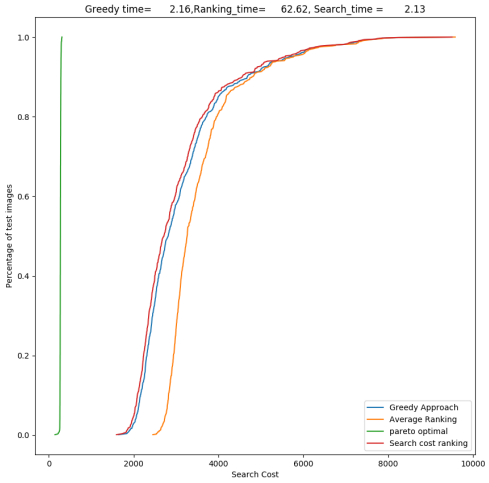


(c) Number of prioritized features = 1000.

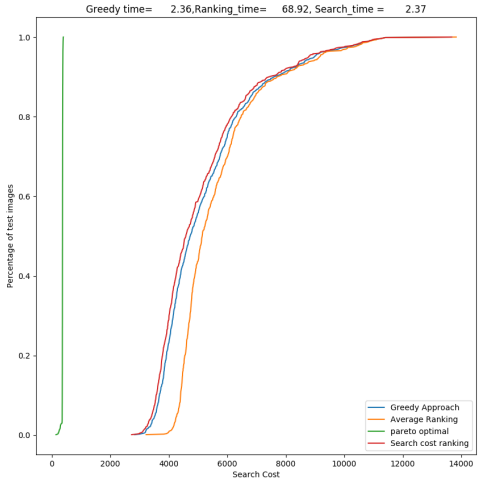


(d) Number of prioritized features = 1500.

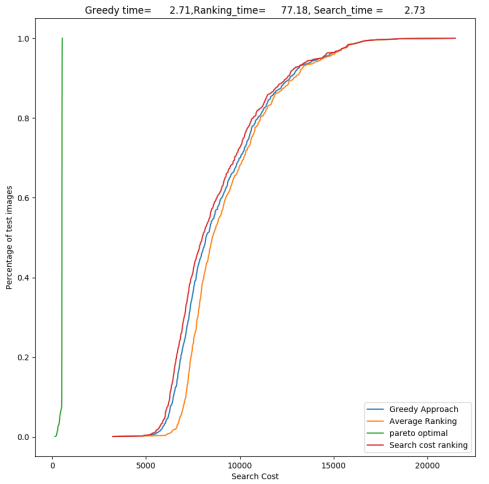
Figure 4.9: Cumulative histogram of Search costs over the percentage of images for the Greedy, Average Ranking prioritization function along with the optimal solution when the number of descriptors is fixed - St. Mary's College dataset.



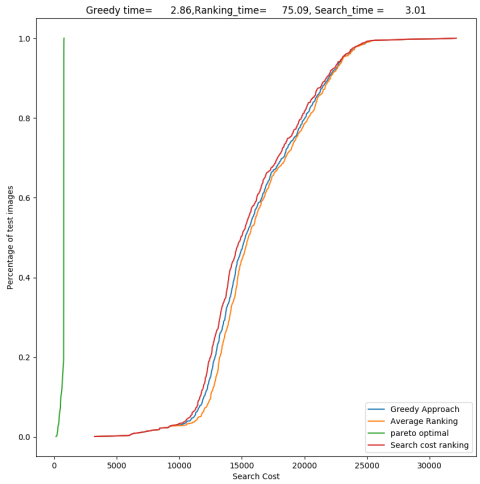
(a) Number of prioritized features = 500.



(b) Number of prioritized features = 700.



(c) Number of prioritized features = 1000.



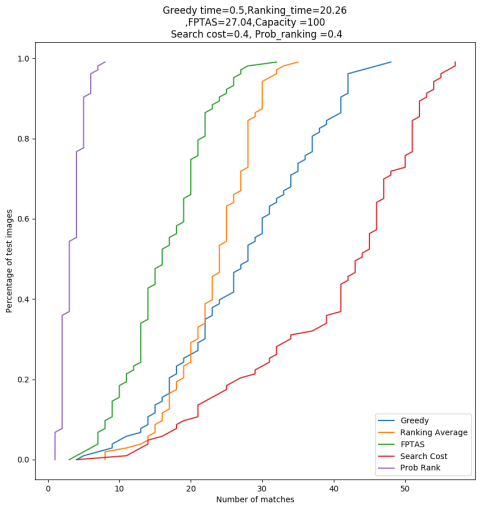
(d) Number of prioritized features = 1500.

Figure 4.10: Cumulative histogram of Search costs over the percentage of images for the Greedy, Average Ranking prioritization function along with the optimal solution when the number of descriptors is fixed - Trinity Great Court dataset.

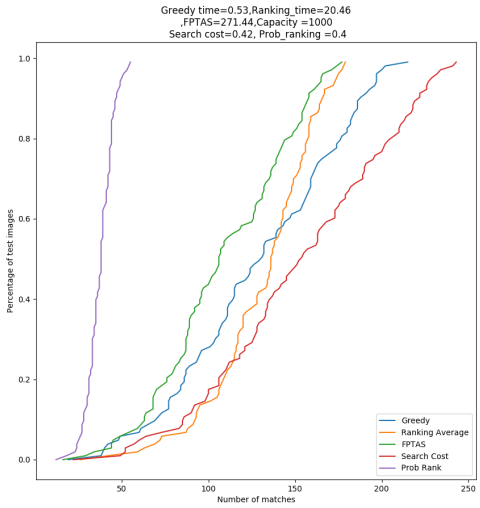
4.4.2 Performance on Fixed Search Costs

The top N features that satisfy the search cost constraint were used for the evaluation of the prioritization function. The plots in the figures 4.15, 4.13, 4.14, 4.12 and 4.11 in this section represent the cumulative distribution over the number of correct 2D-3D feature matches in a test image (section 4.2.2) against the percentage of test images. The scaling factor ϵ for the prioritization function based on FPTAS (section 3.4.3) knapsack was set to 2. In the figures 4.15, 4.13, 4.14, 4.12 and 4.11 the legend

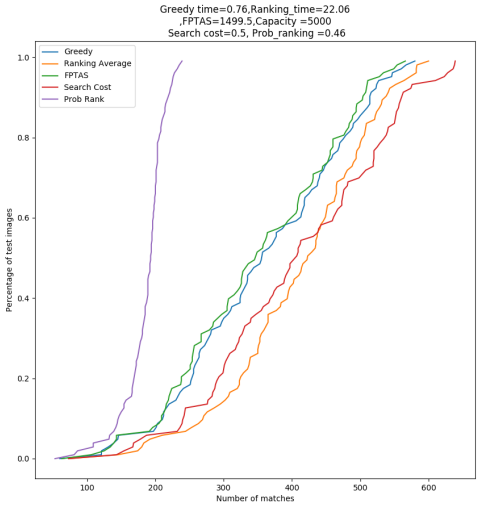
Prob Rank refers to a prioritization function which ranks the 2D features of the test image based on their probabilities to have a 3D point in a descending order. The legend *Ranking Average* refers to the average ranking prioritization function (section 3.4.2), the legend *FPTAS* refers to the fully polynomial time approximation scheme prioritization function (section 3.4.3), the legend *Greedy* refers to the greedy approximation method for prioritizing the features (section 3.4.1) and the legend *Search Cost* refers to the prioritization based on search costs [14].



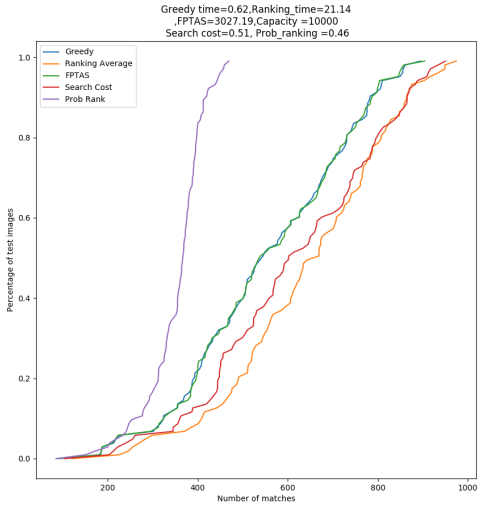
(a) Search cost limit =100.



(b) Search cost limit =1000.

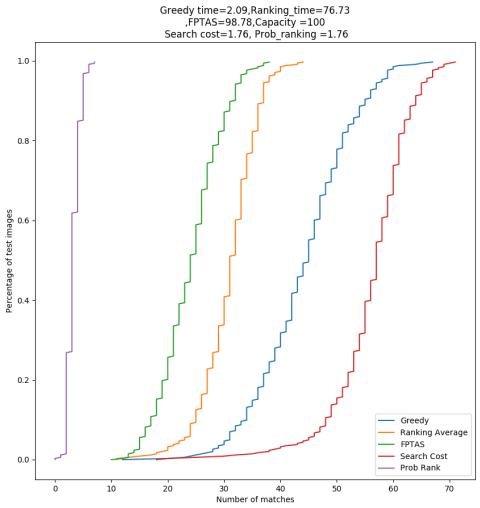


(c) Search cost limit =5000.

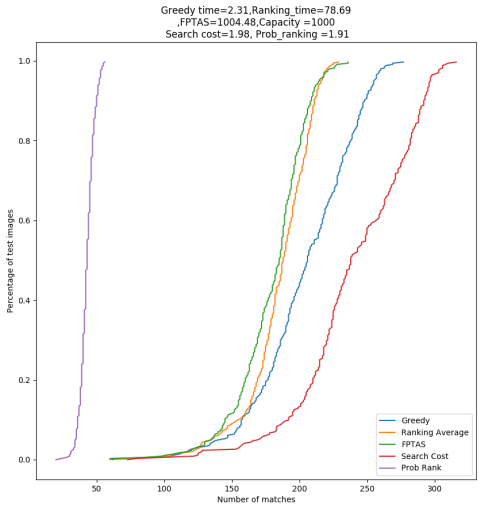


(d) Search cost limit =10000.

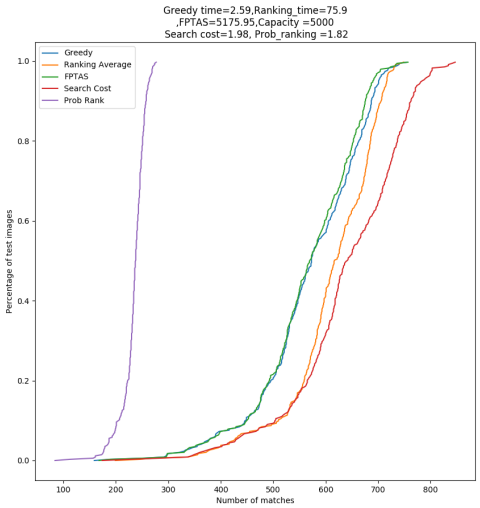
Figure 4.11: Performance of all the prioritization functions on the Shop Facade Dataset with fixed search costs



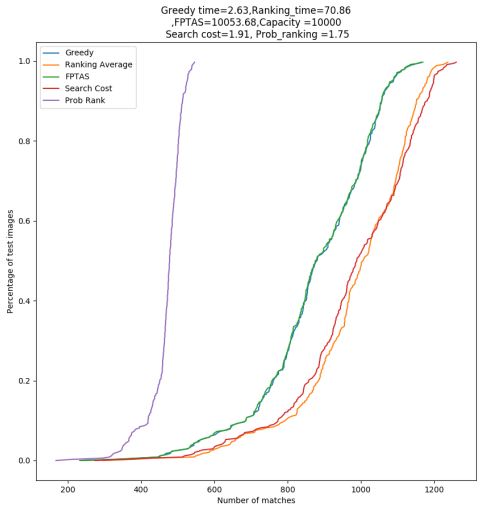
(a) Search cost limit =100.



(b) Search cost limit =1000.

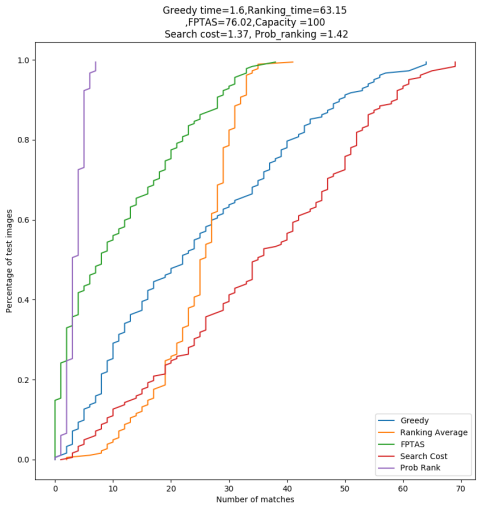


(c) Search cost limit =5000.

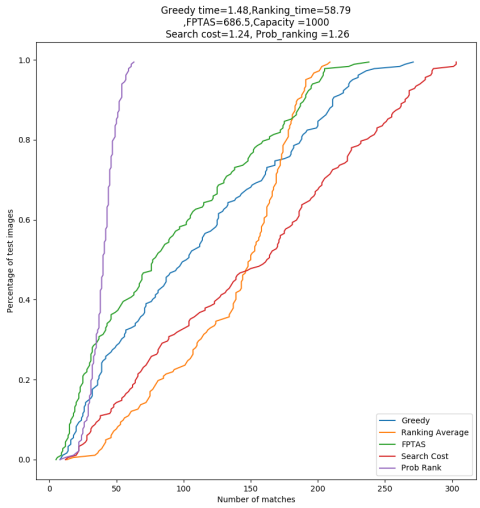


(d) Search cost limit =10000.

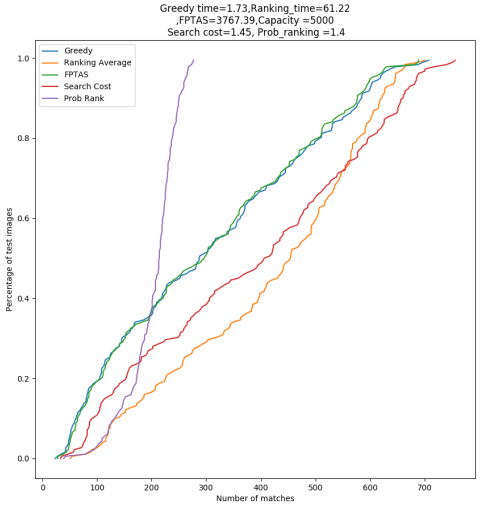
Figure 4.12: Performance of all the prioritization functions on the Kings College Dataset with fixed search costs.



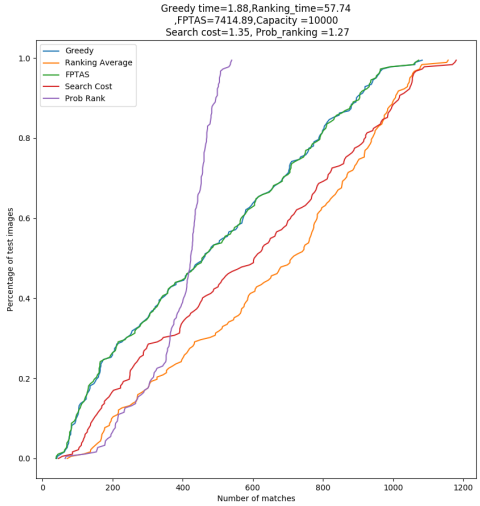
(a) Search cost limit =100.



(b) Search cost limit =1000.

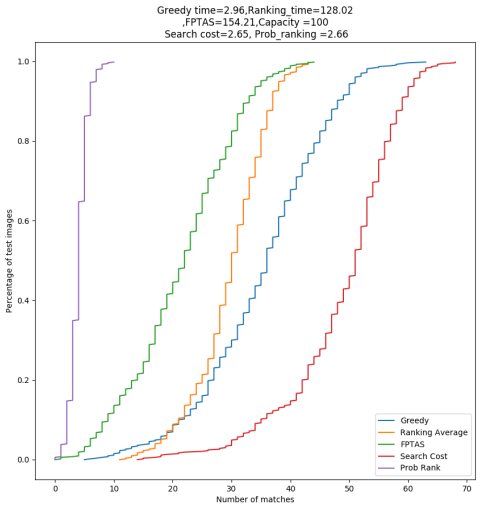


(c) Search cost limit =5000.

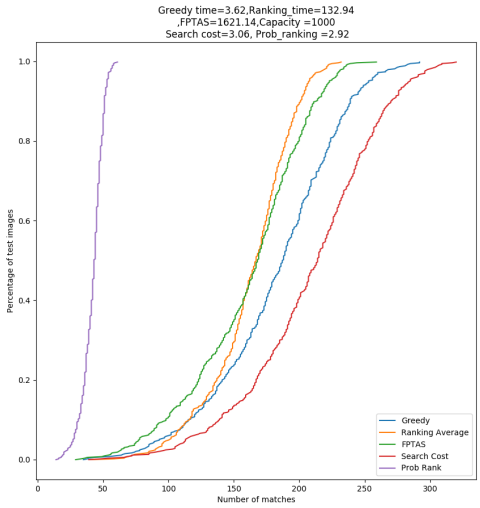


(d) Search cost limit =10000.

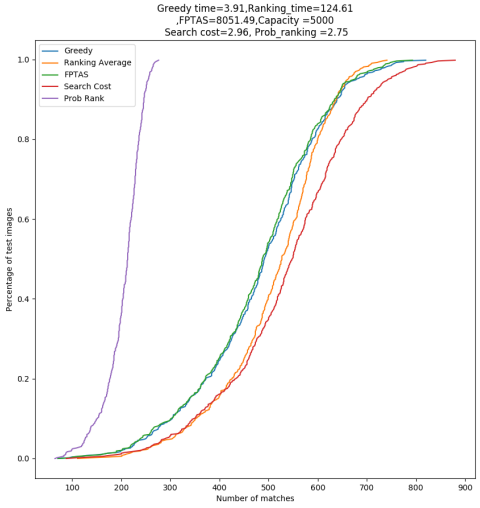
Figure 4.13: Performance of all the prioritization functions on the Old Hospital Dataset with fixed search costs.



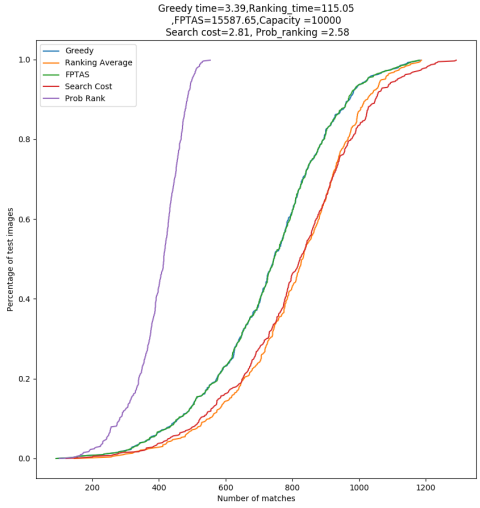
(a) Search cost limit =100.



(b) Search cost limit =1000.

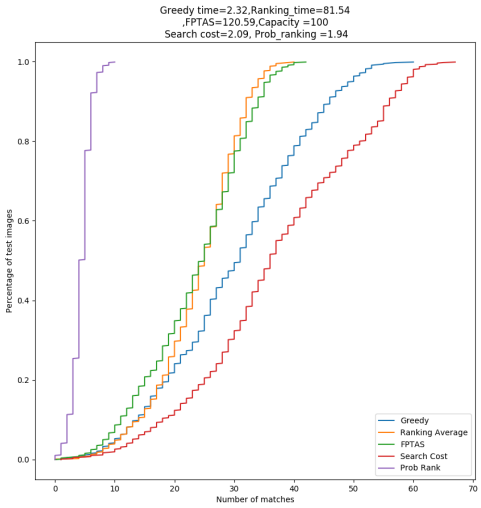


(c) Search cost limit =5000.

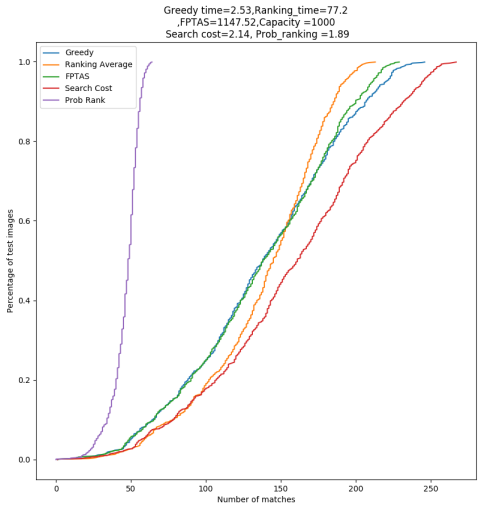


(d) Search cost limit =10000.

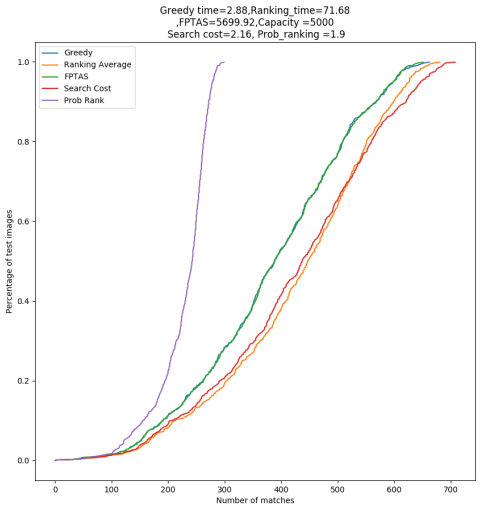
Figure 4.14: Performance of all the prioritization functions on the St. Mary's College Dataset with fixed search costs.



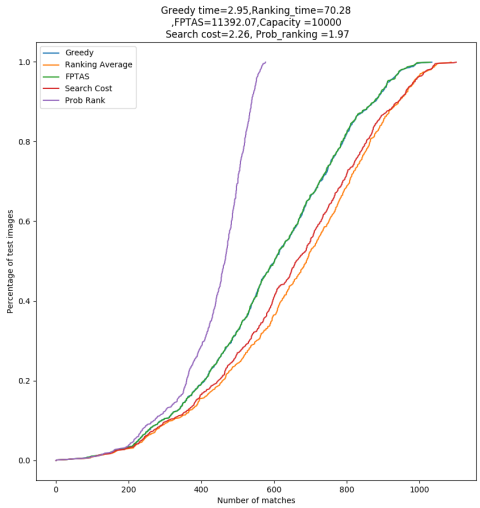
(a) Search cost limit =100.



(b) Search cost limit =1000.



(c) Search cost limit =5000.



(d) Search cost limit =10000.

Figure 4.15: Performance of all the prioritization functions on the Trinity Great Court Dataset with fixed search costs.

4.4.3 Discussions

In this section a detailed discussion on the results of the classifier and the prioritization functions is given.

4.4.3.1 Prediction Accuracy

The best classifier was chosen based on the time taken for making the prediction and the accuracy with the test dataset. The classifier which had the highest accuracy

but with the lowest time to classify the test image descriptors was chosen as the best classifier. The figures 4.1b, 4.4b, 4.2b, 4.5b show that the classifier does not correctly predict any descriptor with 100 % accuracy with a probability of 1 except for the classifier for the Old Hospital dataset (figure 4.3). When the probabilities of all the features were viewed, the classifier had 0 2D image features which had 1 or 0 probability. An ideal classifier will have a probability of 1 for all the positive descriptors and a probability of 0 for all the negative descriptors. But this is not the case in neither of the best classifiers for the datasets used in this thesis. Though the results of the classifiers are not optimal, the classifiers are still able to predict with an accuracy greater than 60% which would be useful for prioritizing the features.

4.4.3.2 Prioritization function

From the figures 4.6, 4.7, 4.8, 4.9 and 4.10, it could be seen that the search cost based prioritization function has the lowest search cost for the top N prioritized features, which is expected as the points are just prioritized using the search costs. But the reason for the greedy prioritization function (section 3.4.1) to have a lower search cost when compared to the average ranking method (section 3.1.2) is quite unclear.

From the figures 4.15, 4.13, 4.14, 4.12 and 4.11, it could be seen that number of matches found by the fully polynomial time approximation scheme based prioritization function (section 3.4.3) is low. This because of the scaling factor (ϵ) used to reduce the time taken for prioritizing. As the limit on the search cost increases the significance of ϵ (scaling factor) reduces, therefore the time taken to prioritize the features increases. The limited performance of all the proposed prioritization under limited search costs when compared with the search cost based prioritization function may be due to the accuracy of the classifier. That is, the classifier might suggest points with a greater probability but these point may not have a 3D point. This would make the prioritization functions to prioritize these miss-classified points at the top of the prioritization list. Therefore, a certain amount of time is being spent for searching for corresponding 3D points for image points which might not have a 3D point. It could be also seen from the figures 4.15, 4.13, 4.14, 4.12 and 4.11, the average ranking method (section 3.4.2) continues to improve with respect to the search cost based prioritization function. As the limit on the search cost increases, the average ranking method (section 3.4.2) outperforms the search cost based method in the Shop Facade dataset (figure 4.11d). In the other datasets when the search costs limit is maximum, the performance of the average ranking method (section 3.4.2) is almost similar to the search cost based method which could be seen in the figures 4.12d, 4.14d, 4.13d, 4.15d. The reason behind such reduction in the performance of the search cost based prioritization function as the search cost increases when compared with the average ranking prioritization function (section 3.4.2), may due to the fact that the average ranking method (section 3.4.2) is being assisted by the classifier to prioritize the points. But the down-side of the average ranking method (section 3.4.2) for prioritization is the time taken for prioritization, since the query image features are sorted three times.

In practical application, one would have to choose between two constraints, either the time taken for prioritization function or the number of correct matches found using a prioritization function. If the time taken for the prioritization function is a constraint, it is better to use the search cost based prioritization function. But if maximizing the number of correct matches found is important than the time taken for prioritization, then average ranking prioritization function (section 3.4.2) would be the better approach.

5
Conclusion

In this thesis, I have developed different methods to prioritize the query 2D image features by using their probabilities to have a 3D point and their search costs, for 2D-3D feature matching in a 3D structure based visual localization. The probabilities of the image features to have a 3D point were predicted using a random forest classifier and the search costs of image features was the number of 3D point features in the visual word with which the image feature had shortest euclidean distance with. Through extensive evaluation, I show that the proposed average ranking based prioritization (section 3.4.2) performs on-par with the search cost based prioritization technique proposed by [14], when there is a lower constraint on the time taken for prioritization.

5.1 Future Work

The efficiency of the proposed prioritization functions should be evaluated in a 3D structure based localization pipeline and also compared with other state-of-the-art techniques like [4].
The random forest classifier used to predict the probabilities was not able completely to learn the patterns of the 2D image features that have a 3D point and the 2D image features that do not have a 3D point. An alternative approach would be to use a neural network [58] for predicting the probabilities of the 2D image features that have a 3D point.
The efficiency of the proposed prioritization functions could be improved by using a 3D structure based localization pipeline which actively prioritizes the 3D points along with the 2D input image features based on the visual appearance once a matching 3D point is found [4].

Bibliography

[1] Frisk, D. (2016) A Chalmers University of Technology Master's thesis template for LaTeX. Unpublished.

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

[3] Schonberger, Johannes L., and Jan-Michael Frahm. "Structure-from-motion revisited." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[4] Sattler, Torsten, Bastian Leibe, and Leif Kobbelt. "Efficient & effective prioritized matching for large-scale image-based localization." IEEE transactions on pattern analysis and machine intelligence 39.9 (2016): 1744-1756.

[5] Liu, Liu, Hongdong Li, and Yuchao Dai. "Efficient global 2d-3d matching for camera localization in a large-scale 3d map." Proceedings of the IEEE International Conference on Computer Vision. 2017.

[6] Zeisl, Bernhard, Torsten Sattler, and Marc Pollefeys. "Camera pose voting for large-scale image-based localization." Proceedings of the IEEE International Conference on Computer Vision. 2015.

[7] Svärm Linus, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. "City-scale localization for cameras with known vertical direction." IEEE transactions on pattern analysis and machine intelligence 39.7 (2016): 1455-1461.

[8] Arandjelovic Relja, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. "NetVLAD: CNN architecture for weakly supervised place recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[9] Lowry Stephanie, Niko Sünderhauf, Paul Newman, John J. Leonard, David Cox, Peter Corke, and Michael J. Milford. "Visual place recognition: A survey." IEEE Transactions on Robotics 32.1 (2015): 1-19.

[10] Torsten Sattler, Michal Havlena, Konrad Schindler, and Marc Pollefeys. "Large-scale location recognition and the geometric burstiness problem." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[11] Torii Akihiko, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. "24/7 place recognition by view synthesis." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[12] Kendall, Alex, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization." Proceedings of the IEEE international conference on computer vision. 2015.

[13] Hartmann, Wilfried, Michal Havlena, and Konrad Schindler. "Predicting matchability." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014.

[14] Sattler, Torsten, Bastian Leibe, and Leif Kobbelt. "Fast image-based localization using direct 2d-to-3d matching." 2011 International Conference on Computer Vision. IEEE, 2011.

[15] McGlone, C., E. Mikhail, and J. Bethel. "Manual of photogrammetry: American society for photogrammetry and remote sensing." Bethesda, MD (2004).

[16] Tuytelaars, Tinne, and Krystian Mikolajczyk. "Local invariant feature detectors: a survey." Foundations and trends® in computer graphics and vision 3.3 (2008): 177-280.

[17] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." European conference on computer vision. Springer, Berlin, Heidelberg, 2006.

[18] Brown, Matthew, Gang Hua, and Simon Winder. "Discriminative learning of local image descriptors." IEEE transactions on pattern analysis and machine intelligence 33.1 (2010): 43-57.

[19] Lowe, David G. "Distinctive image features from scale-invariant keypoints." International journal of computer vision 60.2 (2004): 91-110.

[20] Fischler, M. A., and R. Bolles. "Random sample consensus. a paradigm for model fitting with apphcahons to image analysm and automated cartography." Proc. 1980 Image Understandtng Workshop (College Park, Md., Apr i980) LS Baurnann, Ed, Scmnce Apphcatlons, McLean, Va. 1980.

[21] Rokach, Lior, and Oded Z. Maimon. Data mining with decision trees: theory and applications. Vol. 69. World scientific, 2008.

[22] Rokach, Lior, and Oded Maimon. "Top-down induction of decision trees classifiers-a survey." IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 35.4 (2005): 476-487.

[23] Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

[24] Kam, Ho Tin. "Random decision forest." Proceedings of the 3rd International Conference on Document Analysis and Recognition. Vol. 1416. 1995.

[25] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) 28.1 (1979): 100-108.

[26] Kendall, Alex, Matthew Grimes, and Roberto Cipolla. "Posenet: A convolutional network for real-time 6-dof camera relocalization." Proceedings of the IEEE international conference on computer vision. 2015.

[27] Breiman, Leo. "Bagging predictors." Machine learning 24.2 (1996): 123-140.

[28] Dellaert Frank, Steven M. Seitz, Charles E. Thorpe, and Sebastian Thrun. "Structure from motion without correspondence." Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662). Vol. 2. IEEE, 2000.

[29] Moré, Jorge J. "The Levenberg-Marquardt algorithm: implementation and theory." Numerical analysis. Springer, Berlin, Heidelberg, 1978. 105-116.

[30] Beder, Christian, and Richard Steffen. "Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence." Joint Pattern Recognition Symposium. Springer, Berlin, Heidelberg, 2006.

[31] Schonberger, Johannes L., and Jan-Michael Frahm. "Structure-from-motion revisited." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016.

[32] Hartley, Richard I., and Peter Sturm. "Triangulation." Computer vision and image understanding 68.2 (1997): 146-157.

[33] Li, Hongdong. "A practical algorithm for L triangulation with outliers." 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007.

[34] Shotton Jamie, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. "Scene coordinate regression forests for camera relocalization in RGB-D images."Scene coordinate regression forests for camera relocalization in RGB-D images." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013.

[35] Robertson, Duncan P., and Roberto Cipolla. "An Image-Based System for Urban Navigation." Bmvc. Vol. 19. No. 51. 2004.

[36] W. Zhang and J. Kosecka. Image based localization in urban environments. In3DPVT, 2006

[37] Schindler, Grant, Matthew Brown, and Richard Szeliski. "City-scale location recognition." 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007.

[38] Nister, David, and Henrik Stewenius. "Scalable recognition with a vocabulary tree." 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06). Vol. 2. Ieee, 2006.

[39] Avrithis, Yannis, Yannis Kalantidis, Giorgos Tolias, and Evaggelos Spyrou. "Retrieving landmark and non-landmark images from community photo collections." Proceedings of the 18th ACM international conference on Multimedia. ACM, 2010.

[40] Se, Stephen, David G. Lowe, and James J. Little. "Global localization using distinctive visual features." Iros. 2002.

[41] Irschara Arnold, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. "From structure-from-motion point clouds to fast location recognition." 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2009.

[42] Arth Clemens, Daniel Wagner, Manfred Klopschitz, Arnold Irschara, and Dieter Schmalstieg. "Wide area localization on mobile phones.Wide area localization on mobile phones." 2009 8th ieee international symposium on mixed and augmented reality. IEEE, 2009.

[43] Li, Yunpeng, Noah Snavely, and Daniel P. Huttenlocher. "Location recognition using prioritized feature matching." European conference on computer vision. Springer, Berlin, Heidelberg, 2010.

[44] Tran Ngoc-Trung, Dang-Khoa Le Tan, Anh-Dzung Doan, Thanh-Toan Do, Tuan-Anh Bui, Mengxuan Tan, and Ngai-Man Cheung. "On-device scalable image-based localization via prioritized cascade search and fast one-many ransac." IEEE Transactions on Image Processing 28.4 (2018): 1675-1690.

[45] Kellerer, Hans, and Ulrich Pferschy. "A new fully polynomial time approximation scheme for the knapsack problem." Journal of Combinatorial Optimization 3.1 (1999): 59-71.

[46] Cramer, Jan Salomon. "Predictive performance of the binary logit model in unbalanced samples." Journal of the Royal Statistical Society: Series D (The Statistician) 48.1 (1999): 85-94.

[47] Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." Neural processing letters 9.3 (1999): 293-300.

[48] Bresson Guillaume, Zayed Alsayed, Li Yu, and Sébastien Glaser."Simultaneous localization and mapping: A survey of current trends in autonomous driving." IEEE Transactions on Intelligent Vehicles 2.3 (2017): 194-220.

[49] Se, Stephen, David Lowe, and Jim Little. "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks." The international Journal of robotics Research 21.8 (2002): 735-758.

[50] Henderson, Steven J., and Steven Feiner. "Evaluating the benefits of augmented reality for task localization in maintenance of an armored personnel carrier turret." 2009 8th IEEE International Symposium on Mixed and Augmented Reality. IEEE, 2009.

[51] Milford, Michael J., and Gordon F. Wyeth. "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights." 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012.

[52] Schindler, Grant, Matthew Brown, and Richard Szeliski. "City-scale location recognition." 2007 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2007.

[53] Brachmann Eric, Alexander Krull, Sebastian Nowozin, Jamie Shotton, "DSAC-differentiable RANSAC for camera localization." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[54] Brachmann Eric, and Carsten Rother. "Learning less is more-6d camera localization via 3d surface regression." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018.

[55] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." Communications of the ACM 24.6 (1981): 381-395.

[56] Valentin Julien, Matthias Nießner, Jamie Shotton, Andrew Fitzgibbon, Shahram Izadi, and Philip HS Torr. "Exploiting uncertainty in regression forests for accurate camera relocalization." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015.

[57] Beis, Jeffrey S., and David G. Lowe. "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces." cvpr. Vol. 97. 1997.

[58] Mehlig, B. "Artificial Neural Networks." arXiv preprint arXiv:1901.05639 (2019).

[59] Dantzig, George B. "Discrete-variable extremum problems." Operations research 5.2 (1957): 266-288.

[60] Garey, Michael R., and David S. Johnson. Computers and intractability. Vol. 29. New York: wh freeman, 2002.

[61] Toth, Paolo, and Silvano Martello. Knapsack problems: Algorithms and computer implementations. Wiley, 1990.
[62] Pushpa, S. K., T. V. Mrunal, and C. Suhas. "‘A study of performance analysis on Knapsack problem." Int. J. Comput. Appl.
[63] The Knapsack Problem and Fully Polynomial TimeApproximation Schemes (FPTAS)
[64] Hristakeva, Maya, and Dipti Shrestha. "Different approaches to solve the 0/1 knapsack problem." The Midwest Instruction and Computing Symposium. 2005.