



The Interdisciplinary Center, Herzlia  
Efi Arazi School of Computer Science  
M.Sc. program

# SFM using CNISO

by  
**Shalev David**

Final project, submitted in partial fulfillment of the requirements  
for the M.Sc. degree, School of Computer Science  
The Interdisciplinary Center, Herzliya

January 2019

This work was carried out under the supervision of Prof. Yael Moses from the Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya.

# Acknowledgements

I would like to acknowledge and thank the following important people who have supported me during my research project.

Firstly, I would like to express my gratitude to my supervisor Prof. Yael Moses for her unwavering support, guidance and insights throughout this research project, and through my Master's degree. It was an honor working with Yael who encourages creativity and out of the box attitude combined with depth into details and seeing the overall picture.

I would also like to thank Lior Talker, for allowing access to his work with explanations and advisory when needed. His work was the basic grounds and the starting point for this research project.

Finally, I would like to thank my family for helping me focus on what has been a hugely rewarding and enriching process.

# Abstract

Structure From Motion (SFM) is the process of camera calibration and 3D structure reconstruction out of a set of images captured on a static scene. SFM systems are complicated and are usually composed of many layers and comprise computer vision algorithms such as feature extraction, feature matching, RANSAC, Bundle Adjustments, Triangulation and more.

Recently, SFM open sources offer opportunity to reconstruct scenes composed of image datasets, and a major challenge has become dealing with large scale datasets. For this purpose, a novel and efficient method for estimating the number of correct matches using only spatial order is suggested in order to leverage and enhance SFM pipelines. This method exploit the characteristic that the spatial order of feature matches should be preserved between a pair of images and is referred in the project as CNISO.

In this project we attempt to improve an open-source SFM system. This comes into practice by implementing CNISO algorithm and integrating it into an existing SFM pipeline for achieving better performance in terms of runtime and quality.

This project contributes a C++ implementation of CNISO algorithm available on github for the community. In addition, it provides deep insight of the emerging COLMAP's SFM open source, and offers some experimental improvements to COLMAP pipeline. Moreover, in context of COLMAP, this study discovers and reveals practical techniques for performing R&D by a limited resource individual or a novice developer.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Components of Structure for Motion . . . . .	4
1.1.1	Scene Graph . . . . .	4
1.1.2	3D Reconstruction and Camera Estimation . . . . .	5
1.1.3	RANSAC in SFM . . . . .	6
1.2	Outlier Removal using Features Spatial Order . . . . .	7
1.3	Brief Overview of this Project Report . . . . .	8
<b>2</b>	<b>Related Works</b>	<b>10</b>
2.1	Feature Matching . . . . .	10
2.2	Choose a Subset of Image Pairs . . . . .	11
<b>3</b>	<b>Leverage SFM using CNISO</b>	<b>14</b>
<b>4</b>	<b>Research &amp; Development</b>	<b>17</b>
4.1	Choosing SFM system . . . . .	17
4.2	Implementation of CNISO in C++ . . . . .	19
4.2.1	Details . . . . .	19
4.2.2	Unit Testing . . . . .	21
4.3	Colmap Build . . . . .	22
4.3.1	Conclusion and best practices for COLMAP Build . . . . .	25
4.4	COLMAP Pipeline Code & Dataflow . . . . .	26
4.5	Integrating CNISO into Colmap . . . . .	28
4.5.1	Remove pairs using CNISO . . . . .	28
4.5.2	RANSAC halting conditions using CNISO . . . . .	32
4.5.3	Combine pairs removal and RANSAC halting . . . . .	33

<b>5 Experiments</b>	<b>34</b>
5.1 Handling Feature Extraction issues in COLMAP . . . . .	35
5.2 Remove pairs using CNISO . . . . .	36
5.2.1 Vocab Tree geometric verification . . . . .	36
5.2.2 Exhaustive geometric verification . . . . .	39
5.3 RANSAC Halting Conditions using CNISO . . . . .	44
5.4 Combine Pairs Remove and RANSAC Halt . . . . .	46
5.5 Runtime Measurement for CNISO on COLMAP . . . . .	47
<b>6 Summary and Future Work</b>	<b>49</b>
<b>Appendices</b>	<b>54</b>
<b>A False Negative/Positive measures</b>	<b>55</b>
<b>B COLMAP Dataflow Enrichment</b>	<b>58</b>

# List of Figures

1.1	SFM General Pipeline . . . . .	9
2.1	Building Rome in a Day - Zooming verification . . . . .	12
2.2	SFM Building Rome In Day - Block Diagram . . . . .	13
4.1	Overview of Different SFM Systems . . . . .	19
4.2	CNISO Full Search method iteration demonstration: . . . . .	21
4.3	CNISO get Sigma method: . . . . .	22
4.4	Data Flow Feature Matcher Class On Colmap . . . . .	28
4.5	Data Flow Geometric Verifier Class On Colmap . . . . .	29
4.6	COLMAP Exhaustive Matching dataflow . . . . .	30
4.7	COLMAP's Vocabulary Tree main function . . . . .	31
4.8	COLMAP Query Function for Vocabulary Tree . . . . .	31
4.9	putting it all together for COLMAP's vocabulary tree matching . . . . .	32
4.10	Data Flow Geometric Verifier With CNISO in COLMAP . . . . .	33
5.1	Handling COLMAP runtime issues . . . . .	36
5.2	Vocab Tree Baseline Evaluation . . . . .	37
5.3	Evaluation of Vocab Tree in COLMAP - CNISO against VAV . . . . .	38
5.4	Evaluation of Vocab Tree in COLMAP - CNISO against VAV . . . . .	38
5.5	Evaluate Combinations CNISO RANSAC. . . . .	40
5.6	Evaluate Exhaustive . . . . .	40
5.7	Estimate Running Time With CUDA . . . . .	43
5.8	DB COLMAP Raw matches and Inliers matches . . . . .	45

5.9	DB before And After CNISO . . . . .	46
5.10	COLMAP with CNISO in RANSAC condition . . . . .	47
5.11	CNISO used both on pairs removal and RANSAC's halting condition . . .	48
A.1	False Positive/Negative Measures for image pair verification process . . .	56
A.2	false Positive/Negative as a funnction of $N_G$ and RANSAC thresholds. . .	57



# Chapter 1

## Introduction

When capturing image from a standard camera, we get a 2D descriptor of the 3D world. One of the major challenges in computer vision is reconstructing the 3D out of a set of scene images. This challenge is embedded in the fact that real world is 3D, while the images are 2D, hence single image holds missing information and the use of multiple images is intended to complete the missing information.

Structure From Motion (SFM) is the process of reconstructing 3D structure and camera intrinsic/extrinsic parameters (pose & orientation) from a given set of images of the same scene. SFM is referring to the sparse reconstruction of a static scene, and is a preprocessing stage for Multi-View-Stereo (MVS) that deals with the dense reconstruction.

SFM input is a set of images captured by uncalibrated set of cameras, and the output is a set of 3D points of the world and the projection matrix of each of the cameras. Therefore, it is widely applicable for geo-localization, augmented reality, autonomous cars, tracking, object recognition, noise reduction and more.

This problem can be described formally as an optimization problem, when the goal is to find the 3D structure and camera parameters such that an appropriate cost function is minimized. Under the assumption that the scene is static the common cost function to minimize is the re-projection error.

Finding the optimal model of a set of 3D points  $\{P_j\}$  and camera matrices that satisfy the optimization demand for the re-projection error is a hard problem that

has been studied for many years and has long history for modern approach. As will be described in SFM components section 1.1, SFM is composed of two main phases: scene graph build up phase, on which a graph describing overlapping images is build, followed by incremental/global 3D reconstruction, where scene 3D points and camera estimation take place.

One of the limitations of existing methods is the running time, since often hundreds of images should be considered. The goal of this project is to decrease the running time of the SFM pipeline stage using a State-of-the-Art Spatial Order verification, resulting in more robust and concise scene graph, with an improvement or at worst only negligible damage to the quality of the 3D reconstruction output. This will be done by using a novel algorithm for estimating the Correct Number of Inliers using Spatial Order (CNISO) introduced by Talker article [1].

There are several high standard SFM open-sources such as Bundler, Theia, and COLMAP that one can download, compile and extend for his own needs. This project's contribution will be an implementation of CNISOs algorithm in C/C++ language for integration to SFM open-sources, including reaserching and developing CNISO in COLMAP SFM open-source. Our implementation is published in Github (see link in [2]). Details and code for COLMAP integration can also be found on the github page.

## 1.1 Components of Structure for Motion

SFM standard data flow can be seen as a pipeline that consists of two main phases: scene graph build up, and 3D reconstruction with cameras calibration.

### 1.1.1 Scene Graph

First phase of a common SFM is building a Scene Graph, where nodes represent images, and edges represent overlapping images with successful geometric model computed.

At first, features are extracted for every image, usually using SIFT/SURF for accomplishing scale, pose and illumination invariance. Secondly, feature matching between every pair of images is done, mostly followed by verification using some geometric technique that involve Random Sample Consensus (RANSAC)[3][4][5][6]. RANSAC is intended for decreasing outliers contamination and computing Fundamental/essential matrix [5][7] that are relevant for the 3D reconstruction and camera calibration phase, where triangulation and bundle adjustment take place. This phase requires an initial guess and the fundamental/essential matrix serve this purpose.

Eventually, every pair of images can be considered as connected in the graph according to some decision making based on RANSAC output, such as threshold for the number of inliers.

### **1.1.2 3D Reconstruction and Camera Estimation**

Also referred as reconstruction phase, where triangulation and bundle adjustment take place. The input for this phase is the overlapping images described by the scene graph and matching points between a pair of images. In general, matching points can be computed here as well instead of given as an input, depends on the pipeline that was mentioned earlier.

There are two popular approaches, the first is known as global SFM [8] and the second is incremental SFM. In global SFM the entire scene graph is taken into consideration, so a relevant cost function with respect to all cameras and 3D matching points is defined and minimized for finding the optimal model. In incremental SFM at each step one image pair in the scene graph is chosen according to some strategy (mostly pairs with best overlapping scores are chosen first) and at each step a new image is registered and the 3D model is corrected and updated, thus gradually adding all images from the scene graph. Incremental SFM is widely adopted since for large scale image sets its running time and robustness are better.

### 1.1.3 RANSAC in SFM

Here a general description of the Random sample consensus (RANSAC) algorithm is given, and then the use of it in SFM context is specified into more details.

#### RANSAC algorithm:

Given some data mostly contaminated with outliers, RANSAC process randomly selects a small subset of the data, computes a model for this set, applies the model on the entire data and counts inliers that satisfy the model. This is done repeatedly for some number of trials, and the model with the largest set of inliers counted is chosen as the best fit model. The largest set of inliers found is kept for Re-Computing estimation of the final model according to the entire set of inliers.

Mostly, after RANSAC is done updated number of inliers is checked against some threshold. This is the final geometric verification mentioned for the scene graph edges. The stage mentioned earlier of feature matching and verifying geometries can be done naïvely between all image pairs resulting in  $O(n^2)$  comparisons. Alternatively, feature matching can be performed only on a selected subset using some preliminary fast test over all pairs resulting in a score according to which selection of  $O(n)$  image pairs is done. We next describe some techniques such as these into further details on the upcoming sections.

#### RANSAC applied to SFM:

Here, in SFM context, The input for the RANSAC process is some initial correspondence. The output is a reliable distinguish between inliers and outliers for reliable correspondence mapping, and an initial fundamental matrix between image pairs.

For each image pair selected, RANSAC is applied. The input data for the RANSAC is the correspondence (feature matching). The model is an essential/fundamental matrix that describes the geometry between the pair of images. An essential/fundamental matrix correlates a 2D point in one image to an epipolar line in the second image. At least 8 corresponding points are needed between the two images to solve the fundamental matrix model (known as the 8 points algorithm). So for every trial

of RANSAC this matrix is calculated, the 2D features points from one image are transformed to epipolar lines in second image, and if the corresponding point in the second image is close enough to the epipolar line then the pair of points are regarded as inliers.

In Figure 1.1 a general SFM pipeline is demonstrated.

## 1.2 Outlier Removal using Features Spatial Order

An efficient estimation for the Correct Number of Inliers is proposed in Talker et. al. [1]. The method is based on the Spatial Order of matched features (CNISO). Given the matching between features in images  $I_1$  and  $I_2$ , computed by e.g. Lowe's ratio Nearest Neighbours [9], the goal is to efficiently compute the number of correct matches before RANSAC is applied. The matching defines a permutation between the order of features in  $I_1$  and in  $I_2$ , when the order is defined by the  $x$  component of a feature. It was shown by Talker that the number of inversion in this permutation, the Kendall distance [10], can be used to compute the number of correct matches. Moreover, it was shown that the Kendall distance function for SFM purposes outperformed the Spearman foot-rule [11] distance function in every experimental aspect and therefore was suggested to be used.

The basic idea that the spatial order of correctly matched features in  $I_1$  is preserved in  $I_2$  [assumption  $A_1$ ], the spatial order of incorrectly matched features is random [assumption  $A_2$ ], and that the ranks of correctly matched features are distributed uniformly in  $[N]$  and in  $\sigma$  [assumption  $A_3$ ].

It is shown that under these assumptions, for  $N$  features matches (1-to-1 correspondence) contaminated with outliers, and a Kendall Distance  $K$  computed, the number  $N_G$  that stands for good matches and represents our inlier estimation, can be evaluated using a simple quadratic equation of  $N_G$ :

$$0 = \frac{1}{6}N_G^2 - (\frac{1}{2} - \frac{1}{3}N)N_G - N(N-1)(\frac{1}{2} - \hat{K})$$

Where  $\hat{K}$  is defined as:

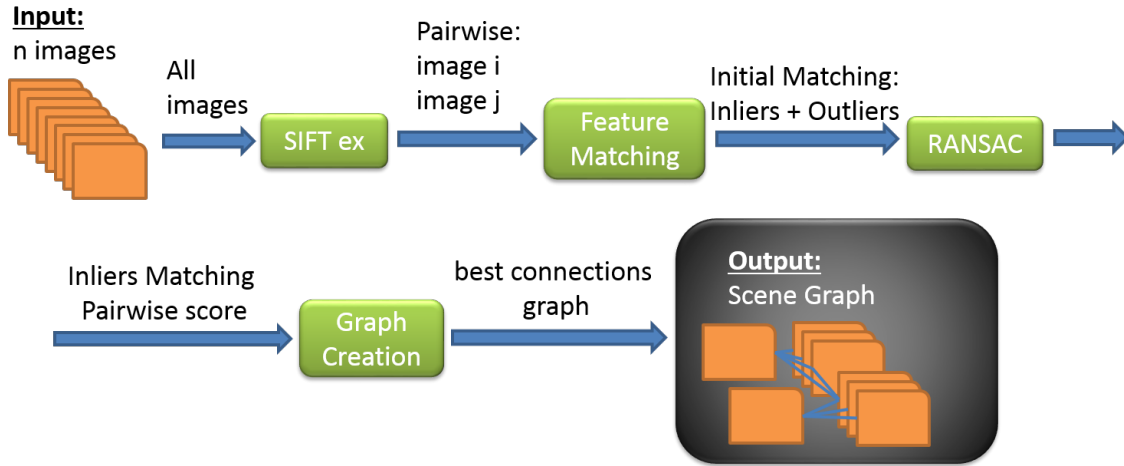
$$\hat{K} = \frac{2K}{N(N-1)}$$

K is the Kendall distance computed and N is number of feature matches. There are two solutions for this equation: for  $0 \leq \hat{K} \leq \frac{1}{2}$ , we take the only solution in the range  $[0, N]$ , and for  $\hat{K} > \frac{1}{2}$ , we set  $N_G = 0$ .

## 1.3 Brief Overview of this Project Report

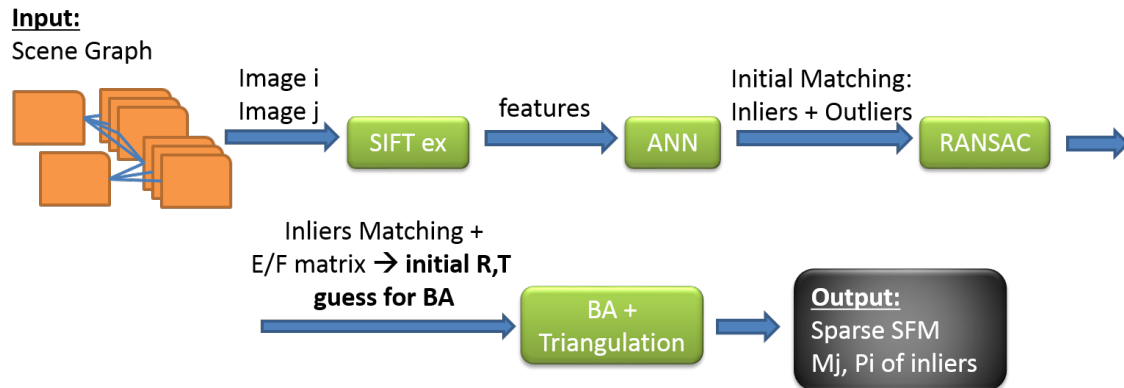
The goal of the project is to leverage in practice the efficiency and quality of CNISO in a State-of-The-Art open source SFM system that applies popular approaches and best practices improvements from literature.

At the following section popular related work for improving efficiency of SFM and RANSAC are mapped and discussed. Afterwards, CNISO implementation details are presented with respect to popular open-source SFM systems, in particular COLMAP SFM that was chosen for this project. In this section, challenges, planning consideration, and hints for efficient future work will be given both technical and technological. Finally, at the experiments section, evaluation will be thoroughly explained and comparisons to related works will take place.



(a) **SFM General Pipeline - common scene graph phase**

The input for this phase is a set of images captured on a static scene, and the output is a graph describing the overlap between every pair of images. The data flow of this phase is shown in a block scheme. At first features are extracted for each image (SIFT extraction block). At most this is SIFT descriptors but not necessarily. Then, for every pair of images feature matching is done. The stage of matching results in correspondence mapping between the features of the two images (image pair). Afterwards for each pair, due to matching given, RANSAC is performed for removing outliers, and keeping matching between inliers only. So, before RANSAC an initial matching was available and after RANSAC only inliers matches are kept. The next and final block is decision making for graph creation. At this block each image pair is examined against RANSAC inliers count and/or other criterion measured on earlier blocks (feature extraction, matching) and decided weather the pair is connected or not (an edge on the graph).



(b) **SFM General Pipeline - common reconstruction phase**

The input for this phase is the scene graph built and the output is a 3D reconstruction of the static scene including cameras calibration. At first, if needed, feature extraction is performed. Mostly features extracted on the scene graph phase is used here, but not necessarily. Coming up next, for each pair, Approximate Nearest Neighbour (ANN) feature matching is performed (ANN block), followed by RANSAC. On the same principle it is possible to use the matching and RANSAC output given from scene graph phase. However, on many occasions, a light coarse grained matching such as visual words distance is done on scene graph phase, and a more heavy fine grained matching such as NN is done at reconstruction phase. On the final BA block bundle adjustment and triangulation is performed in an incremental/global fashion for solving the 3D reconstruction and cameras calibration.

Figure 1.1: SFM General Pipeline

# Chapter 2

## Related Works

As explained on [12][13][14][15][16][1][17], one of the most heavy computation, which takes the largest portion of time in SFM, is the feature matching stage (including RANSAC process). It is a crucial task for a high quality 3D scene reconstruction, since it directly effects the quality of the scene graph and the initial solution for bundle adjustment, which are important for the incremental phase to converge fast and accurate. In this chapter we review related work of SFM. In particular, we consider the computation of the matching.

### 2.1 Feature Matching

Feature matching is an extensive studied field in computer science. There are variety of matching algorithms. The Nearest Neighbour (NN) on which all possible features pairs are compared in a brute-force manner is known to be good but is extremely expensive. Approximate NN (ANN) holds runtime improvements using hierarchical approach. Lowe's ratio NN suggests improvement for the NN algorithm by performing a ratio test between the first and second best match per feature.

Lighter feature matching techniques involve using vocabulary tree[18][19] of Visual Words (VW), where each descriptor is accosiated with the closest VW and matching is performed faster between the two images by matching features that are closest to each other on the vocabulary tree. Then, to complete a reliable feature



matching process, outlier removal should be applied. Thus, RANSAC process is executed and outliers are removed.

## 2.2 Choose a Subset of Image Pairs

In SFM context avoiding the feature matching bottleneck, with yet achieving high quality dense reconstruction is a major challenge. Many techniques and strategies were adopted over the years. In [12] the popular and perhaps most adopted SFM approach was proposed and developed. According to this approach, it is suggested to perform a light weight initial image pairs association based on Bag Of Visual Words (BOVW) histogram TF-IDF similarity [20] between all  $O(n^2)$  image pairs. Then filtering out image pairs is done by choosing for each image only  $k$  best overlapping images candidates based on the BOVW histogram similarity score, remaining with a graph of order  $O(n)$  overlapping image pairs candidates.

The heavy NN feature matching algorithm with RANSAC is carried out only on  $O(n)$  image pairs, thus gaining scalability of the SFM process in relation to images set size.

The common pipeline that was explained here is demonstrated in block diagram in Figures 2.1 2.2

In [17][21][22][23] both general and detailed overview of COLMAP is spelled out. In particular, in [21] Vote-And-Verify (VAV) verification is proposed to follow a vocabulary tree based feature matching over all  $O(n^2)$  image pairs. This relatively light weight process outcome is an initial  $O(n)$  edges scene graph, to be later verified using standard and expensive NN Lowe's ratio feature matching and RANSAC, to produce the final scene graph. The VAV method is a hybrid between RANSAC's hypothesize-and-verify framework and the Hough voting-based attitude[21]. The fast Spatial Verification VAV algorithm receives as an input a matching between two images and outputs an estimation for the inliers count.

On [17] [21] evaluation against the BOVW histogram similarity is done and VAV was announced to grant a robust scene graph that results in better performance

both in terms of running time and reconstruction re-projection error. Hence, VAV Spatial Verification strategy was advised as an approach for improving the SFM pipeline and COLMAP SFM open source system implemented its functionality.

An alternative method for choosing a pair of images for scene graph was suggested by Talker et. al. [1]. A suggestion to use CNISO's inliers estimation as an additional halting condition for RANSAC process was raised as well. Since this is the method we implement in this project, a detailed description is given in Chapter 3. In addition, according to Talker [1] using CNISO output number of inliers estimation  $N_G$  as an additional halting condition for RANSAC process saves significant without effecting RANSAC's output in context of SFM. Therefore, this will also be evaluated on this work.

## Building Rome In a Day (2011)

### 1. Dictionary + Histogram on $O(n^2)$ image pairs



### 2. Consider only $O(n)$ best image pairs and for each perform verification (verify = SIFT+ANN+RANSAC)

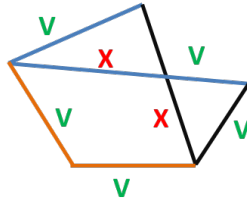


Figure 2.1: Building Rome in a Day - Zooming verification

The general scene graph approach of building Rome In a Day paper: at first on 1st step a light weight histogram similarity is computed for each image pair. Afterwards on 2nd step, only  $O(n)$  image pairs remain after filtering low histogram score (this is the initial scene graph edges), on which verification is performed: v are verified edges, x are disqualified edges. The final scene graph is composed only of verified edges.

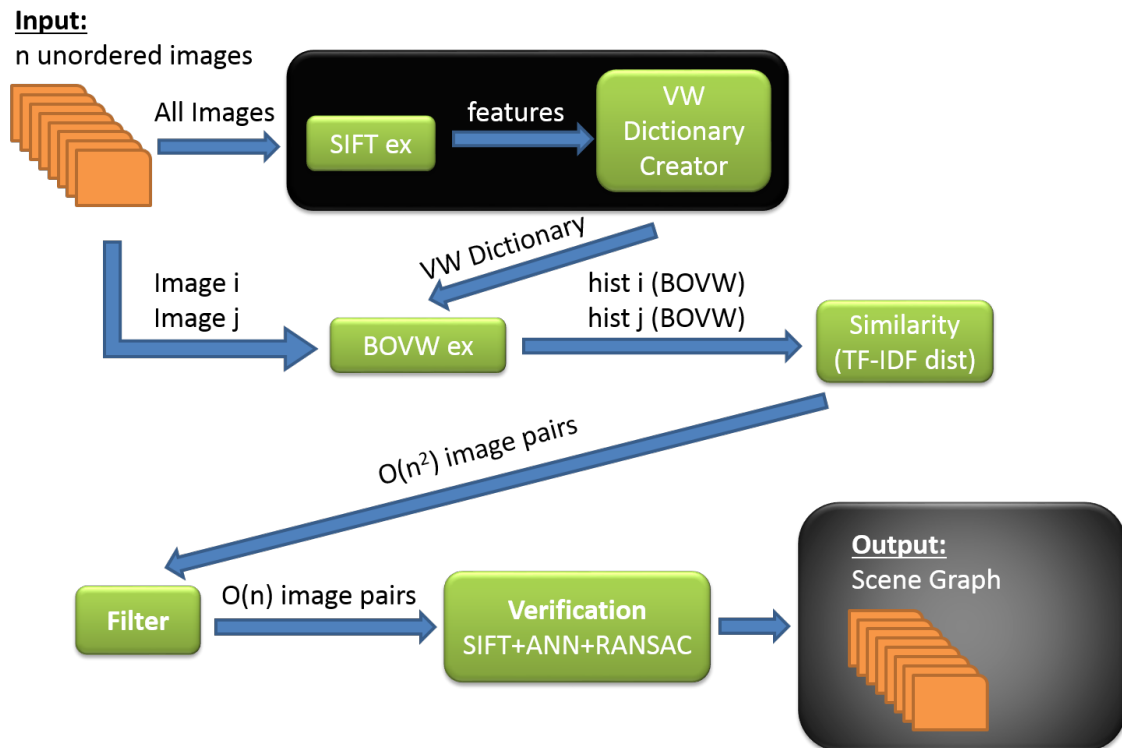


Figure 2.2: SFM Building Rome In Day - Block Diagram

Scheme of the entire blocks consisting the scene graph creation phase for building Rome In a Day. The VW dictionary is optionally created over all images of the given scene, but practically nowadays this step is unneeded since offline processed vocabulary trees over universal datasets are published and can be used instead. Feature key points are extracted for all images of the scene, each image pair histogram similarity is computed and only best score pairs build the initial scene graph, which go through heavy verification step for determining the final scene graph.

# Chapter 3

## Leverage SFM using CNISO

According to CNISOs paper [1] replacing BOVW filtering with CNISOs result in running time improvement for SFM since the pairs are more reliable, resulting in more robust and correct scene graph with less edges. Thus, even though CNISO consists of more computations for matching features between images, less RANSAC calls are needed. Hence, first phase of scene graph creation becomes faster and more robust.

In addition, since the scene graph describes better the overlap between the images, it is expected and shown how the process of minimizing re-projection error involving Bundle Adjustment and Triangulation is done faster (decreasing running time) and more precise (decreasing mean re-projection error) due to better initial guess and better overall association between images.

Moreover, an innovative attitude towards the dataflow is considered using CNISO. on the paper of Talker it is shown how an expensive NN Lowe's ratio feature matching over all  $O(n^2)$  image pairs, followed by CNISO, achieves better running time performance than running the light Vocabulary matching at first, use CNISO, and then running the NN Lowe's ratio only on  $O(n)$  image pairs. This is due to the fact that RANSAC initial matching input is less contaminated, and therefore can converge faster.

In Talker's paper there are 2D and 1D search techniques that take place. The 2D search refers to a full search where all possible sub windows are examined for

highest overlapping score. The 1D search on the other hand, refers to a two phase search approach which is faster but less accurate. More details are explained in Talker’s paper[1].

For this discussion some relevant definitions related to SFM scene graph creation strategies are shortly introduced:

$\mathbf{K}_2^V$  – CNISO test is done over an initial vocabulary tree matching, and the best pairs go through NN matching for scene graph build-up. CNISO is done using full search approach.

$\mathbf{K}_1^V$  – Same as  $K_2^V$  only that CNISO is done over a two-phase search approach.

$\mathbf{K}_2^L$  – CNISO test is done over NN matching, and the best pairs construct the scene graph. CNISO is done using full search approach.

$\mathbf{K}_1^L$  – Same as  $K_2^L$  only that CNISO is done over a two-phase search approach.

According to the Talker it is advised to perform the  $K_{1,2}^L$  approach with Kendall distance which performs better then  $K_{1,2}^V$  approach, both in running time and quality, and  $K_{1,2}^V$  outperforms BOVW histogram similarity approach both in running time and quality.  $K_2^L$  performs better quality results then  $K_1^L$  and  $K_2^V$  performs better quality then  $K_1^V$ . Clearly, by definition  $K_1^L$  is faster then  $K_2^L$  and  $K_1^V$  is faster than  $K_2^V$ . It should noted in this context that according to Talker the Kendall distance function performs better than Spearman foot-rule in all dataflows mentioned and therefore will be used during this project.

In this project CNISOs spatial verification approach will be directly evaluated against the VAV spatial verification approach on the  $K_2^V$  dataflow attitude in COLMAP SFM. In addition, CNISOs  $K_2^L$  innovative dataflow approach will be added to COLMAP and relevant R&D evaluations and comparisons will take place.

In addition, according to Talker [1] using CNISO output number of inliers estimation  $N_G$  as an additional halting condition for RANSAC process saves significant time without effecting RANSAC’s output in context of SFM. Therefore, this will also be evaluated on this work.

Before getting into more details in upcoming sections it should be mention up-

front that Talker's tests were implemented in Matlab, and here in this project we ran them in C++ for COLMAP SFM open source.

# Chapter 4

## Research & Development

As mentioned earlier, the goal of this project is testing in practice the efficiency and quality of CNISO in a State-of-The-Art open source SFM system, where evaluation and comparison against best practical applications can be done.

For this purpose it is necessary to: (i) Choose an appropriate up to date SFM system, which refers to most modern and novel applications. (ii) Download, compile and execute it with some known image dataset as input. (iii) Implement CNISO's algorithm in the required programming language. (iv) Integrate CNISO's implementation into the SFM system and compile it. (v) Perform R&D (Research and Development) of the SFM pipeline using CNISO capability added. (vi) Evaluate each change against other proven approaches described in related works section.

### 4.1 Choosing SFM system

Choosing a proper SFM system holds some important considerations taken into account. When coming to choose an appropriate SFM system some very good options arise, among which standing above all are the closed-source Visual SFM, and the open-sources Microsoft's Bundler, Theia, and along them almost side by side the emerging COLMAP. A wide list of SFM systems can be found in Figure 4.1.

At first, a primary work with Visual SFM was done for getting the feel of an

SFM system and exploring the variety of options. Visual SFM is very suitable for researches of these kinds and it gave a good sense of what to expect. In addition, Visual SFM holds simplicity for work as it is designed cross-platform especially for the purpose of serving users that intend Research for SFM and do not intend to develop (Windows, MAC, LINUX etc.). In this sense, it is better suited and tested for a daily basis cross-platform application. This is unlike open-sources that are addresses the audience of developers, and therefore may have a deficiency in fulfilling a more simplified purpose.

After working with Visual SFM for a kick-off, attention was directed to development. A variety of SFM systems were explored. Three of the most major systems mentioned where the Bundler, Theia, and COLMAP. While Bundler and Theia are longer period of time on the market, it seemed COLMAP introduced some more updated applications referring to papers from 2016-2017 that where not applied nor in Bundler nor Theia.

Next challenge was to check whether the SFM pipeline supports the functionality that CNISO algorithm is based on, such as Lowe's ratio nearest neighbour matching. According to Theia and Bundler code documentary they do support. According to COLMAP documentary there was a little bit of doubt since there was no direct notation available, but since all of the three SFM systems were implemented in C/C++ it seemed choosing COLMAP is reasonable in sense of reducing the risk for in-vain C++ implementation. Therefore, after verifying that the relevant part for Bundler, Theia and COLMAP are in C/C++, choosing to implement CNISO in C++ was the decision taken. It will be noted that the compiler of C++ support C and the three SFM systems compilation combine C and C++, so there is no technical limitation that can not be handled for compiling with C++ rather than C. Furthermore the relevant files according to Application Programming Interface (API) of each system were C++ so technically it is a natural choice as well.

More deep details of COLMAP dependencies will be discussed in the upcoming sections.




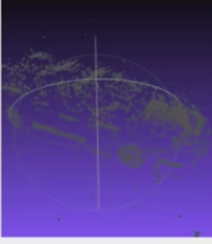



	Point cloud	Dense cloud	Surface	Textured
				
Bundler				
CMVS				
COLMAP		CUDA only	CUDA only	
Meshlab				
MVE				
MVS-texturing				
openMVG				
openMVS				
Theia				
VisualFSM				

Figure 4.1: Overview of Different SFM Systems

## 4.2 Implementation of CNISO in C++

### 4.2.1 Details

We implemented CNISO algorithm in C++ programming language for open-source projects such as COLMAP, Theia, and Bundler.

CNISO C++ main function gets as input two integer vectors with X positions values of the image features, one for each image. The two vectors are of the same size and they reflect the matching between the two images: the  $i$ 'th index of the two vectors are X positions that correspond to each other (a match). The output is  $N_G$ , and the overlapping sections given both by the indices and the X positions values.

SFM open-sources projects of such as Theia, Bundler and COLMAP lean on CMAKE cross-platform technology, for which adding source files and header files require updating cmakeList.txt. So, In order to make it simple for the community and ease testing & integration later on, implementation is done within a single C++ file. This way, CNISO algorithm can be involved within an existing single C++ file

of the SFM open-source.

MATLAB Code for CNISO can be found on the site of its author Talker[24][25] and was used as a reference for implementation and for output comparison testing. For implementation to be done that fits open-sources SFM systems it was necessary to understand CNISOs matlab code and algorithm and implement it with proper API adjustments in order for it to fit a general SFM open-source, i.e. to grant SFM open-sources systems a comfortable CNISO API for an immediate access to its routines.

Thus, an important API decision was allowing input flexibility. Therefore, the input for the main function of CNISO C++ implementation is one-to-one correspondence, not necessarily ordered by the X position and not necessarily with unique X values. This is in contrary to the MATLAB code whose generated input was a one-to-one correspondence ordered by the X position. Regarding the un-uniqueness of X position it should be noted that according to CNISO's this is a qualified input thanks to statistical characteristics. It was very important to expand CNISO's functionality for the purpose of making it available for open-sources as widely as possible, in particular for COLMAP integration.

So, finally the main function for CNISO is comprised of two tasks. First one is getSigma method by which the API adjustments is done and  $\sigma$  output is created. This output is a single vector,  $\sigma$  such that  $\sigma(i)$  consists of the rank of the matched feature of  $i$  in image  $I_1$  in  $I_2$  according to the  $x$  component of their location. Second one is FindOverlapFullSearch that encapsulates searching for the best overlapping region between the two images. This method receives sigma as an input, searches for the best overlap and outputs  $N_G$  estimation of inliers according to CNISO's algorithm. Another output of this function is the best overlap regions between the two images.

For a single iteration of overlapping search, there are given two subregions, one for each image. These subregions are checked one against another by some distance criterion. In our project, counting number of inversions, also known as the Kendall

distance, was implemented.

The following is demonstrated in Figures. Figure 4.2 Figure 4.3

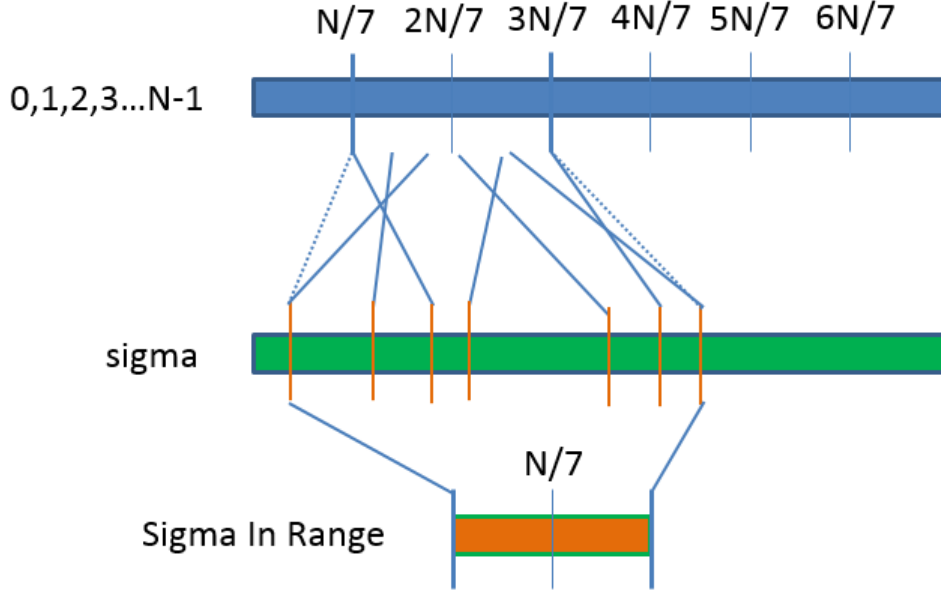


Figure 4.2: CNISO Full Search method iteration demonstration:

On this step the sub section to compare is from  $N/7$  to  $3N/7$  indices. for these  $\sigma$  is computed out of the corresponding keypoints, and the effective  $\sigma$  for this range is extracted, and  $N_G$  is computed for.

### 4.2.2 Unit Testing

Testing of CNISO implementation was done by creating synthetic input in Matlab using the function `randperm`, then running these sequences generated on both Matlab code and on CNISO C++ implementation, and comparing results. For large arrays of size over 100 and up to 16000, this involved writing a matlab function that prints the generated array in a C++ array syntax, copy-paste the printed text in C++ syntax to the CNISO C++ project and run it.

In addition, the API functionality of `getSigma` function was tested by inserting vectors with repetitive X position values and with mixed order and verifying that the results are indeed invariant to order changes, and correct.

- CNISO API adjustments – getSigma function

**Input:**  $x_{1j}$  match  $x_{2j}$  and are unordered integer position values

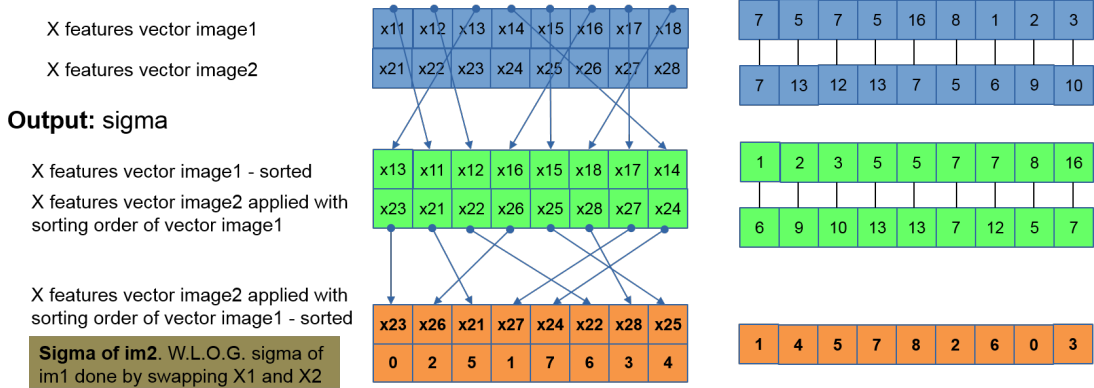


Figure 4.3: CNISO get Sigma method:

The input is a 2D array of X values, where cell  $i$  contains X values of two corresponding features between image1 and image2. An unordered and repetitive X values may appear for features of any images. On the left a general description of the process is given, followed by a numerical example on the right. At first, the 2D array is sorted using the X values of the first array. Then, for simplicity and conform to Talker's implementation and definitions, unique values from 1 to N (or 0 to N-1) should be assigned for  $\sigma$ . This is done by getting the sorting order indices demonstrated by the transition from green to orange. Using the indices order automatically generates unique numbers between 0 to N-1.

## 4.3 Colmap Build

As stated previously, COLMAP was the open-source SFM system chosen for the project. SFM system is very complicated at default since it involves massive computations for non-linear algorithms optimization solutions such as bundle adjustment process, mandatory 3D GUI windows for results display and research, access to image files formats, DB management.

As such, COLMAP SFM depends on heavy open-source libraries some of which depends on others themselves. Specifically speaking COLMAP is built above google's Ceres-Solver, glew, gflags, glog, Eigen3, Boost, suiteSparse, QT, OpenGL, SQLite, Eigen3, freeimage.

In addition, a special and recommended capability using CUDA GPU was adopted in COLMAP. In order for this capability to take place it is needed to add compatible hardware GPU with CUDA enabled capability. These are mostly supported by graphics card of NVIDIA such as GeForce GT series. If one wants to make use of

this advanced feature they need to buy some GPU that is CUDA enabled, install the proper driver from NVIDIA site, install NVIDIA drivers and CUDA, and then install COLMAP through CMAKE with CUDA flags enabled.

For working with CUDA one should prepare his computer in advance by connecting an advanced NVIDIA graphic card and install massive CUDA toolkit. According to COLMAP documentation CUDA capability decreases significant running time of the feature extraction and feature matching functionality. This will be further explained and discussed in the experimental section. It will be demonstrated that working without CUDA for large image datasets performs poorly in the feature extraction and matching stages. However, during this project research of COLMAP SFM pipeline and dataflow, it will be shown how one can overcome a lack of CUDA capability for the purpose of pure research and development by adopting some techniques based on COLMAP DB management (not effective though for real time purposes of running all COLMAP pipeline stages). It should be pointed out that on other SFM systems other than COLMAP, CUDA is not a recommendation, and it was found that CUDA hardware requirements are a drawback in that sense (nowadays Intel graphics cards HD series does not support CUDA).

In this project the work was done without CUDA capability and this turned out to be a very challenging task that involved careful decisions carried out due to a thorough code research with deep understanding of COLMAP data flow and DB management. On this spirit resides an advantage for working some low level along with high level evaluations.

Firstly, It should be mentioned that the build system upon which COLMAP is build is CMAKE. CMAKE is a tool for creating cross-platform open-sources. The input for CMAKE is a cmakeList.txt file that maps the dependencies between all files in the project, and its output is a makefile in a format that can be run by MAKE program (MAKE is a program that builds C/C++ according to makefile). Therefore, first step is installing CMAKE.

Afterwards BOOST and QT should be installed with some extra QT packages to

support the suitable compiler (These does not come on default on QT installation process).

Here QT can also serve as a cross-platform Integrated Development Environment (IDE), but if one prefers to work with Microsoft Visual Studio (MSVS) IDE, for Windows OS, he can do so with some proper commands and references to MS visual C++ (MSVC) compilers. At the last 2-3 years Microsoft has applied its IDE for other Operating System such as LINUX, given the name Microsoft Visual Studio Code. In this context, COLMAP documentary stated that MSVC 2013 and above should support COLMAP project.

After installing heavy Boost and QT (QT takes around 1-2 days minimum due to its large size), all other dependencies in the appropriate order should be build with Ceres-solver being the last one. Afterwards the last step is building COLMAP upon all. In this context, it is important to mention that COLMAP includes helpful python build scripts for all platforms. These scripts require python 3.4+ version and they are very useful for Windows OS especially. Linux installation guide on COLMAP site was found good enough to get the build done.

In this project the platform for R&D and experiments was Windows PC with IDE of MSVS 2017 and Intel graphic card HD 2500.

After installing CMAKE, BOOST, QT and MSVS 2017 manually on Windows7 platform, COLMAP and its dependencies were installed and build using the python script. At first, COLMAP python script didn't work properly and didn't complete the installation. After running the python in debug mode with external parameters, a bug in the python script for windows was found, fixed, and then the script was completed successfully. The updated script can be found on projects github page[2].

The build process was done in both release and release with debug info in order to be able to debug the code research and development of implementation and dataflow.

It was attempted to use CUDA but the failure to download it from the site, combined with PCs hardware limitations blocked this effort at its peak.

### 4.3.1 Conclusion and best practices for COLMAP Build

COLMAP build was carried out on a PC QuadCore i5 8 GB RAM with Windows7 OS using MSVS IDE, and also on a Lenovo ThinkPad E330 laptop with Ubuntu 16.04 OS using QT IDE. COLMAP open-sources dependencies and installations required around 100 GB free space on hard disk.

The build process on Ubuntu was way faster and simpler once starting to work with Linux since it guarantees more support for developers on the internet.

The Think-pad with no CUDA however was no way near handling the tasks of COLMAP and it took days to complete the task of feature matching. The PC however, due to it's better general specification was able to perform these tasks under a consistent and controlled limits, so eventually it was chosen to perform R&D and experiments on.

Working with COLMAP on Windows7 however was more challenging than expected, since the build process was done with less help on the internet and since COLMAP documentary was lacking for Windows as a hidden bug was to be captured in its python build script that had to be fixed.

To sum it up, it is recommended to get COLMAP started with QT also as the IDE since it supports CMAKE build system and GUI changes can be applied using its graphical QT windows creators. This can not be done immediately with MS Visual Studio. Moreover, it is recommended to have a CUDA enabled GPU with NVIDIA drivers and CUDA toolkit installed upfront. If this gets to complicated, it is recommended to consider change to another SFM system other than COLMAP or to adopt the techniques and strategies of work presented in future section. There are benefits for working with no CUDA since it enforces better understanding and separation capabilities between the different stages as will be demonstrated on the continuance.

In addition, working with the GUI is a must. It is highly recommended to work with the GUI and not be satisfied with results shown on a remote terminal. The GUI and the reconstruction save a lot of time when one needs to understand what

went wrong.

Moreover, building with debug mode is highly recommended since debug is a basic tool for development and navigation. It is advised to verify that the project does compile in debug or or release in debug info modes.

## 4.4 COLMAP Pipeline Code & Dataflow

COLMAP Feature Extraction: In COLMAP features extracted are SIFTS. It is given the possibility to interactively control parameters of SIFTS. One interesting and important parameter is the peak threshold, which controls the sensitivity, thus increasing this parameter results in choosing stronger features.

COLMAP Feature Matching (including Geometric verification):

Several methods are available in COLMAP. According to documentation the exhaustive matching technique is the preferable choice when dealing with relatively low number of images (up to several hundreds of images), since computation becomes an issue here for large image datasets.

**COLMAP Exhaustive matching** applies the technique of Nearest Neighbour with Lowe's ratio where each feature in one image is tested against all features in second image and best one is taken into account, with respect to some distance function in descriptors space. Afterwards, RANSAC is performed on all image pairs for keeping only pairs that satisfy required minimum number of inliers.

This approach is announced to achieve best results, however it is time consuming and can perform only on smaller datasets. In more general (not only in COLMAP context), it should be said that NN matching is considered good but time consuming approach, leading to some modifications such as ANN or NN Lowe's ratio, that are supposed to improve running-time and complexity. Still these techniques become significantly slow for large datasets collections, where number of image pairs increase with respect to  $O(n^2)$  and NN matching becomes a bottleneck. This led to some image pairs pre-filtering techniques such as histogram similarity (BOVW) that keeps only order of  $O(n)$  image pairs[12] and/or replacing NN matching technique with



vocabulary trees technique. In COLMAP these techniques can be accessed using the Vocab Tree matching.

COLMAP introduces NN matching approach over all pairs by using the exhaustive matching described between all image pairs. Other relevant techniques can be accessed using the Vocab Tree matching.

**COLMAP Vocab Tree matching** applies the VW vocab tree matching between all image pairs with significant running time improvement. Using the GUI the option of `num_of_images` is for choosing the best  $O(kn)$  image pairs according to some spatial verification test. Afterwards, NN Lowe's ratio matching is done only between the  $K \times n$  highest score image pairs, among which RANSAC is performed for keeping only pairs that satisfy required minimum number of inliers.

For CNISO the 1-to-1 correspondence needed is achieved by assigning the value `num_nearest_neighbours = 1` in the vocab tree GUI. In COLMAP, no implementation of BOVW is done. The test for choosing good image pairs candidates is the Vote-And-Verify method.

In COLMAP pipeline the data flow implemented is that for each image,  $K$  images with the highest scores are chosen.  $K$  is interactively chosen by the user using the GUI.

Vote-And-Verify algorithm, similarly to CNISO, counts effective number of inliers based on some spatial characteristic technique. Thus, with immediate adoption of COLMAP dataflow, Vote-And-verify should be replaced with CNISO and further adjustments are avoided for clean one-against-one comparison for evaluation.

**Note:** In COLMAP `num_images_after_verification` is denoted as in  $K-1$  in this paper. Thus, for  $k=10$  the assignment is actually 9.

Data flow diagrams for COLMAP are presented in Figures 4.4 4.5 4.6 4.7 4.8 4.9. In Figures 4.4 4.5 the traverse matching and verification tasks of colmap are drawn and demonstrated. Afterwards, in Figure 4.6 the exhaustive matching with the traverse matcher function is described, and on 4.7 4.8 4.9 the vocabTree matching.

# SiftCpuFeatureMatcher class

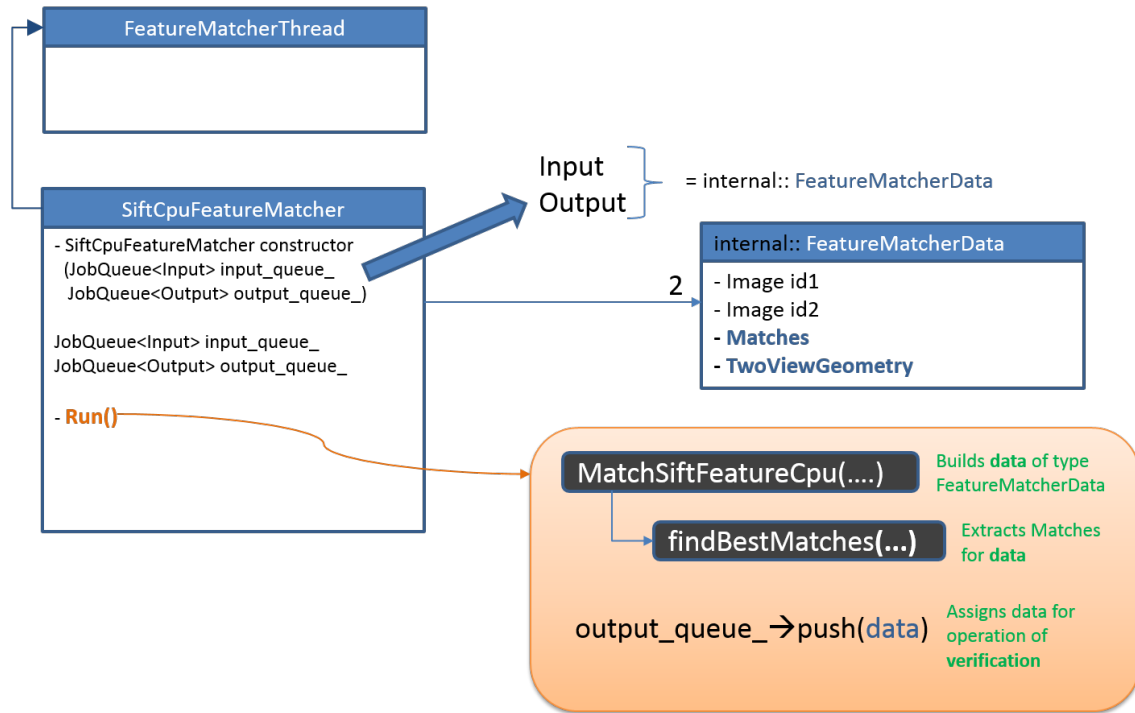


Figure 4.4: Data Flow Feature Matcher Class On Colmap

## 4.5 Integrating CNISO into Colmap

This will include:

- Removing image pairs candidates before RANSAC using CNISO as a geometrical verification and comparing it against baseline, in particular to the Vote-And-Verify approach.
- Using it as a halting condition of RANSAC.
- Combining both - pairs removal and RANSAC's halting condition.

### 4.5.1 Remove pairs using CNISO

Here we refer to the idea of using CNISO algorithm as a test for determining the overlapping score between each pair of images. According to Talker's paper [1] an improvement against BOVW histogram similarity test was achieved using an SFM

# TwoViewGeometryVerifier class

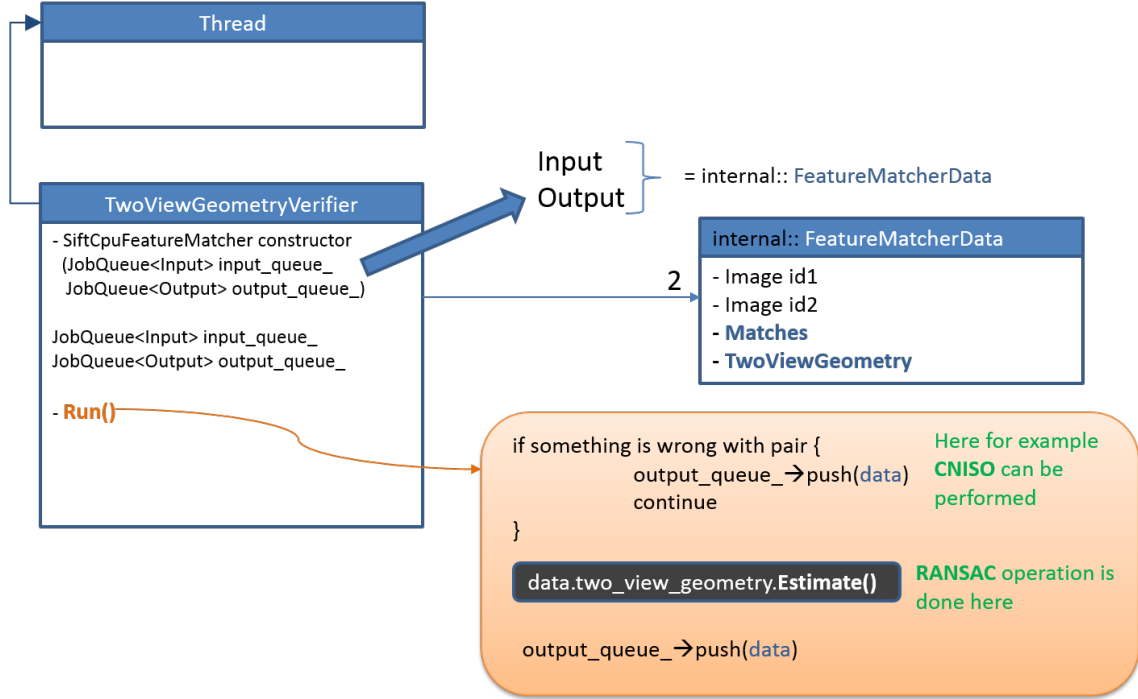


Figure 4.5: Data Flow Geometric Verifier Class On Colmap

pipeline of performing the test for each image pair, and thresholding. In [1] this filtering was implemented in MATLAB on both  $K_2^V$  and  $K_2^L$  approaches and proved itself against BOVW. We remind that  $K_2^V$  relates to performing the verification test over Vocabulary tree matching, and  $K_2^L$  relates to performing the verification test over NN matching. Only then RANSAC is done over NN matching of the remaining image pairs.

## 4.5.1.1 CNISO Remove Pairs in COLMAP's Vocabulary Tree

Here, on COLMAP SFM pipeline, the version of  $K_2^V$  approach is represented by performing the vocab tree matching with VAV spatial verification, and choosing for each image the best K images with highest VAV score. First we replace the VAV spatial verification as is with CNISO in the role of a spatial verification.

In Figure 4.8 it is pointed where VAV verification is executed on COLMAP code. Our implementation is done by replacing the VAV functionality with CNISO's. More

# ExhaustiveFeatureMatcher::Run()

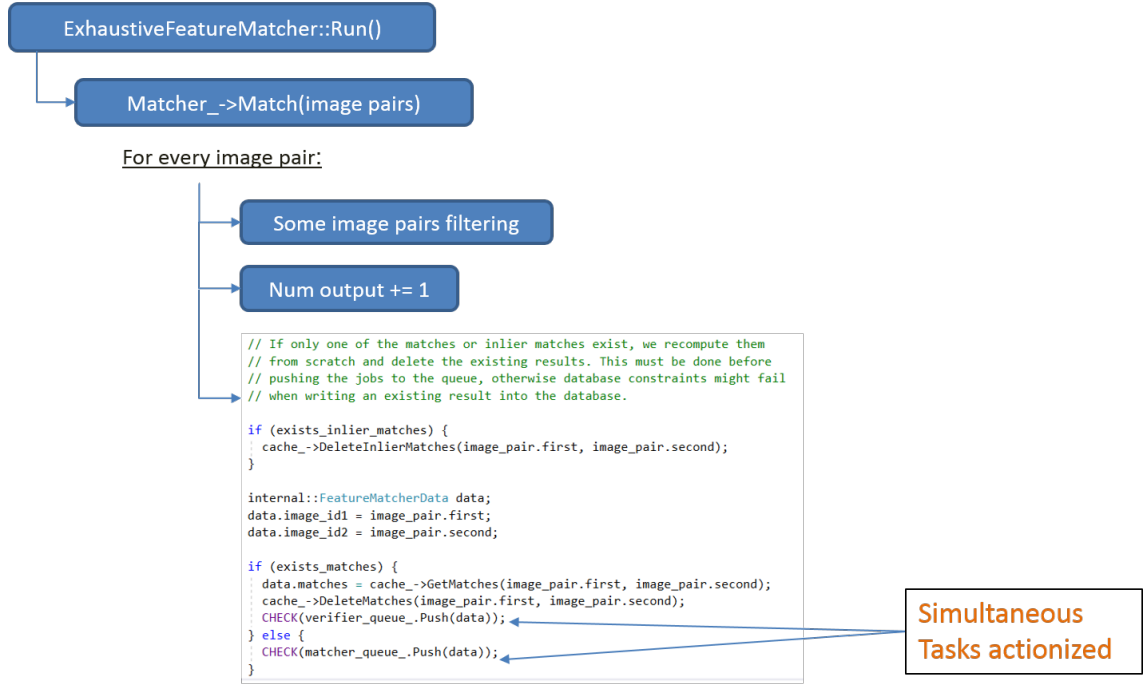


Figure 4.6: COLMAP Exhaustive Matching dataflow

Data Flow for Exhaustive matching on COLMAP is composed mainly on NN Lowe’s ratio matching function `Matcher→Match`. For each image pair out of the  $O(n^2)$  image pairs, a straightforward preliminary filtering is done, followed by verification task NN Lowe’s ratio matching, which on its turn executes the same verification task. verification queue relates to `TwoViewGeometry→Run` function described in Figure 4.5, and matcher queue relates to `SiftCpuFeatureMatcher→Run` function described in Figure 4.4

concrete code documentation can be found in the github page[2].

**Note:** It should be pointed out that implementing the exact MATLAB pipeline in [1] is complicated on COLMAP and explained thoroughly in Appendix B.

## 4.5.1.2 CNISO Remove Pairs in COLMAP’s Exhaustive Lowe’s Ratio

In addition, the version of  $K_2^L$  does not take place in COLMAP. So the exact version of it according to CNISO’s paper was implemented by performing COLMAP’s exhaustive matching and executing CNISO’s threshold filtering (same as RANSAC threshold) before RANSAC is executed. In Figure 4.10 it is shown the exact location on the data-flow in the function `TwoViewGeometryVerifier→Run()`, which is described into more details in [22][23].

Evaluation and detailed comparisons are demonstrated on the following sections.

VocabTreeFeatureMatcher::Run()

```
// Index all images in the visual index.
IndexImagesInVisualIndex
```

```
// Match all images in the visual index.
MatchNearestNeighborsInVisualIndex
```

visual\_index->Query

```
for (size_t i = 0; i < image_ids.size(); ++i)
```

```
for (const auto image_score : image_scores)
    image_pairs.emplace_back(image_id,
    image_score.image_id);
```

matcher->Match( image\_pairs )

Figure 4.7: COLMAP's Vocabulary Tree main function

Data Flow Vocabulary Tree main function, for which at first all possible image pairs are declared as candidates on the IndexImageInVisualIndex block. Afterwards, inside the Query block, for each image pair an initial coarse vocabulary matching is done detailed in Figure 4.8. Finally, The same Lowe's ratio matcher function described in Figure 4.6 is performed.

Query

QueryAndFindWordIds

```
if (options.num_images_after_verification <= 0)
  { return;
```

```
for (auto& image_score : *image_scores)
```

```
// Enforce 1-to-1 matching
```

```
image_score.score +=
    VoteAndVerify(vote_and_verify_options, matches);
```

Figure 4.8: COLMAP Query Function for Vocabulary Tree

Query function is responsible for the vocabulary tree matching. Each image pair undergoes a 1-to-1 correspondence process using vocabulary tree, followed by the VAV verification.

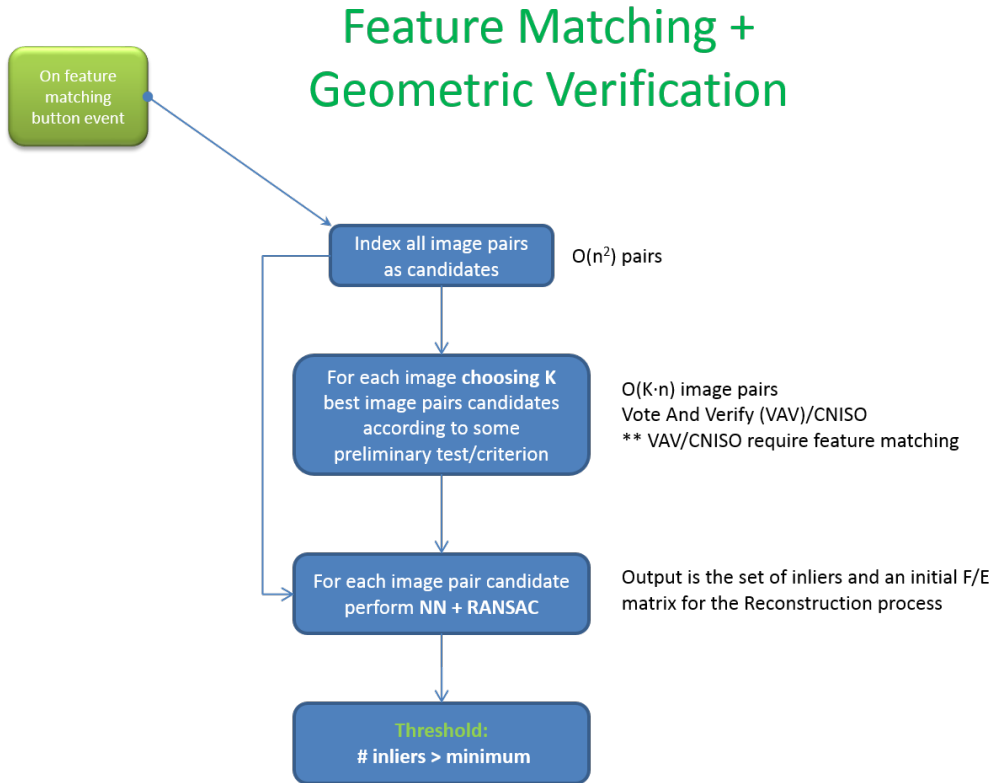


Figure 4.9: putting it all together for COLMAP’s vocabulary tree matching  
All images are indexed to each other, performing the  $O(n^2)$  pairs. For each one only K best VAV score remain as candidates, for which NN+RANSAC is performed. Then, image pairs are threshold based on RANSAC’s inliers count.

#### 4.5.2 RANSAC halting conditions using CNISO

Here we relate to the idea in [1] of using CNISO as an additional halting condition to RANSAC’s loop. This was done by performing CNISO’s test between the current image pair inside COLMAP’s RANSAC’s functions, computing  $N_G$ , and stopping the loop if inliers set size is higher than  $N_G$ . This was done, without interfering the original conditions, i.e. original halting conditions were unchanged and we will stop if RANSAC tells us to stop independent of CNISO.

Code documentation can be found on the github page [2]. Evaluation of performance over COLMAP’s exhaustive matching approach is carried out on the following sections.

# CNISO integrated in TwoViewGeometryVerifier::Run

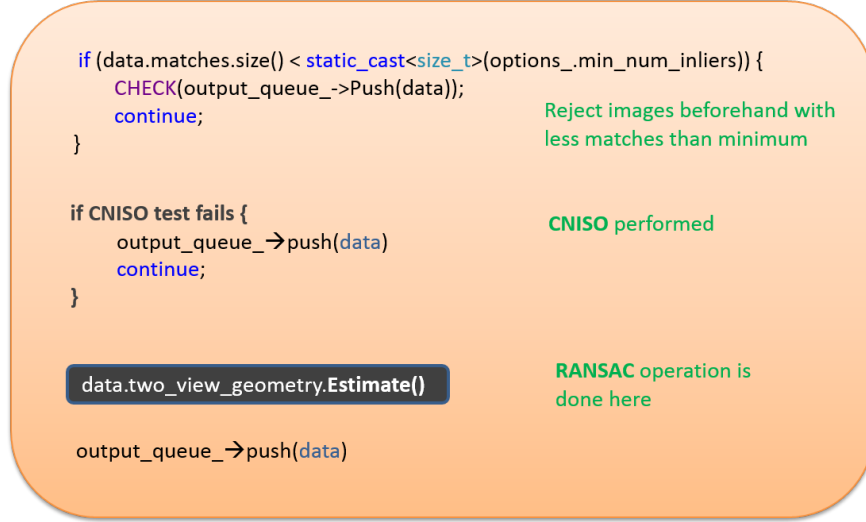


Figure 4.10: Data Flow Geometric Verifier With CNISO in COLMAP

Two View Geometry function data-flow at first rejects some image pairs if number of initial matches (contaminated with outliers) is less than minimum inliers required. Then, CNISO function is performed for the pair the pair is rejected on behalf of CNISO's indication. If CNISO's test indicates success then RANSAC estimation is performed by two\_view\_geometry→Estimate function

## 4.5.3 Combine pairs removal and RANSAC halting

Here, an implementation of COLMAP's exhaustive matching  $K_2^L$  with CNISO filtering described in Section 4.5.1 was done, with the addition of halting RANSAC if condition of  $N_G$  as described in Section 4.5.2.

It was decided to be done over  $K_2^L$  due to results on experimental section that suggested a relative advantage to Exhaustive with CNISO in terms of quality results, and the objective was to test their performance together, since they are supposed to introduce some dependency behaviour.

# Chapter 5

## Experiments

This chapter will focus on evaluation of CNISO improvements for COLMAP. This includes baseline pipeline evaluation with relevant analysis of modes and parameters impact on quality and runtime. Firstly, a preliminary work on COLMAP is done in order to bypass hardware limitations. Details for how to start working with COLMAP for R&D evaluation are mainly referenced through section 5.1. Then, at the following sections evaluating COLMAP will be discussed under CNISO experimental additions that were described earlier in Section 4.5 for CNISO integration section. Further needed experiments and comparisons against COLMAP baseline will be made and explained through this section. This chapter's experiments will be accompanied by ongoing recommendations for practical techniques and ideas on how to work with COLMAP under Hardware limitations of a non-CUDA GPU nor CPU.

During the process of evaluation some difficulties of hardware limitations have been treated in ways that can be inherited and adopted. These techniques will be mentioned whenever relevant in order to give the full picture of dealing these limitations in COLMAP for the purpose of performing R&D on COLMAP despite and thanks to Hardware limitations.



## 5.1 Handling Feature Extraction issues in COLMAP

At first feature extraction was done on datasets of 5 images of size 5616x3744 pixels. This process took about 3 minutes averaging 0.6 minutes per image. This was due to CPU v.s. GPU performance difference. For decreasing running time in order to handle large collections, the images were down sampled to 1280x720 pixels. To sum it up, **the problem of slow feature extraction due to non-CUDA GPU hardware limitation was handled by decreasing image size**. This resulted in 5 seconds for an image, without affecting the number of SIFTS extracted (just changing scale).

However, at this point the number of features extracted was about 10,000 per image, due to COLMAP default peak threshold option parameter for SIFT extractions. Since we did not use CUDA GPU this was a serious bottleneck for the process of feature matching, which was done over an enormous amount of features. Running time of the feature matching stage in vocab tree with  $k=1(!)$  was around 5 minutes for 5 images. Existing matching methods are much faster than that, but COLMAP was optimized for GPU, hence its efficiency is low for non-GPU use.

Therefore, an adaptation of SIFT's peak threshold was done: peak threshold of 0.00667 led to  $\sim 10000$  features as mentioned above, Peak threshold of 0.01667 led to  $\sim 4000$ -5000 features, and peak threshold of 0.02667 led to  $\sim 2000$ -3000 features. For an ability to work with larger datasets the option of  $\sim 2000$  features for 1280x720 pixels was more suitable either in sense of reducing running time, meeting CNISOs advised number of features according to Talker's paper documentation [1], and avoiding significant deterioration in reconstruction results (Figure 5.1).

### Summing it up:

Working with 1280x720 size images  $\rightarrow$  Enhance feature extraction run-time without SFM quality deterioration due to SIFTS scale invariance property, which dictates approximately same key points with similar count. Moreover, it is important to mention that even if features were not scale invariant, the fact that all images un-

dergo same factor down sampling should be enough since still other features would be extracted and in general it is commonly assumed that performance on average will be the similar for large image data sets.

Working with peak threshold of 0.02667 that lead to  $\sim 2000$ -3000 features, independent of image size  $\rightarrow$  Enhance feature matching (geometric verification) run-time with insignificant deterioration to quality performance, and hence not to comparison planned for evaluation as well.

**These are our fundamental adjustments to handle PC's non-CUDA hardware limitations.**

peak threshold 0.01667: num features $\sim 4000$ -5000				
average over 3 times:				
	Exahustive 15 inliers RANSAC	Exahustive 30 inliers RANSAC	Vocab No filtering 15 inliers RANSAC	Vocab No filtering 30 inliers RANSAC
Cameras	30	31/32/32	29	32/24/27
Images	30	31/32/32	29	32/24/27
Registered images	30	31/32/32	29	32/24/27
Points	6748	6227/7634/6660	7167	7854/7308/7836
Observations	26218	25078/30106/26544	25440	30160/25010/28045
Mean track length	3.8853	4.0273/3.94367/3.98559	3.5496	3.84008/3.42228/3.57899
Mean observations per image	873.933	808.968/940.813/829.5	877.241	942.5/1042.08/1038.7
Mean reprojection error	0.567048	0.49339/0.541828/0.508293	0.570544	0.546784/0.489788/0.517
Running Time Matching	41 [minutes]	40.324 [minutes]	46.688 [minutes]	46.598 [minutes]
Running Time Reconstruction	2.342 [minutes]	3.888/1.666/2.049 [minutes]	4 [minutes]	2.5/5/4 [minutes]
peak threshold 0.02667: num features $\sim 2000$				
best 3 out of 4:				
	Exahustive 15 inliers RANSAC	Exahustive 30 inliers RANSAC	Vocab No filtering 15 inliers RANSAC	Vocab No filtering 30 inliers RANSAC
Cameras	28/26	31/31/28	24/24	27/27
Images	28/26	28/26/1931/31/28	24/24	27/27
Registered images	28/26	28/26/1931/31/28	24/24	27/27
Points	2138/2423	3217/3137/2750	2585/2713	2216/2735
Observations	8048/9513	11911/11855/10847	8633/9232	9037/10864
Mean track length	3.76427/3.92612	3.70252/3.77909/3.94436	3.33965/3.40288	4.07807/3.97221
Mean observations per image	287.429/365.885	384.226/382.419/387.393	359.708/384.667	334.704/402.37
Mean reprojection error	0.579026/0.582044	0.527869/0.536371/0.524336	0.595617/0.575261	0.571875/0.583886
Running Time Matching	4.308 [minutes]	4.452 [minutes]	4.42 [minutes]	4.5 [minutes]
Running Time Reconstruction	0.696/1.136	1.256/1.190/0.848 [minutes]	1.22/1.328 [minutes]	0.721/0.858 [minutes]

Figure 5.1: Handling COLMAP runtime issues

A baseline evaluation for COLMAP for preliminary study, based on which decisions and adjustments were carried out. Experiments demonstrated on 36 dataset images picked with rotation consistency

## 5.2 Remove pairs using CNISO

### 5.2.1 Vocab Tree geometric verification

Initial evaluation for baseline feature matching stage of Vocab Tree (including geometric verification):

1. Feature matching baselines with no filtering  $\rightarrow$  Exhaustive v.s. Full Vocab Tree

with RANSAC minimum number of inliers count of 15 and 30. Running time comparison and Reconstruction quality results comparison are demonstrated In Figure 5.2. Results seems slightly better for 30 minimum inliers demand according to all criterions: number of cameras registered, mean re-projection error, and larger number of computed 3D points. Feature matching running time is almost not affected in COLMAP, even though RANSAC is avoided for images that doesn't have enough matching (inliers + outliers) beforehand as input for RANSAC. It turns out that in our testing RANSAC running time is only 1% of feature matching process which is slow on COLMAP. Major time consumption in COLMAP is by the matching performed and this fact is related to the non-CUDA GPU limitation issue.

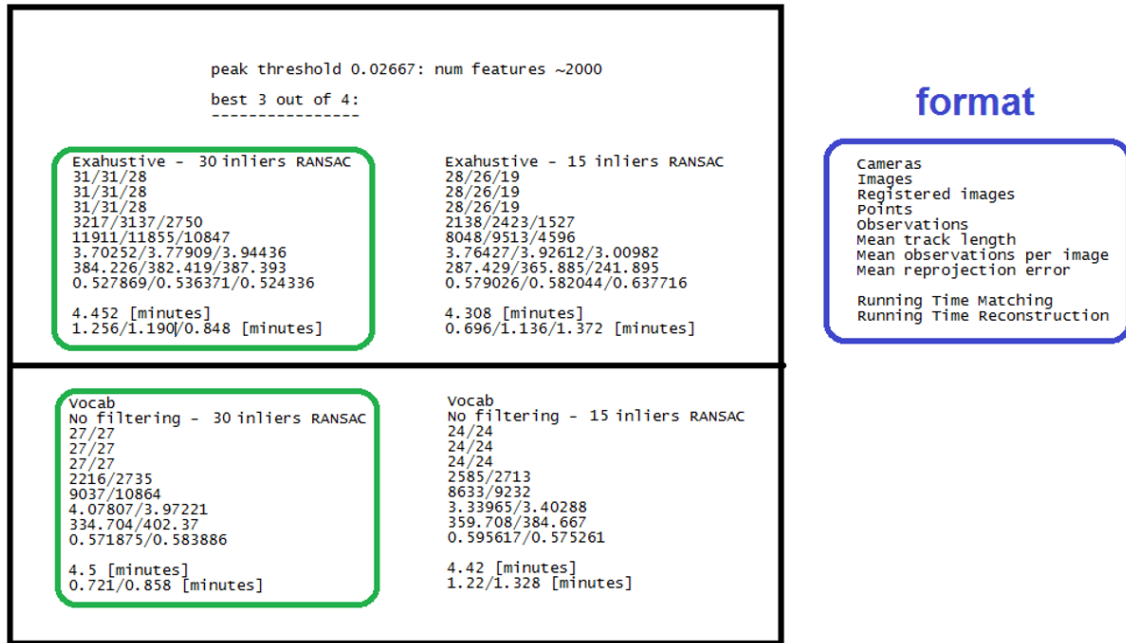


Figure 5.2: Vocab Tree Baseline Evaluation

Focusing on number of inliers and noticing running time is surprisingly unchanged for Feature Matching phase (due to non-CUDA GPU manner) and quality is slightly better

2. Feature Matching → Vocab Tree: At first, applied with k=10 on 36 images. VAV v.s. CNISO v.s. Full Vocab Tree (Baseline) running time comparison, reconstruction quality results comparison, and RANSAC failures count comparison, are shown in Figure 5.3 5.4. For K=3 results are ambiguous and hard to compare, However for K=10 it becomes clear. CNISO is slightly bet-

ter in running time of scene graph phase, but VAV performs better in building the scene graph, resulting in both higher accuracy and better running time of reconstruction phase.

peak threshold 0.01667: num features ~4000-5000		
average over 3 times:		
	Vocab, K=3	Vocab, K=3
	CNISO - 15 inliers RANSAC	VAV - 15 inliers RANSAC
Cameras	26	29
Images	26	29
Registered images	26	29
Points	4000	5000
Observations	14500	17500
Mean track length	3.6	3.5
Mean observations per image	530	580
Mean reprojection error	0.42	0.4
Running Time Matching	10 [minutes]	10 [minutes]
Running Time Reconstruction	0.6 [minutes]	0.8 [minutes]

peak threshold 0.02667: num features ~2000		
best 3 out of 4:		
	Vocab, K=3	Vocab, K=3
	CNISO - 15 inliers RANSAC	VAV - 15 inliers RANSAC
Cameras	15/16/24/25	14/14/14
Images	15/16/24/25	14/14/14
Registered images	15/16/24/25	14/14/14
Points	851/1110/1389/1600	850/855/1100
Observations	3058/3520/5058	3135/3148/3300
Mean track length	3.59342/3.17117/3.64147	3.67958/3.50505/3.55
Mean observations per image	204/220 /210	224/304.8/300.5
Mean reprojection error	0.421248/0.391198/0.399646/0.39	0.433481/0.431/0.414
Running Time Matching	0.713/0.733/0.718/0.72[minutes]	0.736/0.739/0.748 [minutes]
Running Time Reconstruction	0.337/0.219/0.350/0.6[minutes]	0.240/0.447/0.550 [minutes]

Figure 5.3: Evaluation of Vocab Tree in COLMAP - CNISO against VAV Experiment for K=3, with 2000 and 4000 features.

	Vocab, K=10	Vocab, K=10
	CNISO - 15 inliers RANSAC	VAV - 15 inliers RANSAC
Cameras	24/27/23	27/27/27
Images	24/27/23	27/27/27
Registered images	24/27/23	27/27/27
Points	2032/2359/2180	2478/1619/2557
Observations	8022/8834/7880	10074/6763/9166
Mean track length	3.94783/3.74481/3.61468	4.06538/4.17727/3.58467
Mean observations per image	334.25/327.185/342.609	373.111/250.481/339.481
Mean reprojection error	0.538054/0.653009/0.588337	0.525754/0.524831/0.534898
Running Time Matching	1.816 [minutes]	1.863 [minutes]
Running Time Reconstruction	1.629/1.932/1.988 [minutes]	0.869/1.478/0.608 [minutes]
RANSAC failures	26 out of 162-324 at most	20 out of 162-324 at most

\*\*\* Therefore, difference RANSAC failures at percentage is at least  $100 \cdot (26-20)/324 = 1.8\%$ , and at most  $3.6\%$  ( $2.5\%$  approximately). Apparently, this is enough to cause such effect in results.

Figure 5.4: Evaluation of Vocab Tree in COLMAP - CNISO against VAV Experiment for K=10 with 2000 features. 36 images dataset

3. Feature Matching → Exhaustive: At first, done for 36 images. CNISO v.s. Exhaustive (Baseline).

Running time comparison and reconstruction quality results comparison are demonstrated in Figure 5.5. Running NN Lowe's ratio is the bottleneck of the COLMAP system in our project configuration. RANSAC are done extremely

fast compared to NN:  $(3.45-3.35)/3.45 \simeq 2.85\%$ .

Even if CNISO is performed within zero-time and avoid all RANSAC operations, a 2.89% running time improvement will be produced at best. Running CNISO introduces running time improvement which is lower than 1%, and with no quality improvement, as mentioned earlier on results of 36 database images.

So to sum it up, here, with no CUDA GPU capability CNISO is meaningless since it offers a bit of deterioration in quality with no runtime improvements. However, if possible in the future to run with CUDA GPU, it would be legitimate to examine the trade-off between running time improvement to quality deterioration of CNISO against VAV. Here we skip this possibility since it is important to us to produce higher quality in this project.

### 5.2.2 Exhaustive geometric verification

Here we evaluate exhaustive feature matching stage including the geometric verification. At first a set of 250 images out of nearly 500 images were randomly chosen and ran for the project research. When comparing between the results of exhaustive with and without CNISO and the Vocabulary tree of COLMAP vote and verify approach, the following is the outcome.

Results are demonstrated on table in Figure 5.6.

For 250 images the main model reconstructed with CNISO, has around 12700 3D points for 63 images, where the main model of original exhaustive has 13500 3D points for 76 images.

CNISO's re-projection error is better for all models, specifically for main model is 0.40 mean error compared to 0.43 for original exhaustive. Moreover, it introduced improvement in running time in both feature matching stage (5 times faster in RANSAC verification stage, introducing 80% time save) and reconstruction stage (performs around 20% faster).

There are two main reasons for that. The first one is that CNISO filter image

1280X720 image size ~2000 features unless mentioned otherwise, RANSAC thresh is 15 inliers n = 36 image dataset					
features matching possibility	NN Lowe's ration	CNISO	RANSAC	runtime [minutes]	note
option 2 all images are filtered	$O(n^2)$	$O(n^2)$	X	3.431 3.5 3.301	
option 2 all images are filtered No CNISO (... = option 5)	$O(n^2)$	X	X	3.374 3.318 3.298	
option 2 CNISO and RANSAC thresh are the same = 70	$O(n^2)$	$O(n^2)$	selective	3.314	Some needless RANSACs. for different thresh all RANSAC are necessary (70 Ng thresh / 15 RANSAC thresh)
BASELINE (Exhaustive)	$O(n^2)$	X	$O(n^2)$	3.502 3.479 3.339	

Figure 5.5: Evaluate Combinations CNISO RANSAC.

Options:

0 = No CNISO (CNISO disabled)

1 = CNISO with VocabTree

2 = CNISO with Exhaustive in two\_view\_geometry→Run(). Same threshold as RANSAC min\_num\_inliers.

3 = option 2 + erase matches. Same threshold as RANSAC min\_num\_inliers.

4 = CNISO with Exhaustive in siftCPUFeatureMatcher→Run() by erasing matches. Different threshold from RANSAC min\_num\_inliers - 30.

5 = Exhaustive when in two\_view\_geometry →Run() rejecting all pairs.

peak threshold 0.02667: num features ~2000					
n=250 images					
best 3 out of 4:					
	Original Exhaustive 15 inliers RANSAC	Exhaustive + CNISO 15 inliers RANSAC 15 inliers CNISO's NG thresh	Exhaustive + CNISO 15 inliers RANSAC 15 inliers CNISO's NG thresh	notes	Vocab, K=11 (i.e. 10) VAV - 15 inliers
	Original Dataset & manual intervention for consistency	Original Dataset	Dataset after manual intervention for consistency		
Cameras	76	63	72		39
Images	76	63	72		39
Registered images	76	63	72		39
Points	13500	12700	13100		11300
Observations	51000	47000	49000		42000
Mean track length	3.75	3.76	3.74		3.72
Mean observations per image	670	750	890		1070
Mean reprojection error	0.43	0.4	0.42		0.46
Running Time NN	don't care - 12 hours	don't care - 12 hours	don't care - 12 hours	same for all	72 [minutes] = 1.2 [hours]
Running Time Matching	0.69 [minutes]	0.15 [minutes]	0.15 [minutes]		7 [minutes]
Running Time Reconstruction	~6.5 [minutes]	~5 minutes	~5.5 [minutes]		26 out of 162-324 at most

Figure 5.6: Evaluate Exhaustive

pairs with weak correspondence, and consequently less images are needed to perform a more accurate and robust reconstruction in terms of re-projection error, and in

less time, either in matching and reconstruction. The additional images used by original exhaustive produce only marginal expansion for the number of 3D points with a small deterioration in re-projection error. The second reason is apparent when considering the image pairs ignored according to CNISO’s filtering. It turns out that since images are taken from all poses, including rotation of 90 degrees, than some image pairs are inconsistent with x, y axes and therefore might be neglected by mistake, resulting in false negative matching (don’t match result when should match). false negative/positive measures of an image pair verification process such as CNISO/VAV are explained in Appendix A

To ignore this effect (which can also automatically detected), the images in the 250 set were manually rotated in order for all images axes to be consistent in direction for the entire set of 250 images set.

As we can conclude from results in Figure 5.6, there were some false negatives that are now taken into consideration. Examining carefully the images from DB of COLMAP the number of additional image pairs was insignificantly enlarged for the given dataset (1 image pair out of 10 approximately for the given set, relative to number of slaved images rotated). On the one hand, there is only little change for the set, effectively not changing running time of RANSAC stage in COLMAP feature matching. However, it seems the results of the main model constructed has improved in the sense of number of images ( 72 instead of 63) and number of 3D points, with relatively small increase in running time and in mean re-projection error (0.42 instead of 0.4) compared to CNISO exhaustive on the original dataset with no manual intervention. The results are now getting more similar to the original exhaustive with no CNISO in terms of number of images and error – which means that we were able to reduce running-time without deterioration to the model reconstructed. Relative to CNISO exhaustive on original dataset (no manual rotation), it seems that the images added in this experiment are again images with strong correspondence indeed and therefore compared to the results of original exhaustive are better in context of re-projection error. Still, mostly images with low-correspondence were

rejected and running time improved. At the worst case all images of original Exhaustive will be taken into consideration (high and low correspondence image pairs that were qualified according to RANSAC), in which case the results will be the same with original exhaustive and still expensive running time of RANSAC failures will be saved.

It's clearer now how CNISO improved the results of the pipeline both in running time and construction results. Hence, it might be legitimate to think of running CNISO again with  $x$  and  $y$  or add a fast processing of images direction difference or consistency test before running CNISO. This adds up running time issue and it is highly legitimate to perform rotation of images by human interference.

Now, when comparing the exhaustive approach with CNISO to the Vocabulary tree of Vote-And-verify, in terms of quality, the approach of CNISO achieves better results in measures of number of images registered, number of 3D points reconstructed and mean re-projection error.

To sum it up, there are 3 main approaches that can be adopted. The exhaustive approach known to give best outcome but is way too slow, and two fast approaches. First one is Exhaustive with CNISO's approach of running over NN matches and saving time on RANSAC's and reconstruction. The second one is the Vote-And-Verify approach that run it's test over the vocabulary tree matching and focuses on replacing NN matching with a lighter matching for better focus of NN matching and saving in-vain RANSAC's.

#### **5.2.2.1 Estimating Runtime improvements for CUDA**

When running COLMAP exhaustive matching without CUDA for 250 images dataset, as done along the project, NN running time was around 12 hours, which are equal to 720 minutes. Trying to figure out how much time it would have taken with CUDA capability enabled, can be done using some known documented running time combined with our results.

It is given that in COLMAP runtime for RANSAC only and reconstruction



phase are not supposed to enhance with CUDA capability than we can use it for our estimation. In addition it is documented in COLMAP that total running time for SFM pipeline for around 10000 pictures is around 5 days at worst for around 2000-3000 features (documentation might refer to 10000 features).

Therefore, using the known growth rate from literature for each sub-phase, we can compute a coarse running time approximation as follows in Figure 5.7.

<b>legend:</b>					
input					
calculation					

num images \ time param	250	1000	5000	10000	growth rate
NN	3-4 [m]	1 [h]	1 [days]	4 [Days]	$O(n^2)$
RANSAC's NN	0.69 [m]	12 [m]	5 [h]	20 [h]	$O(n^2)$
Reconstruction	5.5 [m]	22 [m]	2 [h]	4 [h]	$O(n)$
				5 [Days]	

Figure 5.7: Estimate Running Time With CUDA

Short explanation: At first, running time for Reconstruction and RANSAC's NN are computed for 10000 images. Afterwards, the 10000 images running time for NN with CUDA can be estimated using the total time of 5 days documented. Then, running time for NN phase with CUDA for 250 images can be calculated backwards to 250 images. So, at the worst case we estimate that running time for 250 images, with CUDA capability, would decrease from 4.5 minutes without running CNISO's test at exhaustive, to 4 minutes when running with CNISO's test at exhaustive matching.

### 5.2.2.2 COLMAP Data Base useful Tip

**Specifically for exhaustive matching** all code changes and editions were coded after NN Lowe's ratio matching were executed and matches were saved on DB with proper user access. Therefore, for evaluating running time it is possible to run exhaustive matching once for the purpose of saving raw matches + inlier matches, and then delete only inlier matches each time before running. In this technique we avoid NN heavy computation caused due to the lack of CUDA GPU.

Thus, evaluation of CNISO on all pairs + verified RANSAC on CNISO's pairs (deleting inliers only and KL1,2) running time will be compared to running all RANSAC pairs (deleting inliers only and running exhaustive).

**On the contrary, in vocab tree matching** filtering is done according to vocab tree matching which is not saved on DB. Moreover, NN Lowe's ratio is dependent on verified CNISO/Vote-And-Verify, so evaluating only this stage is an incorrect measurement comparison.

This is emphasized in Figure 5.8

## 5.3 RANSAC Halting Conditions using CNISO

Halting conditions for RANSAC using CNISO. Was performed and evaluated on the exhaustive pipeline. According to CNISO's article it is advised to use  $N_G$  estimation for RANSAC's stop condition. Stopping condition for RANSAC was added in COLMAP, by calling CNISO inside RANSAC and stopping the process when the inlier set size reached the computed  $N_G$ . This was done in addition to the standard and known stopping condition from literature, so the earlier condition wins and stops the RANSAC process.

This was tested on real world 250 image dataset, and it was demonstrated that runtime execution of RANSAC reduced RANSAC running time from 0.66 minutes to 0.415 minutes, without damaging neither quality nor runtime result of the SFM process.

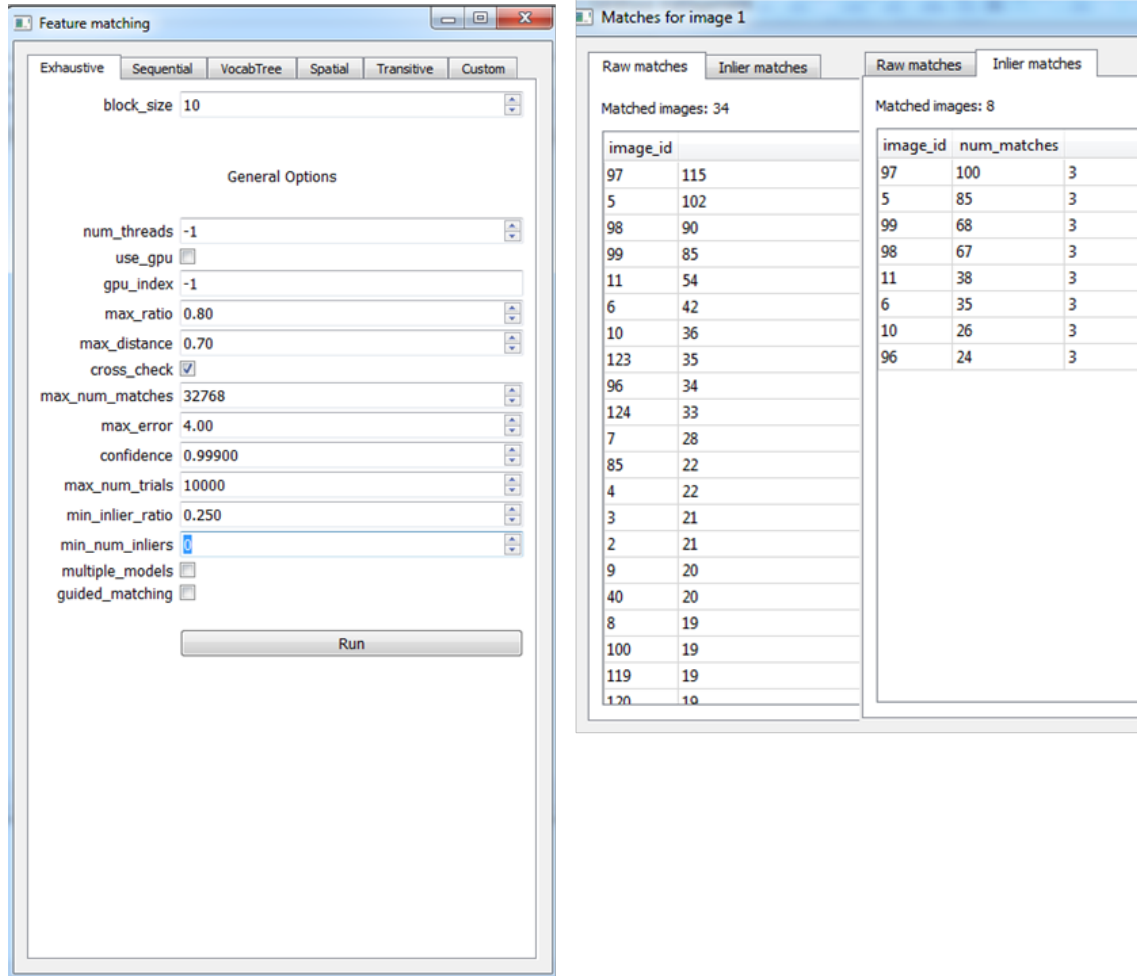


Figure 5.8: DB COLMAP Raw matches and Inliers matches

When looking into COLMAP database there are some little differences in the inliers results between the two approaches. For a lot of pairs the RANSAC process with CNISO stops earlier, and consequently some of these pairs end up with less inliers. RANSAC randomness usually expressed in changes around 2-3 points in inliers set size. For around 10-15% image pairs inliers set size is changed in up to 10 points (up to 10% inliers set size for strong pairs candidates), not because of the RANSAC randomness but indeed due to the  $N_G$  stopping condition added. These effects and behaviour are shown in Figure 5.9.

However, as shown in Figure 5.10, this doesn't damage the reconstruction phase. Strongly connected image pairs have higher  $N_G$  and therefore stay on top and order for the incremental process is similar before and after the change. In addition, the initial fundamental matrices and matched points composed out of the faster

RANSAC process are still very good estimation, in particular good enough for the rest of the process.

To sum it up, results are approximately the same, with runtime save of around 38% in RANSAC process when adding CNISO's test to RANSAC halting condition.

**before and after ransac change for CNISO**

**before**

image_id	num_matches	
97	99	3
9	86	3
99	68	3
100	68	3
8	38	3
7	36	3
5	30	3
124	30	3
96	24	3
123	24	6
6	23	3

**after**

image_id	num_matches	
97	100	3
9	74	3
100	68	3
99	66	3
8	38	3
7	30	3
96	24	3
124	24	6
5	23	3
123	22	6
121	15	3

Figure 5.9: DB before And After CNISO

## 5.4 Combine Pairs Remove and RANSAC Halt

Here a combination of the two earlier experiments took place on exhaustive matching pipeline. Hence, performing CNISO test for removing pairs, and in addition halting RANSAC's loop if the set size has reached estimated inliers count according to CNISO.

In Figure 5.11 the experiment results are demonstrated. It is shown that this experiment achieved running time save of about 15 percent in relation to the approach in which we ran this experiment filtering pairs only. This was granted without compromising the quality of reconstruction.

Curve improvement of RANSAC's halting condition here is lower than the previous experiment of Section 5.3, in which we identified an improvement of about

peak threshold 0.02667: num features ~2000		
	Original Exhaustive	Original Exhaustive + CNISO in RAN SAC
	15 inliers RAN SAC	15 inliers RAN SAC
	Dataset after manual rotation intervention	Dataset after manual rotation intervention
Cameras	76	75
Images	76	75
Registered images	76	75
Points	13500	13200
Observations	51000	49500
Mean track length	3.75	3.75
Mean observations per image	670	660
Mean reprojection error	0.43	0.42
Running Time NN	don't care - 12 hours	don't care - 12 hours
Running Time Matching	0.69 [minutes]	0.43 [minutes]
Running Time Reconstruction	~6.5 [minutes]	~6.5 [minutes]

Figure 5.10: COLMAP with CNISO in RANSAC condition

COLMAP experimental result of CNISO as an additional halting condition for RANSAC. Running time improvement with only noisy and insignificant deterioration for SFM quality performance

40 percent due to the addition of the stop condition. This is since on this experiment there is indeed strong dependency between the pairs that are filtered on the two runs. On the previous experiment all weak image pairs according to CNISO (pairs that correspond to low  $N_G$ , in particular lower than RANSAC's threshold) were stopped much earlier on RANSAC's loop thanks to CNISO's RANSAC halting condition added. On the contrary, in the current experiment a major part of these weak pairs were filtered earlier on the pipeline, therefore time saved on previous experiment was not saved here.

## 5.5 Runtime Measurement for CNISO on COLMAP

Runtime of CNISO was measured by running COLMAP exhaustive matching approach with around 3000 features per image. Note that in order to avoid irrelevant calls, CNISO was called after all other condition checks in TwoViewGeometryVer-

peak threshold 0.02667: num features ~ 2000		
	Exhaustive + CNISO pairs removal	Exhaustive + CNISO in RAN SAC + CNISO pairs removal
	15 inliers RAN SAC 15 inliers CNISO's NG thresh	15 inliers RAN SAC 15 inliers CNISO's NG thresh
	Dataset after manual rotation intervention	Dataset after manual rotation intervention
Cameras	72	72
Images	72	72
Registered images	72	72
Points	13100	13100
Observations	49000	49000
Mean track length	3.74	3.74
Mean observations per image	690	690
Mean reprojection error	0.42	0.42
Running Time NN	don't care - 12 hours	don't care - 12 hours
Running Time Matching	0.15 [minutes]	0.12 [minutes]
Running Time Reconstruction	~5.5 [minutes]	~5.5 [minutes]

Figure 5.11: CNISO used both on pairs removal and RANSAC's halting condition

ifier::Run, i.e. right before calling the RANSAC estimation process. This is also drawn on the data-flow diagram in Figure 4.10.

In an experiment for 250 image dataset, there are 31125 possible pairs, out of which 4306 calls for CNISO were performed, out of which 1309 calls for RANSAC were performed. The rest of image pairs were ignored since it had less matches than the number of minimal inliers. We recall that this process took around 9 seconds. For 9 seconds runtime measured, under a severe (and irrational) assumption of zero RANSAC running time, this assures that real world behaviours of CNISO in COLMAP is much less than 2 ms for an average candidate pair.

# Chapter 6

## Summary and Future Work

We implemented the algorithm of CNISO in C++ for SFM open sources. We integrated it successfully in COLMAP, and performed experiments to test the efficiency and quality improvements of CNISO in COLMAP's SFM pipeline. The integration took place in feature matching stage in image pairs removal and/or RANSAC halting condition.

The results indicate that using CNISO for RANSAC's halting condition establishes around 40% time saving in RANSAC's operations and with no change to quality of exhaustive matching performance. In addition, results of pairs removal introduce three main approaches: (i) Exhaustive matching (ii) Exhaustive matching with CNISO (iii) Vocabulary tree matching with VAV. Experiments show that approach (i) is of best quality but of highest running time as well - 4.5 minutes for 250 images; Approach (ii) is only of slightly lower quality compared to (i) but with better running time estimation of 4 minutes. approach (iii) is of low quality compared to both (i) and (ii) but introduce far faster running time as well.

As a future research, it is worth to run COLMAP with CUDA GPU and measure time savings of RANSAC's halting condition for the full feature matching process and not only RANSAC's operations as in project. More of the same, it is recommended to run and measure vocab tree with VAV as well and compare its running time to exhaustive matching running time. Another CUDA GPU measurement with potential added value is running exhaustive with and without CNISO, and compare

running time to the time measured in Figure 5.7.

Once done, it is worth to use CNISO's number of inliers as weights of scene graph and use overlapping sections to determine left and right images for incremental SFM order decisions. For this purpose it should be noted that for large scale data-sets with preliminary scene graph and matching, it seems unnecessary to use CUDA GPU since our experiments shows that the stage of BA and triangulation is scalable runs fast.

In addition, CNISO overlapping sections of image pairs can also be used for pairs removal in scene graph for example if conflicts are identified between multiple pairs.



# Bibliography

- [1] L. Talker, Y. Moses, and I. Shimshoni, “Estimating the number of correct matches using only spatial order,” *IEEE transactions on pattern analysis and machine intelligence*, 2018.
- [2] S. David, “CNISO SFM Project.” <https://github.com/shalevdavid/SpatialOrderSFM>, 2019.
- [3] L. Goshen and I. Shimshoni, “Balanced exploration and exploitation model search for efficient epipolar geometry estimation,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1230–1242, 2008.
- [4] R. Raguram, O. J. Chum, M. Pollefeys, J. Matas, and J. M. Frahm, “USAC: a universal framework for random sample consensus,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 35, no. 8, pp. 2022–2038, 2013.
- [5] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Comm. of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.
- [6] A. S. Brahmachari and S. Sarkar, “Hop-diffusion monte carlo for epipolar geometry estimation between very wide-baseline images,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 35, no. 3, pp. 755–762, 2013.
- [7] O. Chum and J. Matas, “Optimal randomized RANSAC,” *IEEE Trans. Patt. Anal. Mach. Intell.*, vol. 30, no. 8, pp. 1472–1482, 2008.
- [8] K. Wilson and N. Snavely, “Robust global translations with 1dsfm,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

- [9] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [10] M. G. Kendall, “A new measure of rank correlation,” *Biometrika*, pp. 81–93, 1938.
- [11] P. Diaconis and R. L. Graham, “Spearman’s footrule as a measure of disarray,” *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 262–268, 1977.
- [12] E. I. Unit, “Building rome in a day: The sustainability of china’s housing boom,” *The Economist Intelligence Unit Ltd. London (www. eiu. com)*, 2011.
- [13] N. Snavely, S. M. Seitz, and R. Szeliski, “Skeletal graphs for efficient structure from motion.,” in *CVPR*, 2008.
- [14] M. Farenzena, A. Fusiello, and R. Gherardi, “Structure-and-motion pipeline on a hierarchical cluster tree,” in *ICCV Workshop*, 2009.
- [15] C. Sweeney, T. Sattler, T. Hollerer, M. Turk, and M. Pollefeys, “Optimizing the viewing graph for structure-from-motion,” in *ICCV*, 2015.
- [16] S. Avidan, Y. Moses, and Y. Moses, “Centralized and distributed multi-view correspondence,” *International Journal of Computer Vision*, vol. 71, no. 1, pp. 49–69, 2007.
- [17] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4104–4113, 2016.
- [18] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, “Object retrieval with large vocabularies and fast spatial matching,” in *CVPR*, 2007.
- [19] D. Nister and H. Stewenius, “Scalable recognition with a vocabulary tree,” in *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, vol. 2, pp. 2161–2168, Ieee, 2006.
- [20] J. Ramos *et al.*, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the first instructional conference on machine learning*, vol. 242, pp. 133–142, 2003.

- [21] J. L. Schönberger, T. Price, T. Sattler, J.-M. Frahm, and M. Pollefeys, “A vote-and-verify strategy for fast spatial verification in image retrieval,” in *Asian Conference on Computer Vision*, pp. 321–337, Springer, 2016.
- [22] J. L. Schönberger, A. C. Berg, and J.-M. Frahm, “Efficient two-view geometry classification,” in *GCPR*, 2015.
- [23] J. L. Schönberger, A. C. Berg, and J.-M. Frahm, “Paige: pairwise image geometry encoding for improved efficiency in structure-from-motion,” in *CVPR*, 2015.
- [24] L. Talker, “CNISO matlab on github.” <https://github.com/liortalker/SpatialOrder>, 2018.
- [25] L. Talker, “Talker’s Site.” <http://liortalker.wixsite.com/liortalker>, 2018.

# Appendices

# Appendix A

## False Negative/Positive measures

Measuring and defining false negative and false positive of verification process: A formal measurement for evaluating CNISO can be its RANSAC prediction success. Depending on the fact that RANSAC is supposed be robust to inliers extraction out of a contaminated set, CNISO's  $N_G$  threshold is supposed to predict RANSAC outcome. Therefore false negative and false positive for CNISO's prediction can be defined as:

**False negative** – an image pair that didn't pass  $N_G$ 's threshold, and for RANSAC is successful (number of inliers according to RANSAC is higher than it's threshold). Meaning RANSAC was run unnecessarily.

**False positive** – an image pair that passed  $N_G$ 's threshold, but for which RANSAC was unsuccessful (number of inliers according to RANSAC is lower than its threshold). Meaning that this pair was rejected falsely.

Calculating false negative/positive of CNISO with COLMAP: First we run exhaustive matching **with CNISO's estimation (setting  $N_G$  thresh to some value)** after assigning RANSAC's minimum inliers threshold to 0 using the GUI. Thus, all results of RANSAC are displayed on the inliers matches DB.

Secondly, we run exhaustive matching with RANSAC. Advised to set RANSAC's minimum inliers threshold to 0 in order to be able to learn more about the thresholds ratio if needed.

We get the following picture of the two runs: On the first run we collect the outcome of CNISO. Since RANSAC threshold was inactive (since was set to 0), inlier matches reflect pairs that passed CNISO's test for a given  $N_G$  threshold, and all other matches (that appear on raw matches and not on inlier matches) reflect the pairs that didn't pass CNISO's test for a given  $N_G$  threshold.

On the second run for some RANSAC threshold (15 by default at COLMAP) we collect good and bad pairs. This is well demonstrated in Figure A.1.

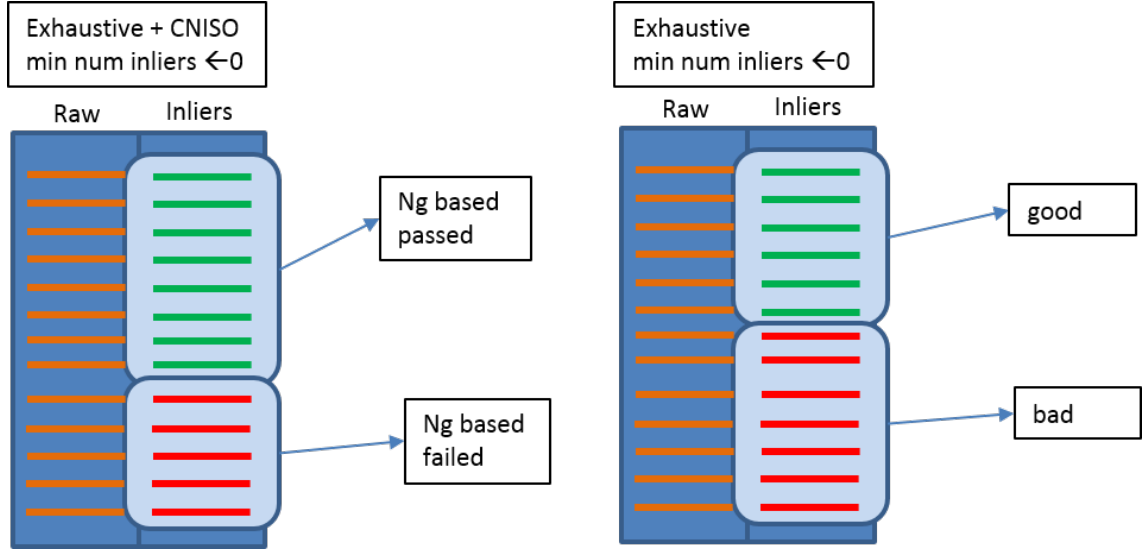


Figure A.1: False Positive/Negative Measures for image pair verification process

So computing false positive and false negative is can be done as follows: False positive count: We count how many image pairs **passed  $N_G$  threshold** and are **bad according to RANSAC**.

False positive rate = (number of pairs passed  $N_G$  thresh and didn't pass RANSAC thresh) / (number of pairs didn't pass RANSAC thresh).

False negative count: We count how many image pairs **didn't pass  $N_G$  threshold** and are **good according to RANSAC**.

False negative rate = (number of pairs didn't pass  $N_G$  thresh and passed RANSAC thresh) / (number of pairs passed RANSAC thresh).

Summing it up in Figure A.2.

Ng thresh ↑ RANSAC thresh constant	false positive ↓	false negative ↑
Ng thresh constant RANSAC thresh ↓		
Ng thresh ↓ RANSAC thresh constant	false positive ↑	false negative ↓
Ng thresh constant RANSAC thresh ↑		

Figure A.2: false Positive/Negative as a function of  $N_G$  and RANSAC thresholds.

# Appendix B

## COLMAP Dataflow Enrichment

For dataset of 250 images used in this project, every image has on average around 10-15 overlapping images in exhaustive matching with CNISO for pairs removal.

For large datasets of thousands and tens thousands images, there might be a phenomena of massive overlapping images (meaning that the sub-scene has massive multiple overlapping images) resulting in a huge scene graph. An approach to deal with this situation when assumption not fulfilled can be done using filtering only 10 or so prioritized matching pairs for each image (similar to done in COLMAP’s vocab tree and similar to Building Rome in a day paper overview).

COLMAP doesn’t support this immediately in pipeline since each candidate pair’s processes of matching and process of geometric verification (that include RANSAC) are ran each one in an isolated thread, blindfold to the entire picture. Thus, avoiding RANSAC for each pair automatically cannot be done without breaking the isolation in some way. It is needed to use some global, or run some code offline, or manipulate some basic classes and structures used transversely in the code.

In more details, if all threads for matching (in particular the function `SiftCPUFeatureMatcher::Run`) were ran before any thread of geometric verification (in particular the function `TwoViewGeometryVerifier::Run`), an opportunity to run CNISO right after Lowe’s ratio NN matching (such as which was developed and tested earlier for skipping a pair with low  $N_G$ ), combined with saving the result in a global array



without rejecting any pair yet, could have been used later on geometric verification for decision to reject pairs and avoid RANSAC based on the big picture. This should have been a natural implementation. **Unfortunately, this is not the case here.** After some research and debugging of COLMAP code it turns out that **threads for matching and threads for geometric verification (that include RANSAC) for different pairs are ran simultaneously.**

If relevant in future, there is an alternative for implementation in COLMAP. It will be suggested to run offline the entire processes of matching and CNISO (and optionally with or without RANSAC), and for each image save a priority list which reflect its 10 best overlapping images according to CNISO (highest  $N_{Gs}$ ) the decision that should have been taken on runtime. Afterwards run COLMAP again online and add a condition for running RANSAC according to the priority list. If necessary, for runtime measurements to be real and exact, it is possible to run CNISO online as well and avoid its output, and instead of reading the decision from a file saved on disc it can be saved internally in memory with some global array or alternatively loaded from file once on some user request added to COLMAP. Offline and online executions can be managed by the GUI. This is clumsy to develop and maintain, but definitely possible.

**Note:** When running for the first time in order to compute the global array decision, a Mutual exclusion and/or Semaphore protections might be needed for data race handling of readers-writers problem.

**Composing all of the above,** for 250 data-set images it is this will not improve results and not needed. For thousands and tens thousands images it is impractical in the project nowadays since no CUDA for NN Lowe's ratio is available (NVidia GPU Hardware + CUDA installation  $\rightarrow$  impractical issue). **Hence, not advised for now since might be relevant only for the project impractical issues.**

# תקציר

Structure From Motion הינו תהליך של כיול מצלמות ושחזור תלת ממד מתוך אוסף תמונות שצולמו מסצנה סטאטית. מערכות SFM הינן סבוכות ומורכבות משכבות רבות הכוללות אלגוריתמים מרכזיים מעולם הראייה הממוחשבת, כגון חילוץ מאפיינים, שיוך מאפיינים, RANSAC, Bundle Adjustments, טריאנגולציה וכדומה.

לאחרונה מערכות קוד פתוח של SFM מציעות הזדמנות לשחזור סצנה מתוך מאגר תמונות, כאשר האתגר המרכזי הוא התמודדות על מאגרים תמונות גדולים הכוללים אלפי ועשרות אלפי תמונות. לשם כך פותח אלגוריתם חדשני ויעיל לשערוך מס' השייכים הנכונים (מדד inliers) בין זוג תמונות. הקרויה CNISO המבוססת על העיקרון לפיו צפוי להישמר סדר מרחבי בין שייכים נכונים בתמונות. האלגוריתם מקבל כקלט תמונות עם matching ביניהן, אשר לא ידוע אם הן חופפות או לא ומוחזרת הערכה בנוגע למדד ה – inliers, סבירות לקיום החפיפה ואזור החפיפה ביניהן אם אכן קיים.

בפרויקט זה ניקח מערכת SFM ממקור קוד פתוח ונסה לשפר אותה באמצעות מימוש אלגוריתם CNISO ושילובו בזרימת המידע להשגת ביצועים טובים יותר בהיבטים של איכות וזמן ריצה. מערכת ה – SFM שנבחרה לעבודה בפרויקט זה היא מערכת ה – SFM החדשנית והצומחת COLMAP.

פרויקט זה תורם מימוש C++ עבור אלגוריתם CNISO ויישומה ב-COLMAP המפורסמים ב – Github עבור הקהילה. בנוסף, הפרויקט מספק מבט מעמיק על מערכת הקוד הפתוח COLMAP SFM ומציע שיפורים עבורה. בהקשר מערכת COLMAP הלימוד מציג טכניקות לביצוע פיתוח ומחקר בסביבה מוגבלת משאב חומרה של CUDA GPU.

עבודה זו בוצעה בהדרכת פרופ' יעל מוזס מבי"ס אפי ארזי למדעי המחשב, המרכז הבינתחומי, הרצליה.

המרכז הבינתחומי בהרצליה  
בית-ספר אפי ארזי למדעי המחשב  
התכנית לתואר שני (M.Sc.)

# שחזור מבנה וכיול מצלמות בעזרת שימור סדר מרחבי

מאת  
שלו דוד

פרויקט גמר, מוגש כחלק מהדרישות לשם קבלת תואר מוסמך M.Sc.,  
בית ספר אפי ארזי למדעי המחשב, המרכז הבינתחומי הרצליה

ינואר 2019