Semantic Sampling Localization: RGB-D Patch-Based Triplet Network

Marco Petri
Politecnico di Milano
marco.petri@mail.polimi.it

Mirko Usuelli
Politecnico di Milano
mirko.usuelli@mail.polimi.it

Abstract

This work analyses the Localization task of RGB-D images using a Deep Neural Network (DNN) tuned to improve the baseline performance of RANSAC by exploiting visual semantic information. We propose a DNN able to extract a semantic sampling distribution from paired key points to improve the Mean Average Accuracy (mAA), chosen as a reference metric. Next, the importance of the depth channel is shown by comparing the same DNN trained on RGB or RGB-D. Finally, Point Clouds are generated from paired images of the same scene, and Registration is performed to visualize the results in 3D space through the depth information.

1. Introduction

Before proceeding into the details of the proposed approach, it is needed to clarify the key concepts on which the method relies.

Epipolar Geometry It is the intrinsic projective geometry between two views. It is independent of scene structure, and it only depends on the cameras' internal parameters and relative pose [3].

Fundamental Matrix The Fundamental Matrix F encapsulates the Epipolar Geometry. It is a 3×3 matrix of rank 2. If a point in 3-space X is imaged as x in the first view, and x in the second, then the image points satisfy the relation $x'^T F x = 0$ [3].

Localization From a Computer Vision perspective, it stands for the task of estimating the camera pose concerning a given reference frame within the same spatial domain. By encoding the camera pose in terms of the Roto-Translation matrix; while the reference frame is established by the Identity Matrix, the following poses incrementally differ from it.

RANSAC It is a model-fitting algorithm that can also be employed to perform Localization. When we have images of the same scene, we can use RANSAC to estimate the Roto-Translation between the cameras by exploiting key point features. However, RANSAC is a probabilistic approach relying on a uniform sampling distribution that randomly selects the samples to be used to build up the model. The pure randomness of the approach makes it efficient up to a certain point; many improved guided-sampling versions such as P-NAPSAC [1] or PROSAC [2] have been introduced in recent years.

In this document, we propose our guided-sampling version of RANSAC.

2. Proposed approach

The proposed methodology has been conceived for the purpose of improving the RANSAC baseline by taking into account the visual paired-semantic information of two images referring to the same scene; as a result, a weighted sampling distribution is produced to build up the model.

2.1. Workflow pipeline

While the standard Localization approach straightforwardly takes place by applying RANSAC right after the detection-matching computation (Figure 1), our technique is structured as follows:

Loading The pipeline consists in taking as reference two different images of the same scene having dissimilar camera poses concerning each other.

Detection It is independently applied all over the two images to obtain the relevant features which need to be correctly paired; this workflow step can be done through several techniques. We have tested it with two common methods: ORB [7] and SIFT [6].

Matching Once the two images have their key points and descriptors, we apply a simple k-Nearest-Neighbour proce-

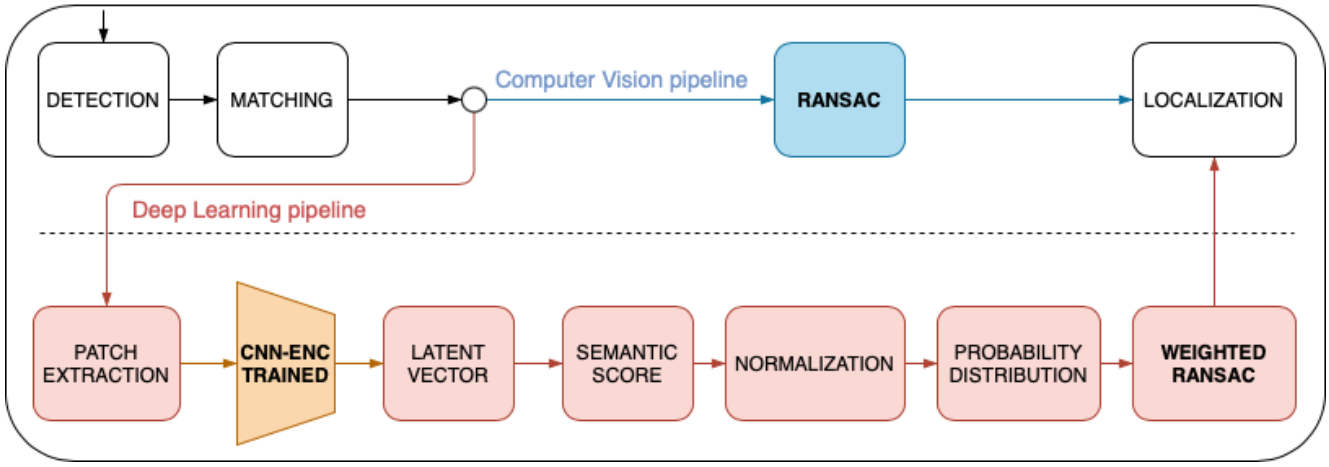

Figure 1. Localization with classical CV and DNN network to improve RANSAC.

dure in order to filter as much as possible the wrong matches and to get a list of them that is assumed to contain both inliers and outliers matches from a semantic point of view. This list is going to be furtherly improved along the pipeline procedure.

Patch Extraction We process all the involved key points in those matches contained in the list we have found in the previous step; afterward, we extract their relative patches around their key points' coordinates through all of their channels (RGB-D or RGB).

Trained CNN Encoder We use each extracted patch - namely each key point involved in a match - as the input of a trained CNN encoder which is going to be described in the details in section 2.2.

Latent Vector The output of the CNN encoder is a latent vector representation of the given patch, meaning that the image section has been compressed in a smaller numerical format domain entailing solid semantic information.

Semantic Score After having obtained a semantic latent representation for each key point involved in a match through their RGB-D patches, we compute — for each match — a semantic score by applying a similarity function to the two latent vectors belonging to the two key points of the match. This measure gives us an estimation of how semantics related are those two key points that have been matched. By default, we decided to use the Euclidean Distance as similarity function.

Normalization As all the matches have been ranked through their semantic score, we need to take into account the fact of magnitudes variety along all of them. Hence — to fairly distinguish the semantic score weights concerning each other — we normalize them with a standard MinMaxScaler.

Probability distribution Later on — since RANSAC is based on uniform sampling distribution — our goal is to reduce as much as possible this stochasticity factor and focus on a faster and more robust consensus through the semantic scores we have computed. Then, we furtherly divide the whole sequences of semantic scores with their summation for the purpose of having a sort of probability distribution along the current matches.

Furthermore, we have noticed that this step drastically improves when we discard those matches below the 60th percentile of the distribution. However, this outcome suggests to us that the semantic ranking filter boosts the overall performance if and only if we assume a priori a relevant presence of outliers; otherwise, this assumption can be discarded or managed through a hyperparameter search on the percentile distribution, based on the data to deal with on validation.

Weighted RANSAC While the classical RANSAC approach relies on a uniform sampling procedure that establishes the randomness, our variation is guided through the above-mentioned semantic probability distribution, which is non-uniform.

Localization Finally, we obtain the model with higher consensus over the entire pipeline which will be exploited to fit the Fundamental Matrix through the weighted RANSAC algorithm. Once the Fundamental Matrix is computed, we multiply it with the Calibration Matrix to obtain the Essen-

tial Matrix; at the end of the workflow, we get the Rotation Matrix and the Translation Vector of the two images concerning each other.

2.2. Training procedure

Data loader Data loader is the core component for training the Triplet Network. It is in charge of retrieving the i^{th} input of the network when called. It is mainly characterized by two different values: n_{shift} and k. The former parameter describes how many images are in between the two selected images, the latter describes the range domain (positive or negative jump) from which the second image is taken as shown in Figure 2. Within the interval of all possible images described by k, one image is taken randomly. This procedure will be done when the object is created and a dictionary of this structure is built for purpose of getting images faster at training time.

Pose pre-processing Once the two frames have been arbitrarily chosen and loaded by the Data Loader, their ground-truth pose is read as well. This *supervised* information is exploited for purpose of estimating the Fundamental Matrix from the two quaternion poses. Then, this matrix will help us to choose the triplet samples through the Epipolar Geometry constraints entailed by it.

Features detection Being the training process heavy to be managed, we decided to deal with it by selecting just 32 features for each frame through the detection phase. Afterward, the first frame acts as the reference frame of the Fundamental Matrix, whereas the other one will be the target for the following sampling process.

Triplet sampling The triplets used for the training are sampled in a semi-stochastic fashion. Starting from the arrays of all the key-points in both images we select the reference image and for each key-point - **anchor** samples $\hat{A}$ - of this image, we find the **positive** $\hat{P}$ and **negative** $\hat{N}$ samples for the triplet. The positive sample is computed deterministically, while the negative sample is chosen stochastically using a seed for random number generation to allow reproducibility between pieces of training. The triplet equations relying on the anchor $\hat{A}$ are computed as:

$$\hat{P} = \arg\min_{x} \{\hat{A}^T F x\} \tag{1}$$

$$\hat{N} = \{x : \hat{A}^T F x \in D_t\} \setminus \{\hat{P}\} \tag{2}$$

Where $\hat{A}$ is the anchor, F is the fundamental matrix, and D_t is the set of Epipolar Lines entailed. Differently, the negative sample is computed by continuously picking random key points of the second image. If the randomly drawn key point is not in the meaningful region of $\hat{A}$, it will be used as a negative example; otherwise, we continue. This operation aims to improve training by providing the network with a negative example any example that is not positive, which may be similar or extremely different.

Patch Extraction After a triplet has been sampled, patches must be extracted. From a list of coordinates of the key points representing the anchor, positive and negative samples, patches from the RGB-D images will be extracted. Depending on the patch dimension (which we always take as $odd \times odd$, e.g. $odd = 17$), a patch will be extracted.

Triplet Network Based on the extracted patches related to the anchor, positive and negative samples — respectively $\hat{A}, \hat{P}, \hat{N}$ — we feed three Convolutional Neural Network encoders using the same weights and biases for each of them, namely in share weights mode. Triplet Loss tries to ensure a margin between distances of negative pairs and distances of positive pairs at the same time, furthermore, it allows to stretch clusters in such a way as to include outliers while still ensuring a margin between samples from different clusters, e.g., negative pairs [8].

Encoder The encoder consists just of convolutional layers to maintain a certain architectural simplicity concerning the small size of the patch. As shown in Figure 3, an RGB-D patch (4 channels) is taken as the encoder input, and an output, a latent vector is provided as the result of the convolutional compression.

Similarity function The similarity function used during the training is the Euclidean Distance measured both with $anchor - positive$ and $anchor - negative$ paired samples.

$$d_{\hat{P}} = simil(l_{\hat{A}}, l_{\hat{P}}) \tag{3}$$

$$d_{\hat{N}} = simil(l_{\hat{A}}, l_{\hat{N}}) \tag{4}$$

We also tried to use the Cosine Distance, however, the final result was worse than the former one, thus we decided to discard it.

Triplet Loss To backpropagate the whole computation done until now, we exploit the characteristic triplet loss function which gives the name to the adopted architecture. As shown by its formula, it tries to maximize the positive similarities and minimize the negative ones at the same time; the α is a regularization hyper-parameter that has been tuned:

$$L = d_{\hat{P}} - d_{\hat{N}} - \alpha \tag{5}$$

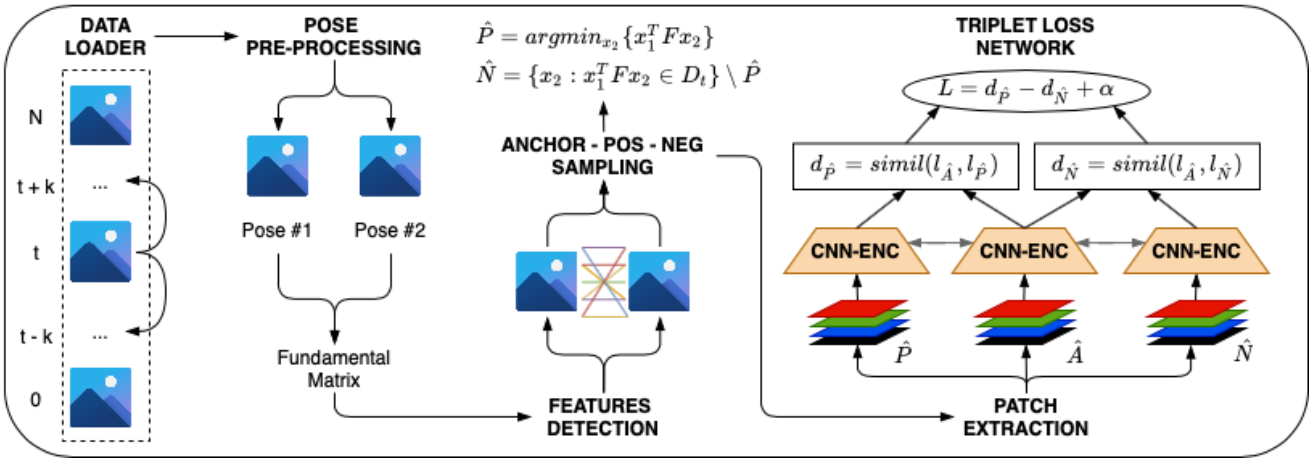

Figure 2. Training of the triplet network used to increase RANSAC mAA.

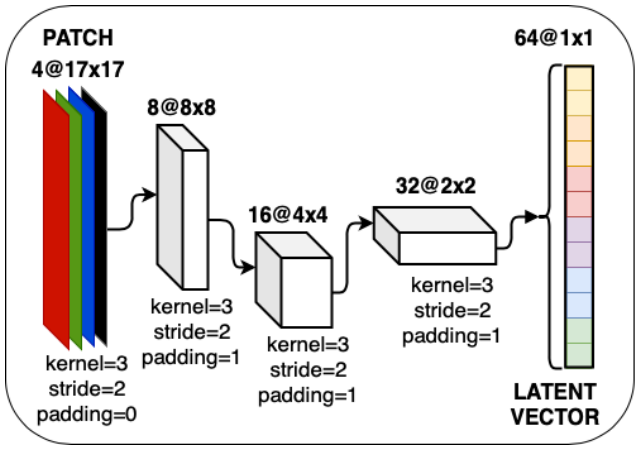

Figure 3. Encoder architecture taking as input a patch 4@17x17, and returning as output a (64, 1) size vector representing the semantic latent space.

2.3. RANSAC Accuracy

Metrics. To be able to evaluate the goodness of a model, [4] proposes the usage of mean Average Accuracy (mAA). It is a metric considering the ratio between correct computed roto-translations and all roto-translations computed within an angle error τ. If either the angle between the true and computed translation or one of the differences between true and computed Euler angles exceeds τ, the computation is considered wrong.

We compute the RANSAC accuracy of both classical RANSAC and DNN-based RANSAC using mAA. To do so, there are some important parameters to be taken into account: the angle threshold τ, the distance between the frames d, the RANSAC iterations r_{iter} and the parameters of the detection. With this metric, the comparison between classical and DNN-based RANSAC must be considered at varying d with constant τ for several values of τ. Moreover, each evaluation follows the general pipeline introduced in the previous sections.

3. Point clouds

When we have RGB-D images (images enhanced with the depth information for each pixel), we can build the so-called point cloud. It is a set of 3D points representing the scene that could be visualized with 3D visualization tools. However, if we have several images of the same scene, we would like to be able to perform what is called **point cloud registration**, i.e., the task of aligning two different point clouds to make them jointly represent the scene. The task is important because due to noise and alignment we need mainly two elements: the roto-translation between the two images (the cameras taking the image) and an adjustment factor due to noise. Images must be printed out in the same reference system while being expressed in a reference system dependent on the camera orientation. Furthermore, the noise will make point clouds not aligned even when we have the roto-translation between the two cameras, which means that we need to slightly adjust the previously obtained point clouds.

To perform the point cloud registration task we have tried different approaches:

- Ground truth registration: it consists in using the ground truth to align the point clouds. It means we only consider the roto-translation between the images.
- Fit roto-translation directly in 3D: we create a 3D model of points from RGB-D images. Then, we find the best 3D roto-translation aligning the two sets of points.

- Iterative Closest Point: we apply Iterative Point Cloud to perform registration.

In the end, the method that has revealed to obtain the best results was ICP. However, each method revealed scarce capability in registering images with extremely different orientations. With extremely different orientations, we mean images that have in common very few points, such as an image of the front of an object and another of its back. Such images are extremely complex to register with every approach.

4. Experiments

We conducted experiments on both RGB-D and RGB/Grayscale image datasets. To evaluate if semantic information could improve the performance of RANSAC, we implemented both the classical computer vision RANSAC algorithm and a version of RANSAC allowing weighting which will be given from the Triplet network. Then, for each configuration and each tested image, we use the same parameters with the only difference of probabilistic weighting for the DNN approach. In such a way, we could compare the mAA of DNN RANSAC and classical RANSAC. Then, we evaluated the metric also on a different dataset (that is [9]) of outdoor scenes (instead of indoor ones) without the depth channel. In such a way, we can evaluate if the DNN approach can exploit semantic information also on completely new and unseen domains (such as Notre Dame cathedral).

Datasets. We used mainly [5] as a dataset in the experiments of this project activity, taken through a KinectV1-like camera. It consists of 14 different **indoor** scenes composed of several images from different points of view. Regarding the images of scenes, the pose of each image is given using quaternions to represent rotations and 3D vectors to represent translations concerning the first image of the scene. Among all the scenes, we used one testing scene (scene 2), three validation scenes (1, 5, 9), and all the remaining scenes as training for the network. It is important to stress the fact that the choice of the testing and validation scenes was made for purpose of having completely unseen scenes for the training process. The testing scene is used to compute the mAA as well as to print results of both DNN-based and CV RANSAC.
Besides using [5], we also use [9] to evaluate the mAA of the model on new unseen and **different** scenes with respect to that of the training. It is a dataset mainly composed of **outdoor** scenes completely unrelated to that of the training. From all the scenes available in [9], we use the Notre-Dame scene to evaluate the mAA of the proposed approach.

PhotoTourism [9] dataset is an important and well-known dataset of scenes. It is an RGB/Grayscale dataset, which means that we shall not be able to test a neural network using the depth information on it. However, it contains scenes that are extremely complex to match, it is a good benchmark to verify if the DNN could improve the accuracy of DNN RANSAC. We will compare DNN RANSAC using only RGB channels and classical RANSAC on this dataset to evaluate if the DNN model trained on indoor scenes is capable of generalizing and producing good results on completely different scenes.

Experiments setup. The model main hyper-parameter is the patch dimension that defines which part of the scene we consider surrounding a key point. This hyper-parameter is extremely important since too small or too high values could bring inconsistent results in which the network won't be able to recognize anything or it will consider too many things. Moreover, the path dimension is also connected to the distance we have from the object photographed. Therefore, images of the same objects from very different distances could bring some problems in that sense.

Results and discussion. Testing on [5] testing scene resulted in higher mAA of the DNN-based approach concerning that obtained by classical RANSAC. However, we observed that by increasing the distance between frames, even though the difference in terms of mAA between DNN-based and classical RANSAC shrinks, the DNN-based approach still outperforms the classical RANSAC implementation.

Testing on [9] the DNN RANSAC trained on the RGB channel, we can see that it is not able to generalize from the training of indoor scenes. The DNN approach tends to have a slightly lower mAA than the classical RANSAC approach, for many values of the threshold. However, as we increase the threshold, the difference between the two approaches shrinks. Moreover, if we continue to increase it, at some point the DNN RANSAC outperforms classic RANSAC.

Furthermore, we also tried to remove the depth channel in the input by assuming that the sensor could be noisy. If such a hypothesis is true, we will have that the same network trained using RGB and RGBD images will have identical accuracy or that the RGBD network will have lower accuracy. To perform such a test, we will use a scene from [9] since it is the only dataset we use with the depth channel. We have observed that for low thresholds RGB Triplet network makes the DNN RANSAC approach identical to classic RANSAC. However, as the threshold increases, the difference between the accuracy of the two approaches enlarges and DNN RANSAC outperforms classic RANSAC. Differently, if we use the RGB-D Triplet network, DNN RANSAC always outperforms classic RANSAC. This result concludes that the depth channel not only is useful to increase mAA of RANSAC, but it also shows that depth does not add noise, it adds useful semantic information.

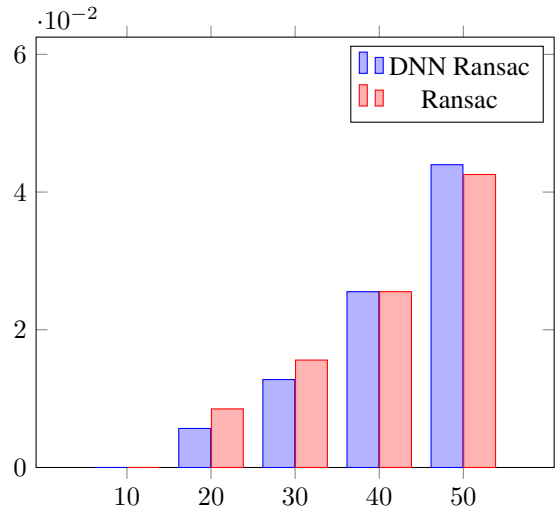

Figure 4. Testing results on [9] as mAA.

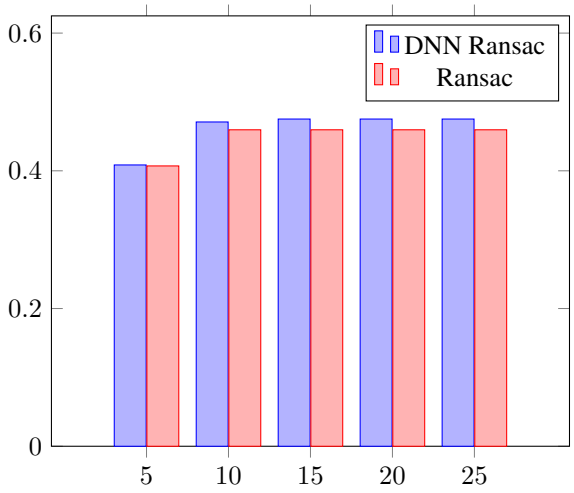

Figure 5. Testing results on [5] as mAA using RGB triplet network.

5. Conclusion

The experimental results performed until now seem promising and this work could be furtherly explored in several directions. For instance, one aspect which is interesting to focus on is the patch size dimension consequences. From a theoretical perspective we can suppose through a Machine Learning parallelism with k-Nearest-Neighbors: if k is too small — namely the patch size — the model is prone to overfit the data by modeling the noise, on contrary, if k is too big, the model is prone to underfit by always predicting the same conclusion — namely the same latent vector representation will be encoded —.

Acknowledgements

We would like to thank every person involved in the evaluation and guidance of this project: Vincenzo Caglioti, Giacomo Boracchi, Luca Magri, and Antonino Maria Rizzo.

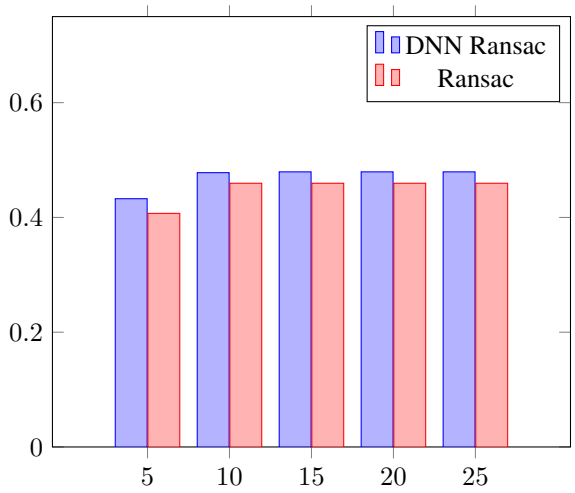

Figure 6. Testing results on [5] as mAA using RGB-D triplet network.

References

[1] D. Barath, M. Ivashechkin, and J. Matas. Progressive NAPSAC: sampling from gradually growing neighborhoods. *CoRR*, abs/1906.02295, 2019. 1

[2] O. Chum and J. Matas. Matching with prosac - progressive sample consensus. volume 1, pages 220 – 226 vol. 1, 07 2005. 1

[3] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. 1

[4] Y. Jin, D. Mishkin, A. Mishchuk, J. Matas, P. Fua, K. M. Yi, and E. Trulls. Image matching across wide baselines: From paper to practice. *International Journal of Computer Vision*, 129(2):517–547, 2021. 4

[5] K. Lai, L. Bo, and D. Fox. Unsupervised feature learning for 3d scene labeling. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3050–3057, 2014. 5, 6

[6] T. Lindeberg. *Scale Invariant Feature Transform*, volume 7. 05 2012. 1

[7] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. Orb: An efficient alternative to sift or surf. In *2011 International Conference on Computer Vision*, pages 2564–2571, 2011. 1

[8] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. 3

[9] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, page 835–846, New York, NY, USA, 2006. Association for Computing Machinery. 5, 6

Testing Set visual results

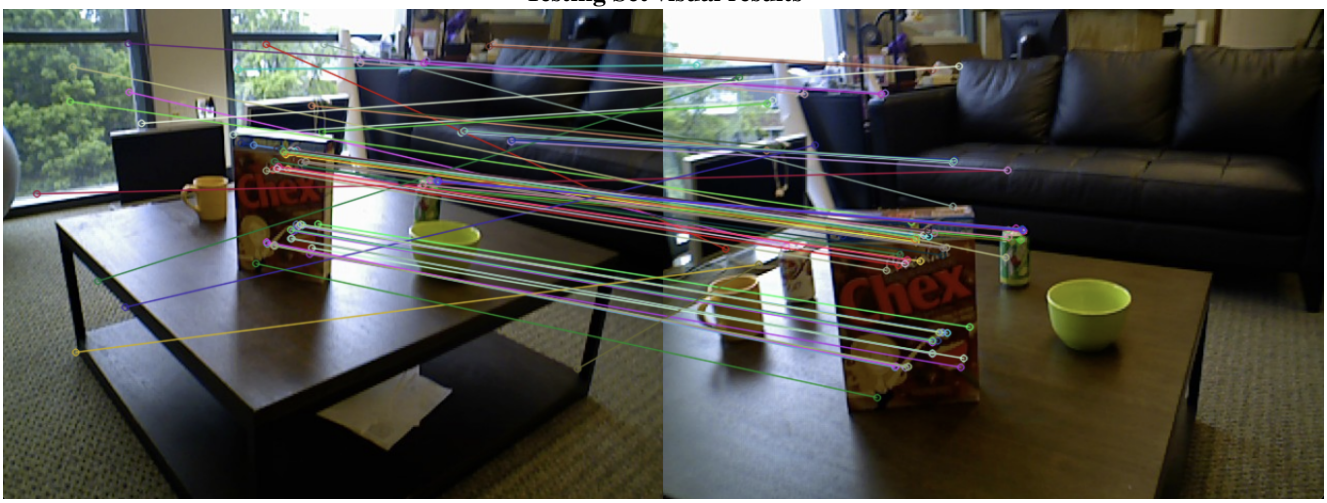

Figure 7. Two different frames of the same scene matched with kNN and ORB detection.

–

Figure 8. RANSAC applied to the above mentioned matches.

–

Figure 9. RGB-D Patch-Based Triplet Network Semmantic Sampling enforcing the above mentioned RANSAC.