Camera Localization for better 3D gaze estimation

Semester Project Proposal
Supervised by: Xi Wang (xi.wang@inf.ethz.ch)
February 24, 2023

BIN YANG

Abstract

This project aims at solving the problem of how frames from the Eye-Tracker could be registered onto an existing 3D data. With the precise registration, we can track the camera position of Eye-Tracker and extract 3D information, which can be used for solving further tasks such as estimation of 3D gaze fixation. To explore different approaches and refinement, a set or markers are placed in the room in advance. Based on that, two frameworks for registering frames with unknown camera poses will be introduced: the first one fully relies on the markers and the second is extended with off-shell methods of Structure From Motion. Furthermore, we tested and evaluated the whole frameworks and their components with different criteria including the numerical analysis and visualization.

I. INTRODUCTION

From the sociological aspect, eye gaze is recognized as one of the most important communicative way in two-person interaction.[11] However, its application has been already extended into wider range of areas in the past few decades, such as human-computer interaction[15], virtual reality[17] and robotics[9]. To automatically track and predict where a person exactly see, known as gaze estimation, various deep-learning based methods are proposed and explored.[22], [1], [2]

In the initial stage of previous work, most of researches focused on the 2D gaze tracking techniques, which map the movement of the sight on the 2D screen for the human-computer interaction, or simply detect the 2D gaze positions.[7] In the more recent approaches, 3D orientation of eye gaze is estimated for further applications, however, only using a 2D input reveals some limitation such as scale ambiguity due to the lack of depth information. Therefore, 3D contents are also encouraged to use to increase the accuracy of the prediction.[2]

To get access to the reliable depth and 3D information, the accurate localization of the camera plays a key role. In the past few decades, different methods and techniques have been studied and developed to complete the task of camera localization. For example, it is straightforward to use the device with lidar and gps sensors. That allows us to track the current position and explore the surroundings in the unknown environment in real-time, which can definitely deliver the result with high accuracy. However, lidar is not always equipped on the device from the economic and technical reasons. Generally, lidar is more expensive compared to other sensors like camera and radar, what's more, lidar has the relatively larger size and it is difficult to assembly on some abbreviated devices such as AR glasses and tiny motion camera.

Besides investigating on the sensors, there are also various visual-based localization methods which can be roughly categorized in two types: deep learning based and visual-odometry based methods. With some specific architectures of the neural network, researches can directly track the correspondence of 2D frame and 3D point cloud by matching their features or instance masks[21] and that could provide a robust correspondences for the fine localization, however, those methods are always sensitive to the domain shift, which means that the result could be randomly bad if we apply it to different dataset without any fine-tuning of parameters. In addition, it is also very costly to produce ground truth features for a large-scale dataset.

Regarding to the visual-odometry based methods such as Structure from Motion, the relative camera pose can be estimated based on the image-to-image matching and a 3D map can be reconstructed. Those methods have been developed for few decades and can generally produce the qualifying results for different datasets. However their 3D reconstruction possesses the arbitrary size and coordinate, which can lead to a scale difference between the real and estimated pose.

Thus, due to the respective limitations of those existing methods, we introduced self-designed pipelines by composing off-shell methods. The pipelines serve to register camera pose of the RGB frames from Eye-Tracker onto the 3D scan from IPhone application. Thus, the contributions of the project can be summarized as following:

- We designed one pipeline based on Structure from Motion, because it has higher adaptability and much lower requirement on the dataset compared to the deep-learning based methods, and it is robust to the domain difference as well.
- Due to the scale ratio problem proposed in Structure From Motion, we will set some markers in the room to get the prior information regarding to the object distances in the reconstruction.
- Inspired by the image-matching concept, we designed another pipeline that could register new frames only relying on the markers.

II. Related Works

In this section, we give a brief review of previous papers that is related to our work. As our task is the camera localization with the help of model-based reconstruction and markers, we will focus on them in the following subsections.

A. Structure From Motion

Structure-from-Motion (SfM) from unordered or sequential images has seen tremendous evolution over the years. SfM is the process of reconstructing 3D structure from its projections into a series of images taken from different viewpoints. Incremental SfM (denoted simply as SfM) is a sequential processing pipeline with an iterative reconstruction component. It commonly starts with feature extraction and matching, followed by geometric verification. The resulting scene graph serves as the foundation for the reconstruction stage, which seeds the model with a carefully selected two-view reconstruction, before incrementally registering new images, triangulating 3D points, filtering outliers, and refining the reconstruction using nonlinear optimization (e.g. bundle adjustment). In this project, we used the open-source implementation of SfM algorithm for the experiments and it is called COLMAP[16].

B. ICP + Global Registration

Iterative Closest Point (ICP) is a classic rigid registration algorithm and is widely used for geometric alignment of three-dimensional models (mesh, point cloud and etc.) when an initial estimate of the relative pose is given.[23] Briefly, it alternates between closest point query in the target set and minimization of distance between corresponding points, and is guaranteed to converge to a locally optimal alignment. However, ICP is commonly known as a local registration method, which means that it relies on a rough alignment as initialization. However, it is hard to give an initial estimation of rigid transformation between two 3d models in the real application. Therefore, a preliminary alignment without initialization is necessary before applying the local registration method. In this project, we used the Fast Point Feature Histogram (FPFH) for the global registration.[14] The method serves to the optimal 6DoF transformation between two 3D models by computing and matching their multi-dimensional features. Furthermore, the initial estimation of rigid transformation can be done by matching the corresponding keypoints sets in two 3d models. As visual tags are set in advance, we can map their center points in 3d coordinates directly and find out their best-fitting with SVD solvers in 3d space.[20]

C. Apriltags

Visual fiducials are artificial landmarks designed to be easy to recognize and distinguish from one another and they have been used in wide range of applications including robotics navigation, augmented reality and camera calibration. AprilTag is one of the most commonly used visual fiducial systems. Tags from the system can be created from an ordinary printer, and the AprilTag detection software computes the precise 3D position, orientation, and identity of the tags relative to the camera by predefining the real size of the tags.[12]

III. Dataset

In this project, the dataset consists of two parts: open-source large-scale dataset and a small set of customized dataset. As mentioned in the previous section, we intended to register the frames with unknown camera poses onto the 3D indoor scan with the help of preliminary setup of markers in the room, and it's hard to collect the open-source dataset matching our requirements. On the other hand, it is time-consuming and expensive to create a large-scale customized dataset before running the experiments. Therefore, we determined to use the open-source dataset to solve the subproblems in the pipelines, and then, to use the customized dataset to evaluate the final results. In that case, we avoided the issue of lack of target dataset to a certain extent.

For the open-source dataset, we used the 3D Indoor Scene data collection from ARKitScene.[3] The dataset contains the 3D indoor room scan, RGB frames and their corresponding camera poses. The custom dataset for testing at the final stage is structured as following:

- RGB frames and depth map recorded by 3D Scanner App (using Lidar-integrated IPhone)
- 3D room mesh scanned by 3D Scanner App (using Lidar-integrated IPhone)
- RGB frames from Eye-Tracker

Procedure of recording the customized dataset is to select a set of markers (5-10) from AprilTags systems and print them out. Then, we put them at the different positions in the room. It is important to keep the marker not too close to others in such a way that each frame can detect at least one marker when recording. In the next step, we used IPhone to scan the room. It is required to do the scanning carefully because we have to guarantee the quality of lidar-based reconstruction and not to miss any markers we set in the room. After that, we recorded some videos with Eye-Tracker. The Eye-Tracer we used is called PupilInvisible which contains a detachable scene camera located on the left arm of the glasses capturing scene video and a binocular pair of infrared cameras with matching 850nm infrared illuminator LEDs[19]. We forced to keep moving and turning when recording data with Eye-Tracker in order that images from different viewpoints can be captured.

IV. An Overview of the Proposed Method

In this section, we present an overview of two proposed pipelines. The procedure of the pipelines are shown in the Figure 1

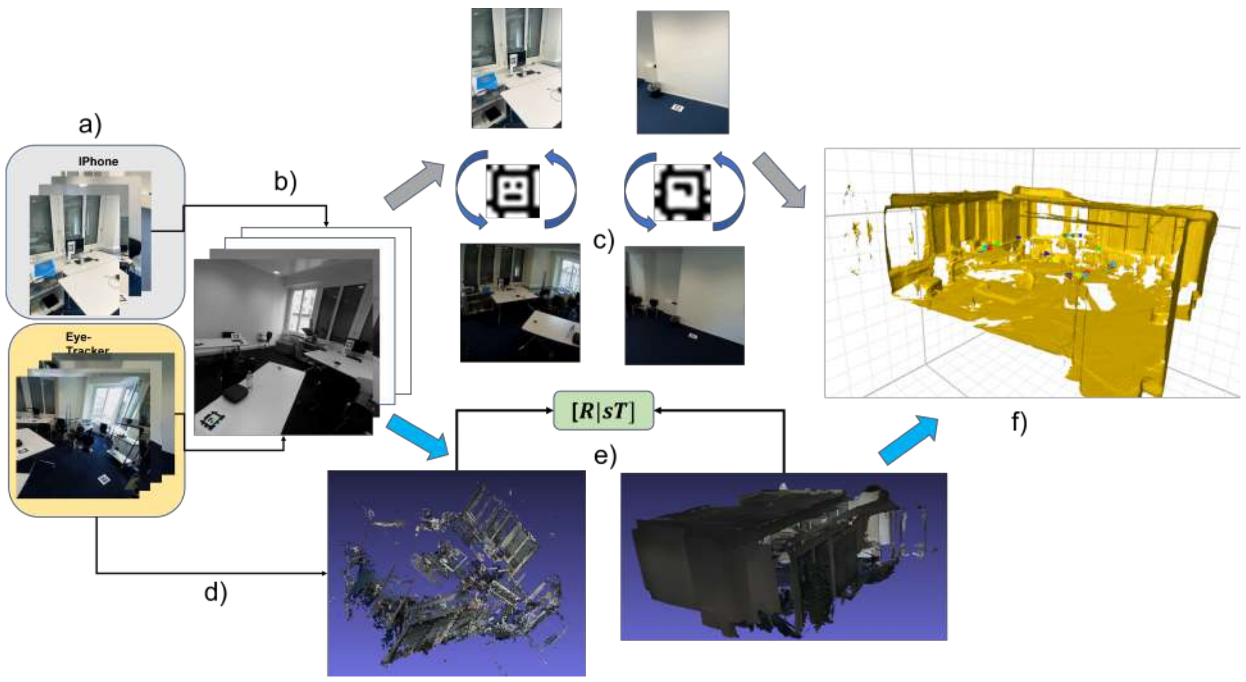


Fig. 1: We registered the frames from Eye-Tracker with two pipelines (Pipeline 1: Branch a - b - c - f, Pipeline 2: Branch a - b - d - e - f). a): Undistort frames, b) Apriltags detection in 2d frames, c) Frames matching by tracking the common tag and Tag Pose Estimation, d) COLMAP reconstruction, e) Point Clouds Alignment with ICP+Global Registration, f) Registration of frames from Eye-Tracker

As the initial step, the RGB frames from IPhone and Eye-Tracker are undistorted respectively. Then, the detection function of AprilTag is applied to mark the detected visual tags in all frames. Two branches are then extended for the registration of the Eye-Tracker's frames. For the first branch (a - b - c - f), the tag poses (rigid transformation between detected tag and camera) are estimated with the built-in solver of AprilTag. A transformation between the camera of IPhone and of Eye-Tracker is able to compute under the condition that two frames share the same tag. As the camera poses of IPhone are pre-recorded, frames from Eye-Tracker can be transformed to the world coordinate of room scan. For the second branch (a - b - d - e - f), a sparse 3D reconstruction is created with the image sequence of Eye-Tracker by using COLMAP. Then, the COLMAP reconstruction is aligned with the recorded room scan with some 3D Registration method. Once two 3D models are aligned properly and accurately, we could then simply transform all camera poses onto the target world coordinate, because frames from Eye-Tracker were already registered onto the COLMAP reconstructed model.

Indeed, the first pipeline is more simple and straightforward, besides, it is potentially much faster than second pipeline, as it avoids the heavy computation of 3D reconstruction. However, it strictly requires that every frame going to register must contain at least one tag that can be detected. Otherwise it can never link to any frame in IPhone dataset. The intuitive solution is to put as many tags in the room as possible, but more artificial tags means more objects. That could cover too much surface of room scan and increase the work capacity of data collection. Therefore, we need a trade-off for the number of tags in use and the number of registering frames. More details related to the pipelines are presented in the following sections.

V. AprilTag

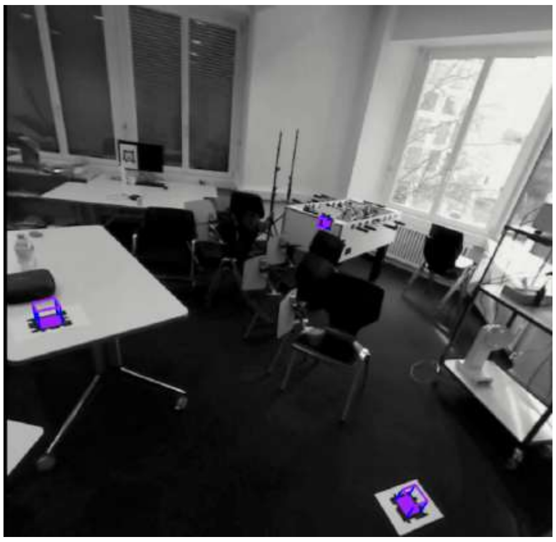
Once the images are properly preprocessed, tags have to be detected with AprilTag functions[12]. Besides the regular detection in 2D image, AprilTag has the implementation of tag pose estimation based on the PnP solver[6]. The ApriltTag system plays a key role in providing the prior information of camera localization. In addition, tags can contribute to building the geographical connection between two frames if viewing the same tag. Beginning with that idea, we built the first pipeline for registering the eye-tracker's frames onto 3D model scanned by IPhone. In this section, we introduced an evaluation criterion for tag pose estimation that is independent from the camera extrinsic.

For using the built-in pose estimation solver of AprilTag, we had to define the intrinsic parameters and image resolution, which are both deterministic. The single fundamental parameter we had to tune in the function is the tag size and it can be measured directly with the print-out tags. In Figure 6, two examples of tag pose estimation are visualized with the augmented reality. As shown in subfigure 2a, the tags appearing in the image can not always be detected correctly, especially when the tag has inherently larger distance to the camera and image has a lower quality (noisy pixel intensities, motion distortion and low resolution).

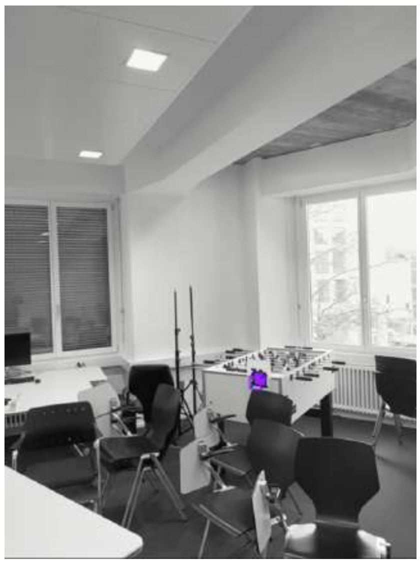
To evaluate the tag detection on 2D frame, the qualitative visualization by marking the detected pixels is enough to show the accuracy. However, the numerical evaluation is required to determine how accurate and stable the tag pose relatively to the camera is estimated under the 3D coordinate. For data with known camera poses, the projection of tags in the 3D space is feasible and the ground truth tags position can be manually annotated in 3D data. The distance error can be regarded as the criterion for evaluation. But for data with unknown camera poses (in our case the frames from Eye-Tracker), it is impossible to do the similar computation. Therefore, we designed a numerical analysis that is only using the 2D images and corresponding camera intrinsic as inputs.

For the first analysis for the built-in solver of AprilTag, the stability of the pose estimation was tested under the scenario that the tag size in the real world is continuously changing. Ideally, the camera-to-tag distance and tag sizes (both in meters) should linearly related: the tag is closer to the camera if it looks bigger. The results of frames from Eye-Tracker and IPhone are in the Figure 3. Indeed, the relationship between two variables can be expressed as the linear function in all cases, which means that the tag position estimated by AprilTag remains stable with the varying tag size.

For the second analysis, we evaluated the linearity, in other words, the error of the slope in Figure 3 numerically. By fitting the straight line, we could explicitly express this constant factor estimated by AprilTag solver, in this case we call it $\alpha_{estimation}$. With the assumption of linearity between distance and tag size, the ground truth value of α_{GT} can be computed as well.

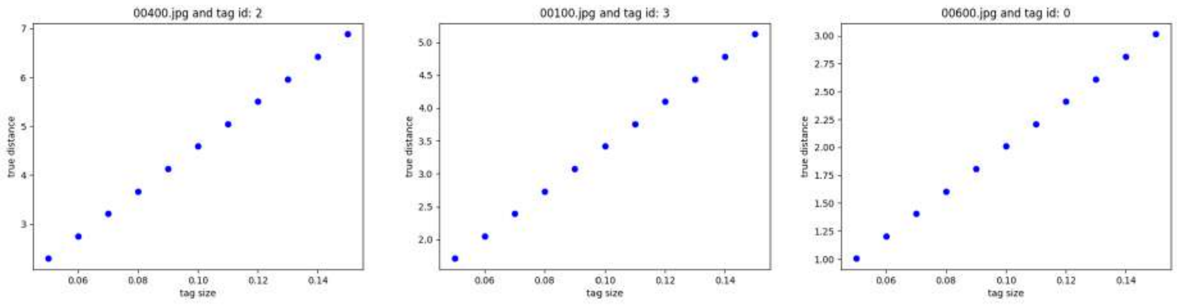


(a) AR-View of an Eye-Tracker's frame

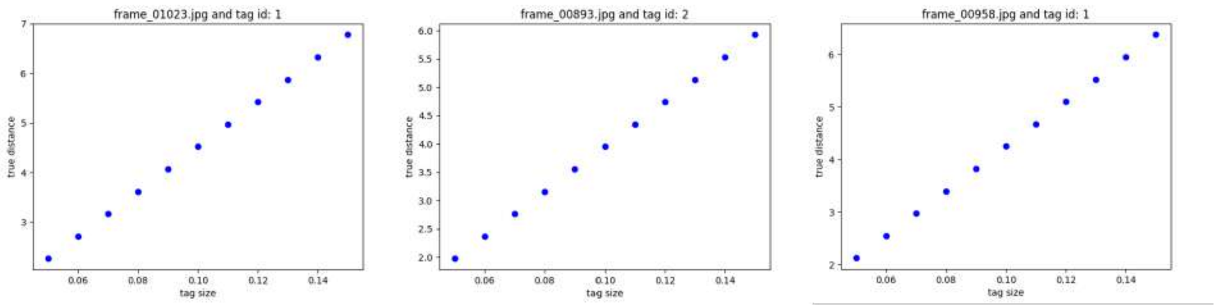


(b) AR-View of an IPhone's frame

Fig. 2: Visualization of Tag Detection and Tag Pose Estimation



(a) Eye-Tracker



(b) IPhone

Fig. 3: Visualization of Relationship between camera-to-tag distance and tag size (in meters)

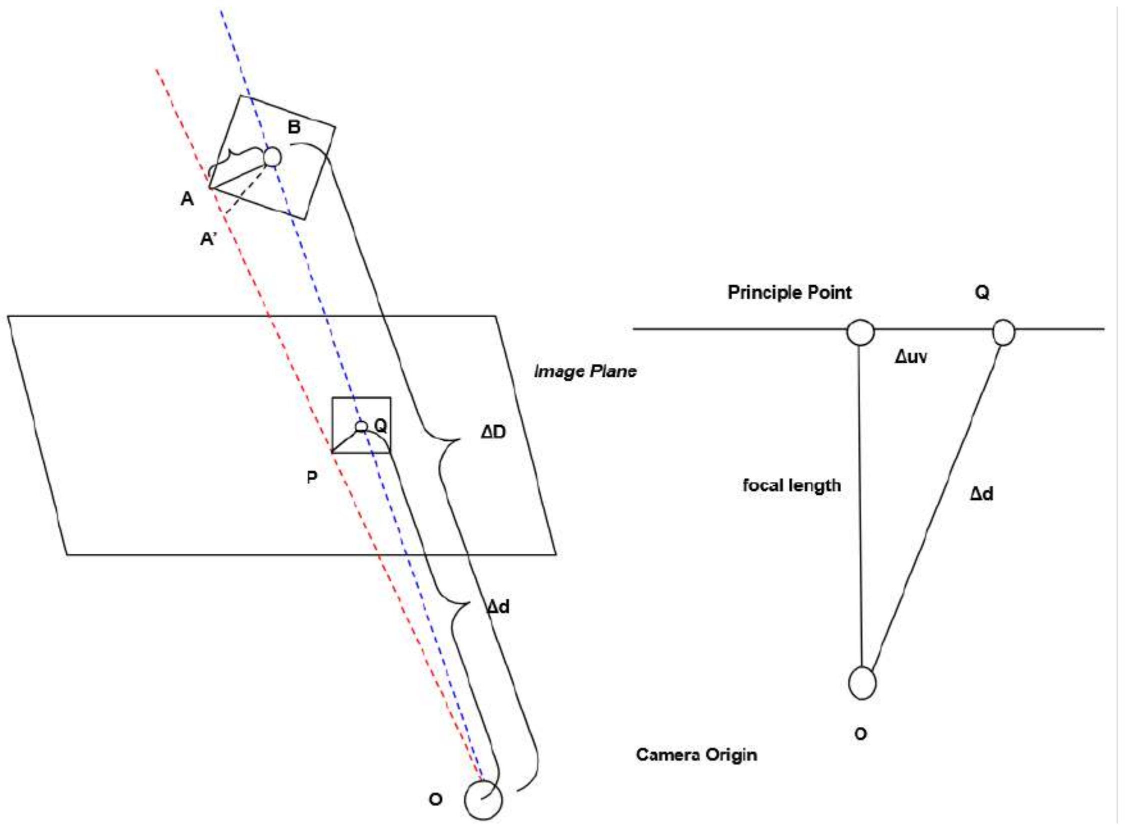


Fig. 4: Pinhole Camera Model: B(A) and Q(P) are tag centers(corners) in 3D space and 2D plane. A' is the projection of A on the plane that is parallel to image plane and intersects point B. ΔD and Δd are distances between camera origin and tag centers. Δuv is the pixel distance between tag center and principal point.

With the knowledge of trigonometry and pinhole camera model shown in Figure4, we have the following mathematical formulations:

$$\frac{PQ}{A'B} = \frac{\Delta d}{\Delta D} \tag{1}$$

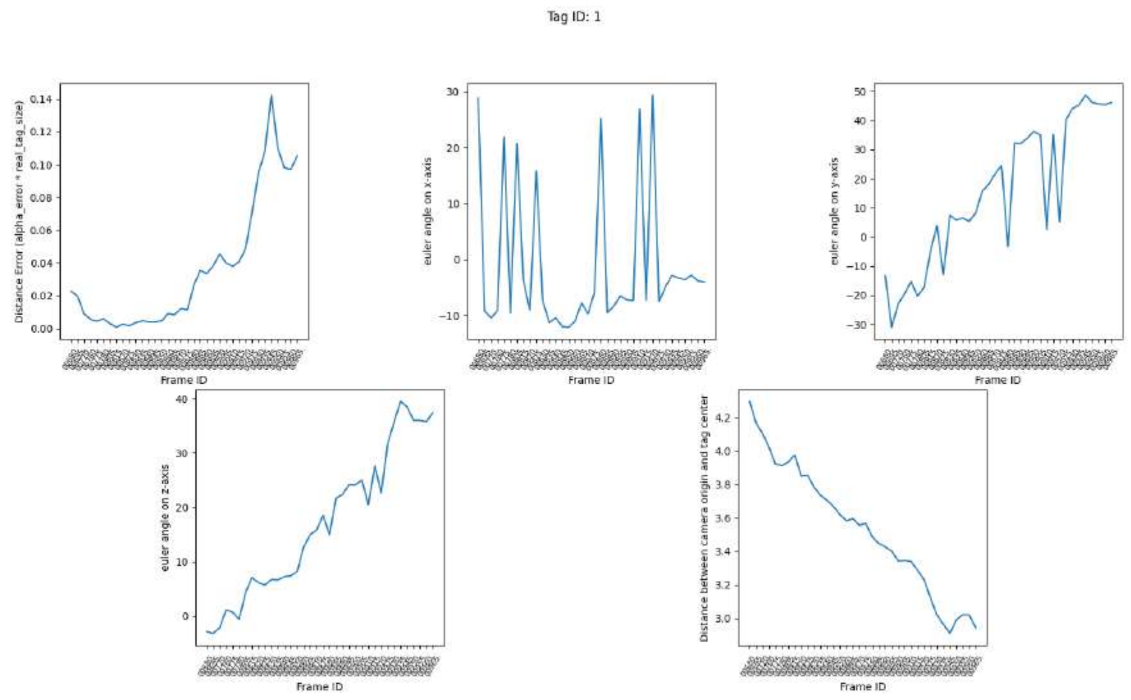
$$\Delta d = \sqrt{(focal\,length)^2 + (\Delta uv)^2} \tag{2}$$

2D point Q and P can be projected in 3D space with estimated tag pose, which correspond to 3D points B and A. A' is mathematically formulable, as we know the image plane is vertical to z-axis and the z-value of point B. Then, the equation 1 can be transformed with the prior condition $\Delta D = \alpha * L_{tag}$ as following:

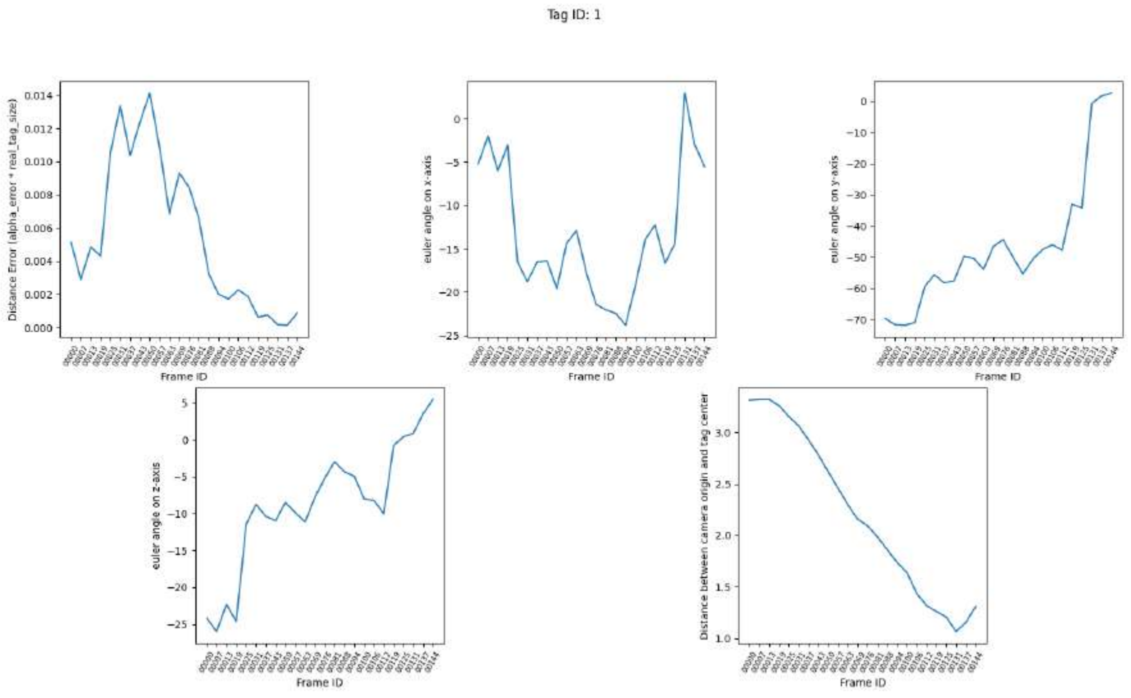
$$\alpha = (\frac{\Delta d}{PQ}) * (\frac{A'B}{L_{tag}}) \tag{3}$$

By setting the L_{tag} with the true size of the tag, we can calculate α_{GT}. Under the ideal situation, α_{GT} and $\alpha_{estimation}$ should remain very close or even identical. Hence, we define the absolute difference between these two parameters as the error. Since $alpha$ is a non-unit parameter and it is hard to explain the physical meaning behind it, we multiply the $\Delta\alpha_{err}$ with the real tag size in the further analysis. In this way we can understand it as the distance error. The results of the evaluation of the same tag for IPhone and Eye-Tracker dataset are presented in Figure 5. Besides the distance error, the change of estimated distance and viewing angles according to the frame iIDs are shown as well.

The Figure 5 manifests that the error of IPhone's frames is lower than of Eye-Tracker's frames overall. The reason of that is the lower image resolution, higher level of distortion (even after image undistortion) and the brightness noise of the latter. Comparing curves of distance error and estimated distance of two dataset, it shows that the distance error rises significantly when the tag is closer to the Eye-tracker while the error drops when tag is closer to the IPhone. If we observe the last few frames in two datasets, we can notice that the viewing angles between tag and camera has a huge gap between two devices. For IPhone, the viewing angles in all 3 dimensions



(a) Eye-Tracker



(b) IPhone

Fig. 5: Evaluation of AprilTag built-in tag pose solver (units are meter and degree). Denote that $alpha_G T$ is computed for each corner of the tag respectively and take the mean value as the output

are close to zero, which means that the tag is almost faced to the IPhone camera straight. On the other hand, the viewing angles between the tag and Eye-Tracker's camera are approximately up to 50 degrees for some dimensions. As discussed by the authors of AprilTag [12], the accuracy of the pose estimation can highly affected by the robot motion, especially when the robot is turning towards larger viewing angles.

Based on the results above and experiments done by the author of AprilTag, we can have an initial assertion that a distinct discrepancy in orientation and distance between the tag and the camera can reduce the accuracy of the tag pose estimation. Under some corner cases (e.g. very large distance and viewing angles), the AprilTag system can even fails to detect the tags in the 2D images. However, further experiments are necessary to verify the inference, for example turning the camera with fixed distance to the tag to test out the accuracy of built-in AprilTag solver with varying viewing angles.

VI. 3D Registration

Besides linking frames from two different devices by tracking the common tag and estimating the tag poses under each camera coordinate, we can firstly register the source frames in an arbitrary 3D space and directly align it to the target one. In this section, we will focus on how to register frames from Eye-Tracker with 3F registration methods instead of relying on the AprilTag system.

A. Global Registration + Local Refinement

As introduced in section II-A, COLMAP is a toolbox with functionality of Structure From Motion which can proceed the reconstruction of 3D structure from its projections into a series of images. During the reconstruction process, COLMAP self-defines a world coordinate based on the initial set of sequential images. After that, camera poses of images, that can be matched with other images in the retrieval, are registered onto this world coordinate. A sparse reconstruction of the point cloud is then obtained by projecting and triangulating the feature points. In addition, the mesh can be created by reconstructing the surfaces for the point cloud with some methods introduced in [5] and [10].

To have the initial knowledge about the quality of COLMAP reconstruction and the performance of 3D registration with indoor scene data, we used the existing large-scale dataset ARKitScenes [3] for the experiments at first. The dataset provides a high quality of room scan with RGBD sensors and RGB frames with low resolution 256 x 192 and high resolution 1920 x 1440. The ground truth room scan and the reconstruction from COLMAP of the sample data are shown in the Figure 6a and 6b. As mentioned in the section I, despite that COLMAP already presents the satisfying quality of output that contains the complete reconstruction of some objects in the room, there exists the issue of random scaling factor in the reconstruction. It comes from the fact that it is impossible to determine the size of the scene from 2D measurements alone without any prior information regarding to some distance in the 3D space. The initial states of reconstructed mesh and room scan are shown in Figure 6c. Consequently, the purpose is to align these two meshes by estimating the translation, rotation and scale ratio.

As the first experiment, we tested the performance of Iterative Closest Point Algorithm (ICP) with global registration. Specifically, the steps are as following:

1) Sample point clouds from two meshes respectively
2) Normalize both point clouds in range of -1m to 1m. This step aims at unifying the hyperparameters voxel size and radius for estimating normal vectors and computing multi-dimensional features in the further steps. Otherwise we have to tune the parameters for each 3D data according to its extent.
3) Global Registration
    a) Downsample point clouds (Boost the computation) and estimate the normal vectors
    b) Compute a FPFH (Fast Point Feature Histograms) [14] features for each points. The FPFH feature is a 33-dimensional vector that describes the local geometric property of a point. A nearest neighbor query in the 33-dimensional space can return points with similar local geometric structures.
    c) Use RANSAC [4] to filter out the outliers of matched features
    d) Computing the initial estimation of transformation matrix and scale ratio
4) Local Refinement with ICP Algorithm
    a) Up/Down-scale the point clouds

b) Set the estimation of Global Registration as initial transformation
c) Compute the Point-to-Point ICP Alignment

The result of the Global Registration + Local Refinement with scale estimation can be seen in Figure 6d. The alignment has the substandard accuracy and is inadequate for further registration of camera poses. The reasons for the poor performance are that there exists still the distinct difference in scale of two point clouds after normalization and global registration. More specifically, Rusu (the author of FPFH features) has experimented and analyzed the phenomenon in his doctoral dissertation [13] that the larger scale of the model covers less points in the adjacent surfaces under the condition that parameters are defined identically, and then, the estimated point feature representations can get distorted. The issue leads to the consequence that the geometrically similar surfaces in two meshes can be mismatched due to their difference in feature space, eventually, the scale ratio failed to be estimated correctly. As ICP Algorithm only works actively when there exists only relative pose between source and target model and does not account for compensating the scale discrepancy, the final estimation of transformation is significantly inferior to the ideal result. In spite of the drawback, we could still see the improvement in the alignment of two meshes from Figure 6c and 6d. To further test the pipeline and inspired by the discussion in Rusu's work, we found that the 3D registration works superbly with prior scale information (the scale factor is simply computed by the manual annotation here). By pre-scaling two meshes to the same size and getting rid of scale ratio estimation in the pipeline, we had the result as shown in Figure 6e. Thus, how to calculate the scale ratio separately from 3D registration became the new subproblem.

B. *Keypoints Alignment*

Coincidentally, visual markers of AprilTag we discussed in the last section can contribute to extracting this piece of prior information, more specifically, we can regard the center points of the tags in the 3D space as keypoints and compute the scale ratio by aligning two set of corresponding keypoints. (seeing Figure 7)

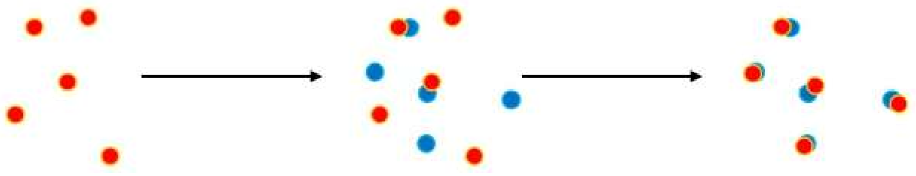


Fig. 7: Aligning sets of points in space

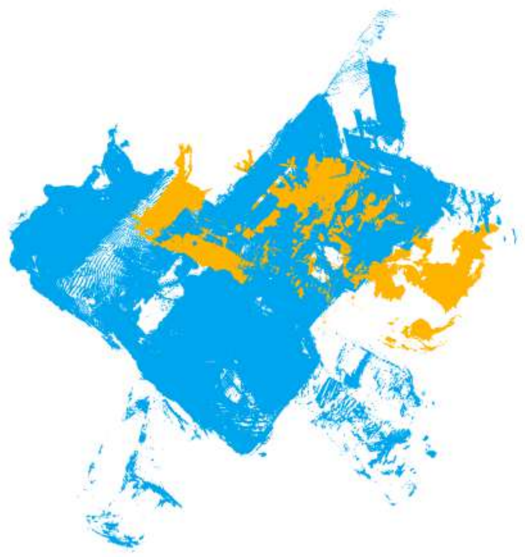
Before doing experiments with the room scan including the markers, we firstly tested the accuracy of keypoints alignment with ARKitScenes' data. To obtain the reliable keypoint sets, we used the open-source 3D annotation tool to manually select the keypoints in the reconstructed mesh and ground truth room scan. In addition, we defined three rules for the manual annotation and strictly followed them:

- The most distinct objects have to be chosen at first. As an illustration, the pink red round desk has the higher priority than the shelf on its left hand in Figure 6a and 6b.
- We have to annotate the keypoints in the middle of the objects. The reason is that the points on the boundary may lead to the faulty annotation because the COLMAP cannot always recover the sharpness of the object.
- We have to make sure that keypoints are annotated distantly. It makes no sense to align two sets of extremely centered points.

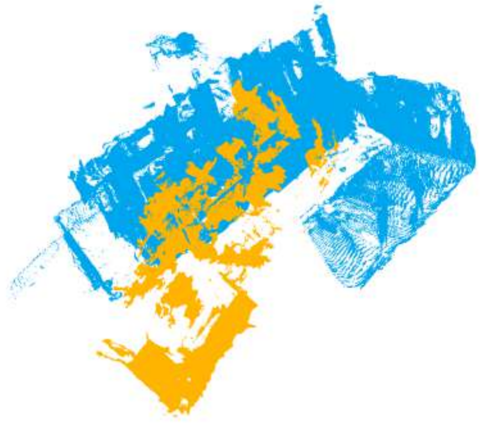
As long as we created the keypoint sets, we applied the algorithm introduced in [20]. The algorithm estimated not only the scale difference between two sets of corresponding points but also the bet-fitting rigid transformation towards them, which means that we can skip the step of global registration in the last subsection, if the estimated transformation from keypoints can similarly fit the meshes as well. To roughly test the effect of the size of keypoint set on the quality of alignment, we computed with use of different number of keypoints and apply the results to the meshes. The visualization can be seen in Figure 8. Obviously, the transformation and scale ratio estimated from the set containing more keypoints is superior to the one with less keypoints. To compute the precision of localization numerically, we also compared the 6-DoF camera pose precision from two methods (Global Registration / Keypoints Alignment + Local Refinement with ICP). Rotation error is measured by $\epsilon = arccos((trace(\mathbf{R}_{gt}^T\mathbf{R}_{es}) - 1)/2)$

(a) Ground Truth Room Scan

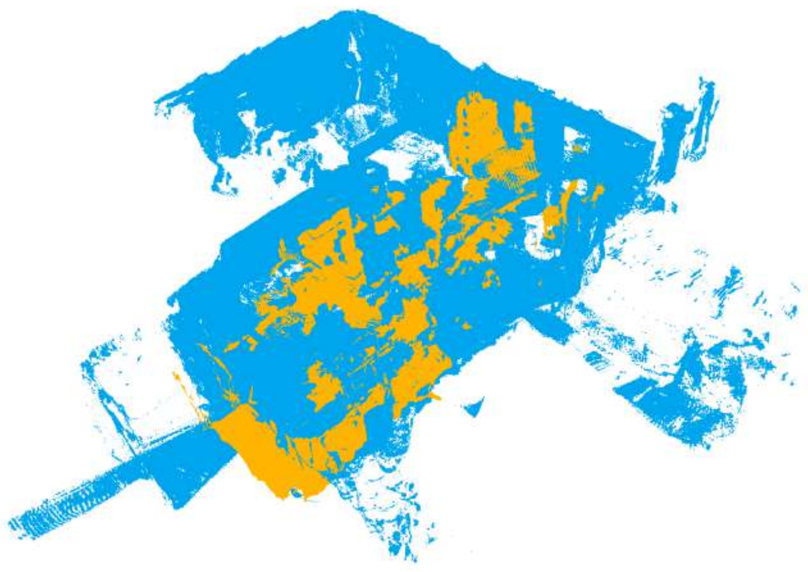
(b) COLMAP Reconstruction with IPhone frames



(c) Scale and Pose difference of the initial state



(d) Global Registration + ICP Local Refinement with unknown scale ratio



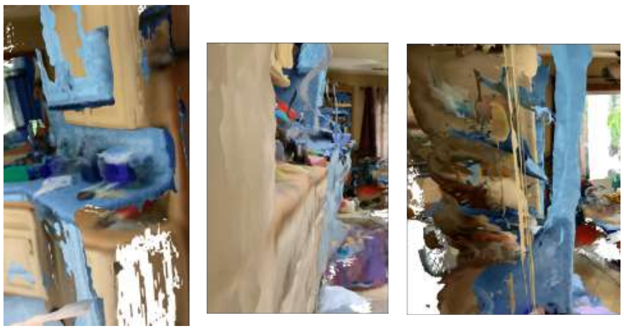
(e) Global Registration + ICP Local Refinement with known scale ratio

Fig. 6: Results of Mesh Alignment (Data is from ARKitScenes Dataset [3], data id: 40777060)

where $\mathbf{R}_{gt}$ is the ground-truth rotation, and $\mathbf{R}_{es}$ the estimated one. Translation error is measured by the absolute distance between the estimate and the ground-truth camera center. The test was operated on the multiple data from ARKitScenes [3]. The 3D registration method using the keypoints alignment outperforms the feature-based global registration at almost all position/rotation levels.

ARKitScenes	Global Registration + Local Refinement		Keypoints Alignment + Local Refinement	
Data ID	$\mathbf{R}_{err}$ (average)	$\mathbf{T}_{err}$ (average)	$\mathbf{R}_{err}$ (average)	$\mathbf{T}_{err}$ (average)
40777060	25.3°	2.421 m	2.35°	0.027 m
41069042	37.3°	1.865 m	3.42°	0.034 m
40958764	26.8°	3.062 m	1.48°	0.012 m
41126500	31.2°	2.491 m	2.42°	0.018 m

TABLE I: The average rotation and translation error of two methods


(a) Mesh Alignment with 4 keypoints

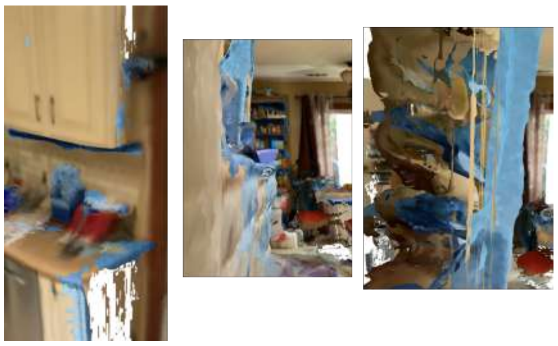

(b) Mesh Alignment with 10 keypoints

Fig. 8: Align meshes with transformation and scale estimated from keypoint sets (we reversed the RGB channels of COLMAP reconstruction for more clear visualization)

C. Keypoints Extraction

Afterwards, with our custom dataset that contains markers from ArpilTags, we can regard the marker's center as the keypoint and extract them from 3d space with some automatic procedure instead of manual annotation. As ground truth room scan is recorded in the real-world environment and reconstruction created by COLMAP contains inherently computational error, which is highly depending on the input image sequence, we will apply different methods to obtain the marker centers respectively.

For the ground truth room scan, the first option is to use the built-in solver of AprilTag to estimate the tag pose, however, as we tested in the section V, the precision of tag pose is not always high under some scenarios and thus, it can not project the target 3D point of the tag center accurately. Another option is to project the tag center from 2D image plane to 3D world directly. As the depth of tag center is unknown, we use the basic ray casting method to find the intersection. The method works perfectly for the room scan, because the ground truth camera poses of IPhone's frames are given and the detection of tags is very precise in 2D image.

For the reconstruction, we determine the keypoints with the sparse model of COLMAP. The sparse model is stored as binary files and it tracks the correspondence between triangulated 3D points and 2D pixels in the images. With that, we can mask the pixels covering the tags in the image and then highlight the corresponding 3D points in the sparse output. Next, to extract the single keypoint for each marker, we simply compute the mean values of all projected 3D points according to the marker, however, the reconstruction contains the noise and outliers, which could negatively affect the accuracy. Therefore, we preprocess those 3D points before calculating the center point. The preprocessing steps are concluded as following:

1) Set the threshold and remove points whose reprojection error exceeds the threshold
2) Classify points corresponding to one tag with K-nearest Clustering
3) Choose the cluster containing the most points as the output

The procedure for extracting the 3D marker points from COLMAP reconstruction is shown in Figure 9

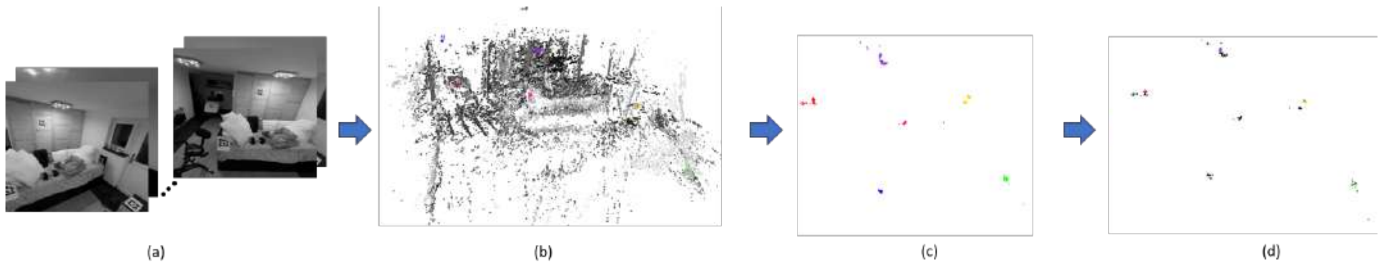


Fig. 9: (a): Input image sequence, (b): Sparse COLMAP Reconstruction with highlighted marker points, (c) Unfiltered marker points, (d) Filtered maker points

D. Complete Pipeline for 3D Registration

Based on the previous experiments and discussion, we finalized the latest version of the 3D registration Pipeline, as shown in Figure 10.

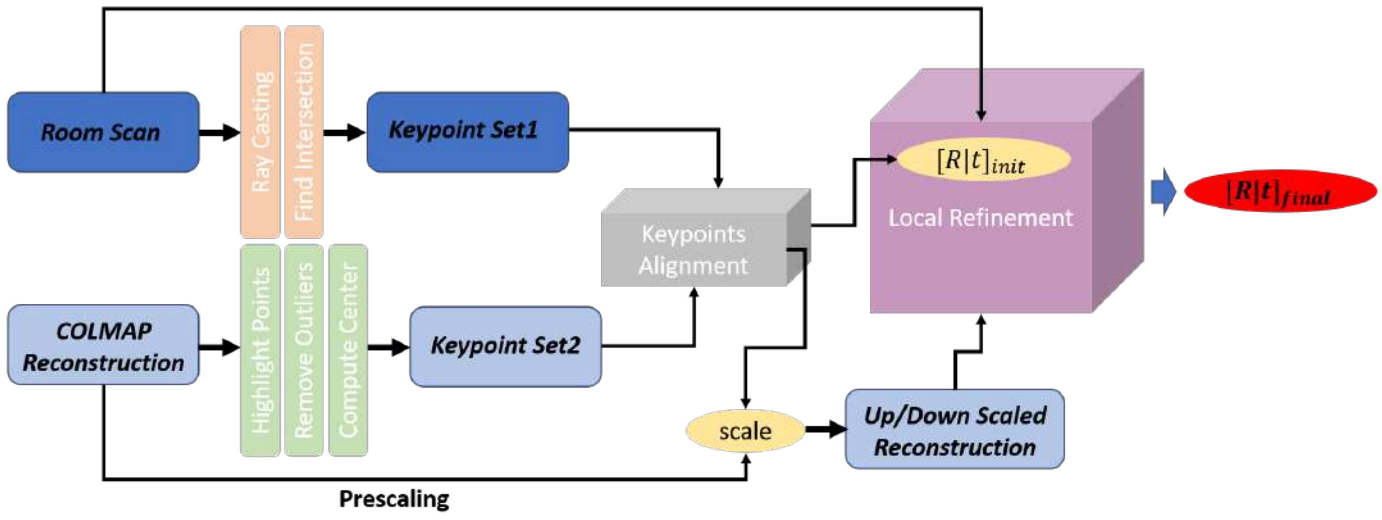


Fig. 10: Complete Procedure of 3D registration

VII. Experiments for Camera Localization

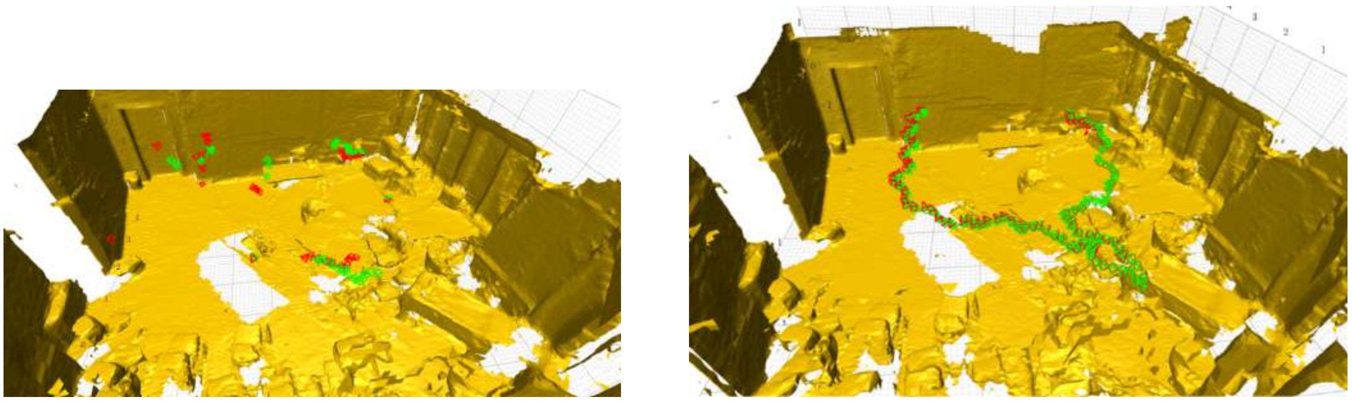
In the previous sections, we introduced the details and experiments for testing different components in two pipelines for camera localization (proposed in section I). In this section, we will validate the efficacy of them with our customized dataset.

Since frames recorded with Eye-Tracker have no corresponding ground truth camera poses, we cannot directly measure the rotation and translation error of the estimation. Thus, we proceed the numerical measurement for registration results with IPhone frames at first. The Table II gives the error and number of registered images by applying two methods. Indeed, the pipeline with COLMAP has obviously the advantages over the one with AprilTag with respect to position and rotation error, in the meantime, the former is able to register a larger amount of frames for all datasets. The discrepancy in number of registered frames is mainly caused by the failure of tag detection in the certain amount of frames.

In addition to numerical analysis, we also provide the visualization of registration results of two methods (seeing Figure 11). From subfigures 11a, we find that there is a distinct gap between the ground truth and estimated camera poses using the AprilTag pipeline. In addition to the larger absolute error, some outliers (distance error > 2 m and rotation error $> 90°$) can be seen from the observation. The drawbacks are mainly caused by the high variability in the measurement of the tag poses estimated by AprilTag systems (as discussed in section V), and additionally, this method is completely relying on the AprilTag, which means that there is no further refinement to alleviate the negative effect of wrong measurements. In contrast to AprilTag pipeline, registration process with COLMAP shows significantly superior results. First of all, there is no presence of outliers in the estimation. Furthermore, the trajectory of camera estimated by the method is highly aligned with the ground truth and the shape of them are notably fitted to each other. For the reason that the COLMAP pipeline contains more components including preporcessing and postprocessing steps, the performance is more stable and generally better than the one with AprilTag.

Custom Dataset	Pipeline 1 (with Apriltag)			Pipeline 2 (with COLMAP)		
Data ID	#(images)	$\mathbf{R}_{err}$ (average)	$\mathbf{T}_{err}$ (average)	#(images)	$\mathbf{R}_{err}$ (average)	$\mathbf{T}_{err}$ (average)
AprilTag1 Dataset1	38	13.934°	0.742 m	171	0.734°	0.050 m
AprilTag1 Dataset2	40	16.310°	0.651 m	145	1.421°	0.097 m
AprilTag2 Dataset1	30	18.450°	0.945 m	121	0.679°	0.067 m
AprilTag2 Dataset2	31	10.021°	0.308 m	162	2.231°	0.104 m

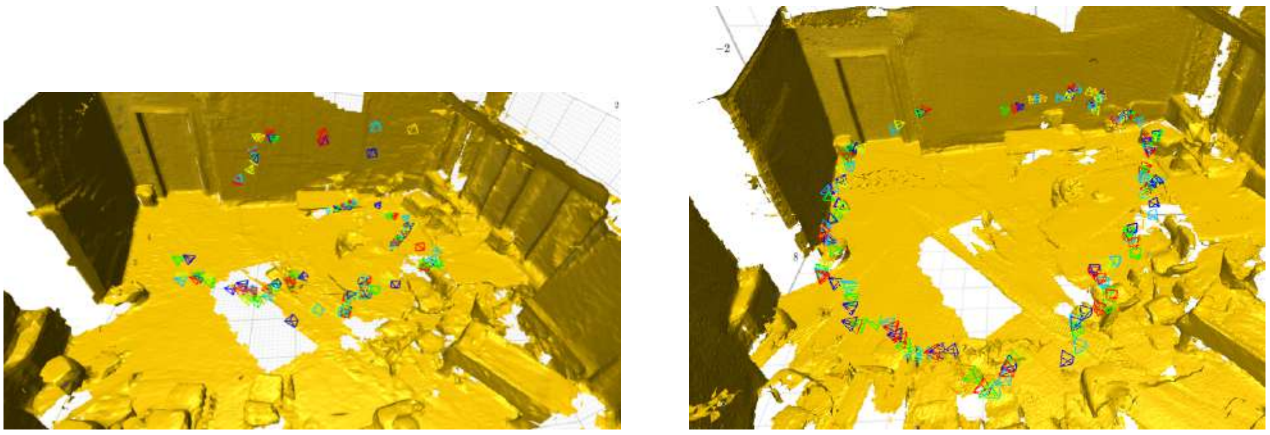
TABLE II: The average rotation and translation error of two methods. AprilTag1 and AprilTag2 are dataset recorded with different setting of markers in the room. Dataset1 and Dataset2 are recorded with different traveling routes.



(a) AprilTag Pipeline (IPhone frames) (b) COLMAP Pipeline (IPhone frames)

Fig. 11: Visualization of registration results with two proposed pipelines (green is ground truth and red is estimation).

Likewise, we apply two methods to the dataset recorded with Eye-Tracker. Figure 12 shows the estimated camera poses visualized in the ground truth room scan. From the visualization, it can be similarly inducted as the experimental results with IPhone frames that camera poses estimated by COLMAP Pipeline shows more continuity and completeness in trajectory of motion compared to AprilTag. To further verify the accuracy of estimation from visual aspect, we randomly picked some frames and rendered the images from mesh with their estimated camera pose from two methods. The results are shown in Figure 13.



(a) AprilTag Pipeline (Eye-Tracker frames) (b) COLMAP Pipeline (Eye-Tracker frames)

Fig. 12: Visualization of registration results with two proposed pipelines (approx. 200 frames)

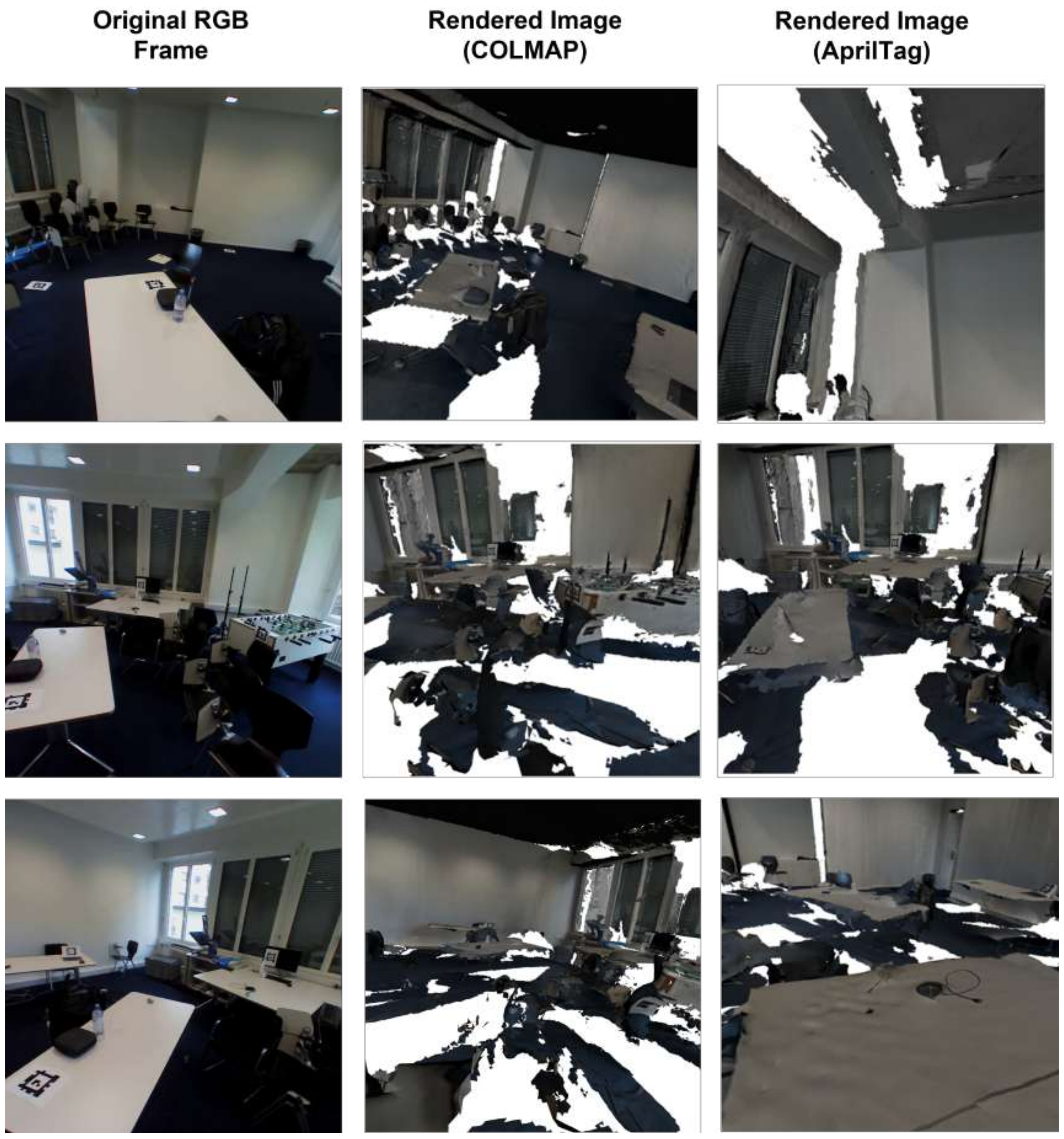


Fig. 13: Original images and rendered images. (images are created based on their estimated camera poses from two frameworks)

In most of cases, some shift in the camera position and viewing direction can be observed by comparing the original RGB frames and rendered images. The preliminary reason for the descending performance with Eye-Tracker is the lower quality of images. As shown in Figure 14, the image taken by the camera of Eye-Tracker has the larger field of view, but the illumination of IPhone frame is obviously better under the same lightning condition in the real world. Underlying the fact that the captured image sequence has the higher quality, the COLMAP reconstruction of IPhone is considerably superior to the one of Eye-Tracker with setting identical parameters in the process. As a matter of fact, the optimal results in SfM-based reconstruction is fundamental in the whole framework with COLMAP because it directly affects the accuracy of both keypoints we select for the initial alignment and the ICP algorithm for 3D registration. Those errors can be accumulated due to the more complicated structures of COLMAP framework. Therefore, it requires more work for the refinement.

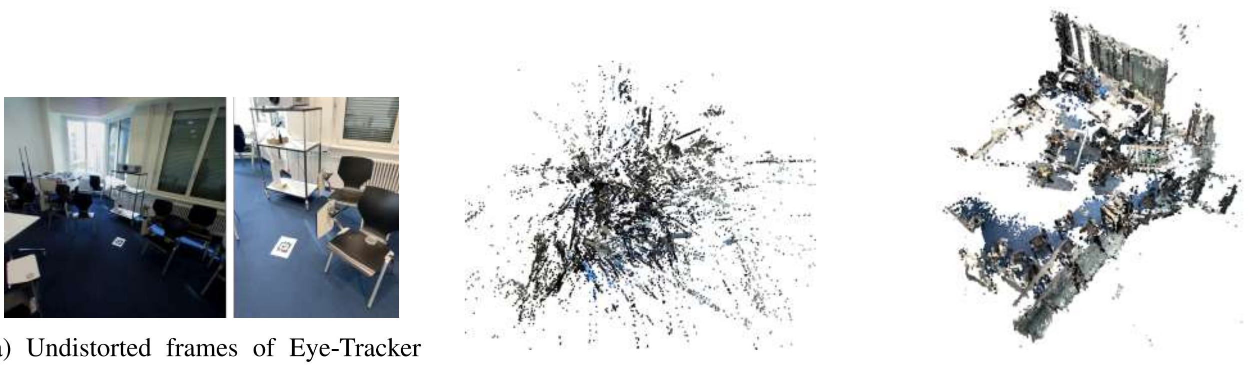

(a) Undistorted frames of Eye-Tracker (left) and IPhone (right) (b) Eye-Tracker (c) IPhone

Fig. 14: Example images and COLMAP Reconstruction with image sequence

VIII. Conclusion

A definitive and accurate camera localization is always the sufficient condition for getting the reliable 3D information for the corresponding frame. In this project, we mainly focused on how to register frames with unknown camera poses onto the given 3D indoor scene data. More specifically, we aims at localizing the camera of Eye-Tracker in the room scan which is recorded by the lidar-based application in order that we can extract more three-dimensional features for each frame such as depth map, object-based mapping and etc. Those features can potentially contribute to improve the performance of gaze estimation in the 3D world. We have proposed two frameworks to complete the task: Marker-based and SfM-based registration.

The key idea of Marker-based registration is to use the visual tag to build the connection between two independent frames. When two frames share the same tag and it can be detected correctly, we can estimate the tag poses with respect to their camera centers respectively. By merging the transformation and the known camera pose of one frame, we can register the another onto the same world coordinate. However, tag pose estimation is not always accurate and it can be significantly affected by the viewing angles and distance between the tag and camera.

Another framework is based on the traditional geometric method: Structure From Motion. With only image sequence, we can compute the 3D reconstruction by extracting 2D features ,matching them to stereo pairs and triangulating to project them into 3D space. Images are automatically localized in the reconstruction during the process. However, the output of SfM and the ground truth scan are different in the scale and initial 6-DoF pose. Then, we solved the subtask by aligning two 3D models with keypoints and ICP algorithm. From the experimental results, we found that this framework outperforms the one with AprilTag in use of both IPhone and Eye-Tracker frames.

IX. Future Works

Future investigations are necessary for both proposed frameworks in this project. For the pipeline based on AprilTag systems, the performance of camera localization is sensitive to the relative position between tag and camera in the real world. Thus, future studies could fruitfully explore this issue further by measuring the error of AprilTag under some specific trajectories of camera motion. Intuitively, the experiments can be concluded as following:

- To get the safety range of viewing angles and distances that could detect the tags and compute the poses over the baseline accuracy:
  - Rotate the camera with small step size by keeping the constant distance to the tag. Repeat it for multiple times with different distances.
  - Move the camera from near to far by keeping the constant viewing angle to the tag. Repeat it for multiple times with different viewing angles
- To learn the effect of number of tags on the camera localization:
  - Compute the distribution of detection rate over the number of tags setting in the room.
    (detection rate $= \frac{\#(\text{frames with detected poses})}{\#(\text{frames})}$)

– Compute the distribution of validity over the number of tags setting in the room. (validity $= \frac{\#(\text{frames with detected poses within safety range})}{\#(\text{frames with detected poses})}$)

For the pipeline with SfM, the key is to refine the results of COLMAP reconstruction. Since the issue of illumination noise in images taken by Eye-Tracker is mainly depending on the integrated camera, the potential improvement for the image quality is very restricted. Therefore, it might be meaningful to use other methods to proceed the reconstruction. Except traditional geometry-based method, some current researches regarding to the deep-learning based scene reconstruction might contribute to the better output. For example, NeuralRecon [18] reconstruct local surfaces represented as sparse Truncated Signed Distance Field (TSDF) volumes for each video fragment sequentially by a neural network. Besides, Neural Radiance Fields (NeRF) [8] is a state-of-the-art method that generates novel views of complex scenes by optimizing an underlying continuous volumetric scene function using a sparse set of input views. Specifically, we can use COLMAP to estimate the camera poses and then reconstruct the scene with NeRF.

References

[1] Oncel Tuzel Josh Susskind Wenda Wang Russ Webb Apple Inc Ashish Shrivastava, Tomas Pfister. Learning from simulated and unsupervised images through adversarial training. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[2] Jun Bao, Buyu Liu, and Jun Yu. Escnet: Gaze target detection with the understanding of 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14126–14135, June 2022.

[3] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. Arkitscenes - a diverse real-world dataset for 3d indoor scene understanding using mobile rgb-d data. In *NeurIPS*, 2021.

[4] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of ransac family. In *British Machine Vision Conference*, 2009.

[5] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson Surface Reconstruction. In Alla Sheffer and Konrad Polthier, editors, *Symposium on Geometry Processing*. The Eurographics Association, 2006.

[6] Vincent Lepetit, Francesc Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81:155–166, 2009.

[7] Mohsen Mansouryar, Julian Steil, Yusuke Sugano, and Andreas Bulling. 3d gaze estimation from 2d pupil positions on monocular head-mounted eye trackers. *CoRR*, abs/1601.02644, 2016.

[8] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[9] Oskar Palinko, Francesco Rea, Giulio Sandini, and Alessandra Sciutti. Robot reading human gaze: Why eye tracking is better than head tracking for human-robot collaboration. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5048–5054, 2016.

[10] Jiju Peethambaran and Ramanathan Muthuganapathy. Reconstruction of water-tight surfaces through delaunay sculpting. *Computer-Aided Design*, 58:62 – 72, 2015. Solid and Physical Modeling 2014.

[11] Anto R.Canigueral. The role of eye gaze during natural social interactions in typical and autistic people. *Typical and Atypical Processing of Gaze*, 10, 2019.

[12] Andrew Richardson, Johannes Strom, and Edwin Olson. AprilCal: Assisted and repeatable camera calibration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, November 2013.

[13] Radu Bogdan Rusu. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Technische Universität München, 2009.

[14] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. In *ICRA*.

[15] Rafael Santos, Nuno Santos, Pedro M. Jorge, and Arnaldo Abrantes. Eye gaze as a human-computer interface. *Procedia Technology*, 17:376–383, 2014. Conference on Electronics, Telecommunications and Computers – CETC 2013.

[16] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[17] Vincent Sitzmann, Ana Serrano, Amy Pavel, Maneesh Agrawala, Diego Gutierrez, Belen Masia, and Gordon Wetzstein. Saliency in vr: How do people explore virtual environments? *IEEE Transactions on Visualization and Computer Graphics*, 24(4):1633–1642, 2018.

[18] Jiaming Sun, Yiming Xie, Linghao Chen, Xiaowei Zhou, and Hujun Bao. NeuralRecon: Real-time coherent 3D reconstruction from monocular video. *CVPR*, 2021.

[19] Marc Tonsen, Chris Kay Baumann, and Kai Dierkes. A high-level description and performance evaluation of pupil invisible. *arXiv preprint arXiv:2009.00508*, 2020.

[20] Jin Wu, Ming Liu, Zebo Zhou, and Rui Li. Fast rigid 3d registration solution: A simple method free of svd and eigen-decomposition. *arXiv: Systems and Control*, 2018.

[21] Fei Xue, Ignas Budvytis, Daniel Olmeda Reino, and Roberto Cipolla. Efficient large-scale localization by global instance recognition. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17327–17336, 2022.

[22] Yiwei Bao Feng Lu Yihua Cheng, Haofei Wang. Appearance-based gaze estimation with deep learning: A review and benchmark. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[23] Juyong Zhang, Yuxin Yao, and Bailin Deng. Fast and robust iterative closest point. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.