Floorplan Localization and Navigation

Ziteng Wang
ETH Zurich
zitwang@student.ethz.ch

Yuhan Zhang
ETH Zurich
yuhzhang@sudent.ethz.ch

Tianyi Zhang
ETH Zurich
zhangtia@sudent.ethz.ch

Lan Feng
ETH Zurich
lafeng@sudent.ethz.ch

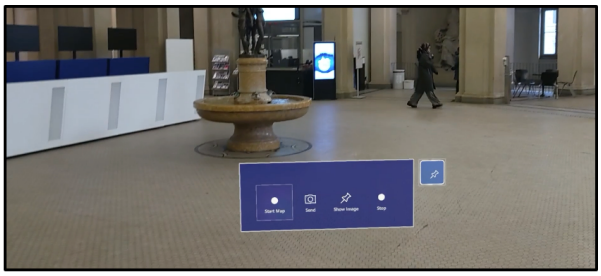
Abstract

Indoor localization and navigation are useful for users to find their way in complex buildings. In this project, we present an Hololens application to support indoor floorplan localization and navigation. We design a frontend interface that displays the floorplan minimap, showing the current location, and navigating users to the user-defined destination. There is a backend server that receives images to locate the current pose and solves the navigation problem based on the user-defined destination. The proposed approach has been tested on the large-scale dataset of ETH buildings and shows good performance.

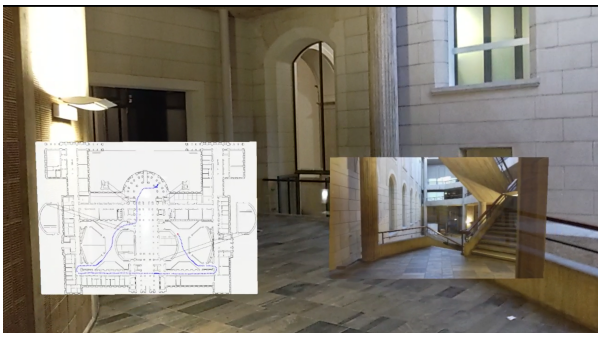
1. Introduction

People always find themselves not only lost in big cities but also sometimes lost inside buildings, including but not limited to libraries, hospitals, shopping malls, etc. These buildings can be extremely big and have complex layouts. Therefore, there is a big need to develop an assistant system for visitors to localize and navigate themselves in complex buildings. A common way is to place some floorplans with their current location at some critical point in the building. However, this is usually not enough for continuous navigation as there are just too few of them. It is also very hard for GPS-based mobile applications (like Google Maps) to work in indoor scenes.

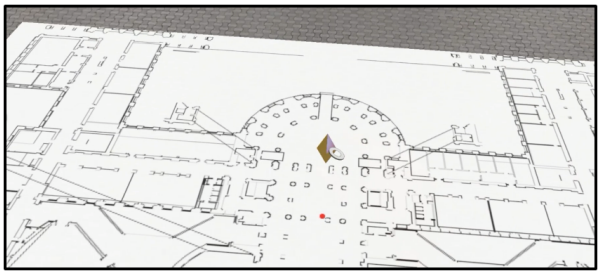
In order to provide a possible solution for the above-mentioned problems of indoor navigation, we develop a purely vision-based mixed-reality App to place minimaps, trajectories, and navigation information on the Hololens2. The HoloLens2 is known as a mixed-reality device that blends digital worlds into the real world. With the HoloLens2, we can bring applications and objects into the world around users to understand and interact with the environment.



(a) Control Panel.



(b) Trajectory and navigation route.



(c) Choose navigation goal.

Figure 1. Screenshot of our application.

In this report, we present how we develop the App that displays the trajectory on the floorplan minimap, and provides navigation capabilities with the HoloLens2. As shown in Fig. 1, a floorplan minimap is displayed in the environ-

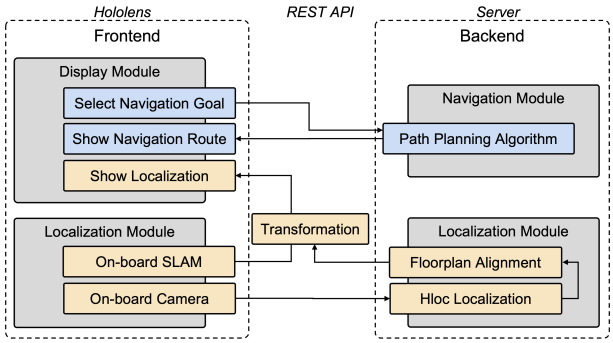


Figure 2. System structure of floorplan localization and navigation.

ment and the past trajectory is automatically updated on the minimap to help users localize themselves. Users can also drag the floating diamond to select a navigation destination. After a destination is chosen, the App will display a route to the destination so that users can follow the path to the desired location.

The system structure is shown in Fig. 2. We create an interface in the Hololens2 to display the current location, past trajectory, and navigation result on the floorplan minimap. Additionally, the user can also choose their own navigation destination. Considering the computation insufficiency of the Hololens2, we choose to use a backend server to process localization and navigation information. To be specific, for the localization service, we combine the onboard SLAM system with the backend server to ensure high precision and real-time performance. The backend server receives images from Hololens every second and localizes them using the visual localization method. This position information is used to align the Hololens2 local trajectory to the floorplan and correct the drift. For the navigation service, the backend server receives the destination from the Hololens and works as the basis to conduct a path planning algorithm and send back the navigation path.

2. Related Work

Since Microsoft HoloLens first appeared in 2016, HoloLens has been used in various industries, such as the medical and healthcare, robotics, industrial engineering, etc. [9] Microsoft HoloLens2 is equipped with various sensors, including four tracking cameras and a ToF range camera. The sensor images, poses, and intrinsics can be accessed by the user under research mode [16]. Moreover, Hololens provides developers to explore the expanding world of Mixed Reality applications with the Mixed Reality Toolkit (MRTK), Windows Mixed Reality, Unity, Unreal, and more for HoloLens and Windows Immersive Headsets. [16] These properties make HoloLens potentially interesting as an indoor mapping and localization device. Several localization and navigation applications are developed in the past years [3, 6, 7, 18].

Localizing a user within the environment is important not only for navigation systems but also in general for any location-aware application. Unfortunately, GPS cannot be used indoors because the satellite signal is usually unavailable when inside a building. A large body of work investigates alternative technologies for localizing users when they are indoors. A widely used strategy is visual localization, the method of estimating the position and orientation from which an image was taken. It is a vital component in many computer vision and robotics scenarios and benefits long-term localization algorithms on generalization to unseen scene conditions. Visual localization methods are classified into structure-based methods [13, 14] and image-based methods [15, 17]. The former performs direct matching of local descriptors between 2D key points of a query image and 3D points in a 3D SfM model. These methods are able to estimate accurate poses, but often rely on exhaustive matching and are thus compute-intensive. The image-based methods are related to image retrieval and can only provide an approximate pose up to the database discretization, which is not sufficiently precise. Hierarchical localization [4, 5, 11] takes advantage of the above approaches by dividing the problem into a global, coarse search followed by a fine pose estimation. Additionally, learned local descriptors, such as SuperPoint [12], DELF [8], etc., have been applied to improve robustness and accuracy.

3. Method

The system architecture is shown in Fig. 2. The whole system is divided into frontend and backend parts. The frontend runs on the Hololens, including the display module and the frontend localization module. In the display module, we develop an interface that allows users to select a destination and displays the floorplan map and paths. When the user chooses to display the map, the multi-threaded asynchronous localization is automatically started. On one thread, the local localization information is read from the onboard SLAM to update the display position. On the other thread, an image captured by the Hololens' front RGB camera is sent to the backend to do the global localization. When the user selects a target point, the frontend sends the position of the target point to the back end for processing.

The backend runs on the server, including the navigation module and the backend image localization module. When the navigation module receives the destination, it starts to run the path-planning algorithm and returned the planed path to the frontend. When the backend image localization module receives an image, it starts to run the hloc [10] algorithm to calculate the location and send it back to the frontend.

3.1. Hololens App Design

This part mainly describes the user interface design of this APP.

When you start the APP, you will see a control pannel with four buttons as in figure 1a.

The first button is in charge of controlling the localization function. As soon as it is pressed, the localization starts. At first, it needs some time to initialize. When the initialization stage finishes, the floorplan minimap will be displayed at the bottom left corner of your view like in 1b. The trajectory will be displayed on it and is updated in real-time. You can press it again to hide the minimap, but the localization will continue in the background.

The second button is in charge of the navigation function. After initialization, if you press it, the floorplan minimap fixed in the view will be hidden. Another horizontal map with a destination choice indicator (the yellow diamond) will appear in front of you and will be fixed in the real world like in pic 1c. You can choose your destination by dragging the indicator. The red dot will follow the movement of the indicator, which represents the destination you pick on the map. Press the second button again to send your destination to the backend server. The floorplan minimap will show up again to replace the horizontal bigger map. The navigation path will be displayed on it after the result is returned.

The third button is in charge of debugging. When you press it, a window showing the last image sent to the backend server for localization will be displayed at the bottom right corner of your view as in figure 1b. You can also press it again to hide the debugging window.

The forth button is exiting the APP.

3.2. Communication

Representational state transfer (REST) API design is a standard guideline for creating stateless, reliable web APIs. Although our application only requires a very simple interface to use the computation power of the backend server, we still follow the REST design rules for possible expansion in the future. Now, the backend provides two interfaces, one for localization and one for navigation.

The localization interface accepts a POST HTTP request with multipart Form data, which includes a string containing the intrinsic of the camera and an image file. The server decodes the string to get intrinsic, checks the format of the received file, reads the image, and sends them for localization. After execution, the localization result is sent back to the front end in string.

The navigation interface also accepts a POST HTTP request, but only with JSON data indicating the starting and target position for the navigation algorithm. The result is also sent back to the front end in a string containing the trajectory or an error message.

For the implementation detail, the back end is implemented in Python in consistency with the localization system, and the API is implemented with FastAPI framework and a Uvicorn server. In the front end, the HTTP requests are sent with HttpClient class in System.Net.Http C# library.

3.3. Asychronous Localization

The biggest challenge of the application is to achieve real-time updating of the user's location. Unlike navigation applications in the outdoor environment, indoor navigation requires a more precise location and therefore less updating latency. But in our case, as the Hololens have very limited computation power, the localization algorithm has to run on a back-end server. And as indoor localization is based on visual-based methods that have high computational complexity, it naturally has big latency even without considering the network communication.

Fortunately, Hololens provides an onboard SLAM algorithm which is a basic component of Hololens and constantly running in the background. From that, a local motion can be obtained in realtime. Based on this, we propose an asynchronous localization method to integrate the local positions from the onboard SLAM system and the global positions from the backend server.

Asychronous Localization Pipeline is a multithreading process on Hololens, as shown in Fig. 3. In our system, the frontend part has a global localization thread and a local localization thread.

There are two steps in the global localization thread to locate an image on the floor plan. The first step is to localize the image in a prebuilt 3D map of the building, in which we calculate the transformation matrix $\mathcal{T}_{WC}$ from the camera coordinate to the world (3D map) coordinate. The second step is aligning the 3D map with the floorplan through the transformation matrix $\mathcal{T}_{FW}$ from the world coordinate to the floorplan coordinate. The final output of the global localization thread is the transformation matrix $\mathcal{T}_{FC} = \mathcal{T}_{FW} * \mathcal{T}_{WC}$ from the camera coordinate to the floorplan coordinate.

In the local localization thread, it continuously records the pose of the Hololens in the Hololens' local coordinate. The poses are represented in the form of the transformation matrix $\mathcal{T}_{HC}^{(i)}, i = 0, 1, \ldots$ from the camera coordinate to the Hololens' local coordinate. This recording process works in real-time.

At the very beginning of the localization t_0, the global localization thread starts the initialization by taking a picture and sending it to the backend. It waits until $\mathcal{T}_{FC}$ is returned at time t_1. While waiting for the return, the local localization thread keeps recording all the local poses $\mathcal{T}_{HC}^{(0)}, \mathcal{T}_{HC}^{(1)}, \ldots, \mathcal{T}_{HC}^{(n_1)}$. After $\mathcal{T}_{FC}$ is known, we can calculate the transformation matrix $\mathcal{T}_{FH}$ from the Hololens

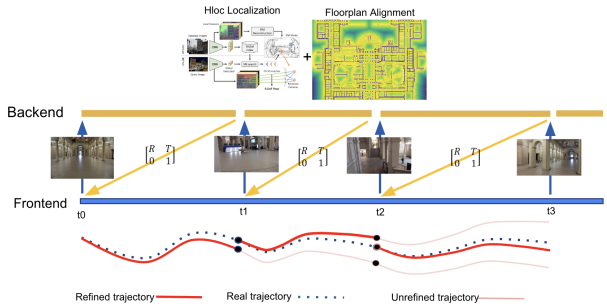


Figure 3. Asynchronous localization.

coordinate to the floorplan coordinate by $\mathcal{T}_{FH} = \mathcal{T}_{FC} * (\mathcal{T}^0_{HC})^{-1}$. It is then applied to all local poses to draw them on the floorplan via $\mathcal{T}^{(i)}_{FC} = \mathcal{T}_{FH} * \mathcal{T}^{(i)}_{HC}, i = 1, \ldots, n_1$.

At the time t_1, a new picture is taken and sent to the backend. The old transformation $\mathcal{T}_{FH}$ is applied to the new poses $\mathcal{T}^{(i)}_{HC}, i = n_1 + 1, \ldots, n_2$ recorded by the local localization thread to draw them on the floorplan minimap. At time t_2 when the new $\mathcal{T}_{FC}$ is returned, $\mathcal{T}_{FH}$ is then updated. In the meanwhile, all the poses between time t_1 and t_2, $\mathcal{T}^{(i)}_{HC}, i = n_1 + 1, \ldots, n_2$, are adjusted according to the new transformation $\mathcal{T}_{FH}$.

In this way, we can update the trajectory in real-time using local localization. The error will not accumulate through time because of the initialization error and the drift from the local trajectory by updating the transformation matrix. The frequency of global localization can be adjusted to exploit the backend computation power, and here we simply start a now global localization task when the last task finishes.

There is also a rejection mechanism to reject wrong localization results from the backend. As the Z axis of the Hololens, the 3D map, and the floorplan are aligned, the $\mathcal{T}_{FH}$ is indeed a $SE(2)$ transformation. So we check the position difference and the rotation difference between the old and new $\mathcal{T}_{FH}$. The threshold is currently set to 3 meters and $10°$.

Asychronous Implementation To implement the asynchronous pipeline, we choose to use the job system of Unity which can create, manage, and execute jobs. One challenge in implementing the pipeline is the job system limits the job can only access blittable data which doesn't need conversion when passed between managed and native code, which means no reference, no string, and even no char can be passed, and stored in the job or passed back from the job. Fortunately, the problem is solvable in our case, what we need to send is an image that can be converted to a byte array, a URL consisting of IP which can be sent as several integers, and intrinsic values which can be sent as several float values.

In the pipeline, a job and only one job need to be scheduled as initialization, this is realized by checking an initialization flag. Possible failure of the initialization also needs to be considered and a new initialization needs to be done. After initialization, the program will check whether the scheduled job is finished by checking the flag of the job handle provided by the system. If the job is finished, the result will be read and processed, and a new job will be scheduled.

Local Localization leverages the results of Hololens built-in SLAM algorithms to obtain the motion of the device. There is an onboard SLAM algorithm running in the background of Hololens2 system using the four gray-scale cameras on it. The pose of the coordinate is the pose when you start the APP, except that the Y axis is aligned with the gravity direction. The pose of the Hololens2 can be obtained in real time so that we can track the trajectory continuously.

Global Localization includes the image localization part and the floorplan alignment part.

The image localization part uses the visual localization method on the backend server to produce accurate and robust localization results of the image. Here We applied the hierarchical localization toolbox (hloc) [10] for global location. Hloc is a learning-based modular toolbox for state-of-the-art 6-DoF visual localization which leverages both image retrieval and feature matching based on SuperGlue [12].

We first use hloc [10] to build a reference 3D SFM model from the mapping dataset before the whole algorithm starts. The mapping process first extracts Superpoints [12] features of all images and finds covisible database images with image retrieval, then matches these database pairs with SuperGlue and triangulates a new SFM model with COLMAP. We use the ETH-Microsoft Localization Dataset [2] as the mapping dataset, which includes images captured at the HG building of the ETH Zurich campus by the 6-camera rig of a NavVis M6 mobile scanner.

During the localization stage, hloc [10] finds database images relevant to each query image by the retrieval and match the key points with SuperGlue. Afterward, the pose can be derived through the localization algorithm.

The floorplan alignment computes the transformation $\mathcal{T}_{FW}$ from the world coordinate to the floorplan coordinate. We make use of the coarse-to-fine registration pipeline proposed in Aligning Lidar Scans with 2D Maps (Our project in the 3D Vision course 2022), where we extract structural features from point clouds using attributes such as density, normal, etc. In coarse registration part, we used an exhaustive search method through SE(2) space to find the transformation with the highest cross-correlation score. In fine-registration part, we build a cost function based on facade distance to achieve a more precise alignment using ceres solver. This approach has been tested on the large-scale dataset of several ETH buildings and shows high accuracy

in each case. This process is an offline process that is computed in advance.

3.4. Navigation

Path Planning is one of the most crucial research problems in robotics from the perspective of the control engineer. Many problems in various fields are solved by proposing path planning. It has been applied in guiding the robot to reach a particular objective from very simple trajectory planning to the selection of a suitable sequence of action. Path planning cannot always be designed in advance as the global environment information is not always available a priori. By proposing a proper algorithm, path planning can be widely applied in partially and unknown structured environments.

In our implementation, we choose the Rapidly-exploring random tree (RRT) as the path-planning algorithm. RRT is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The high efficiency of the RRT algorithm makes it an ideal choice for our navigation module.

A map of that place is required to allow the RRT algorithm to work in a specific place. We demonstrate the navigation function in ETH HG with a floor plan of layer E. The floor plan is an RGB image with a size of 1659x1167. Since the RRT algorithm requires an obstacle list with the radius information included, we preprocess the floor plan to perform RRT on it. We first binarized the RGB map with a threshold. Then the binary map is further downsampled to improve computational efficiency. Finally, the points in the binary map with a value of 1 are abstracted to an obstacle list with a predefined radius.

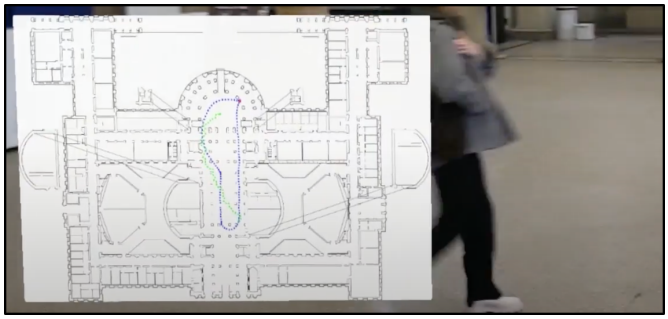
In the navigation module, two positions indicating the start point and the goal are set by the user. Then the RRT algorithm is performed to output a feasible path that can navigate the user to the destination.

4. Experiment

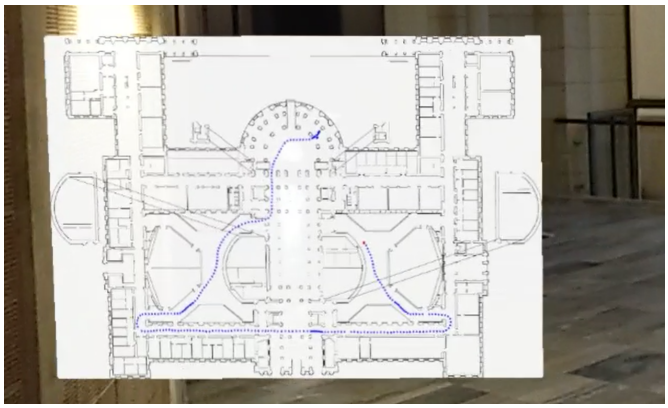
We present the performance of our App in the ETH-HG building. In the remainder of this section, we first introduce our experiment setting. Then we visualize the results of our experiments. Finally, we show some detailed intermediate results and failure cases and give some analysis of the performance.

4.1. Experiment Settings

Localization Settings For visual localization, we first crop the image to the vertical horizon. We use Netvlad [1] for image retrieval and use Superpoint [12] as local features. We set the pairs of retrieval to be 10, which means that the top ten matched images will be selected and used to extract Superpoints. We set the PnP projection threshold to 40.



(a) Experiment 1. Normal localization and navigation result



(b) Experiment 2. Large-scale localization test

Figure 4. Screenshots of experiments results. Trajectories are shown in blue dots. The navigation pathes are shown in green dots.

For local localization, we collect location data every second from the Hololens' onboard SLAM trajectory.

Navigation Settings In the implementation, we treat all the black pixels in the floor map as obstacles with a radius of 1m. The radius of the agent is also assumed to be 1m. To reduce the computational complexity, we $4\times$ downsampled the floor map before feeding it to the RRT algorithm.

4.2. Experiment Results

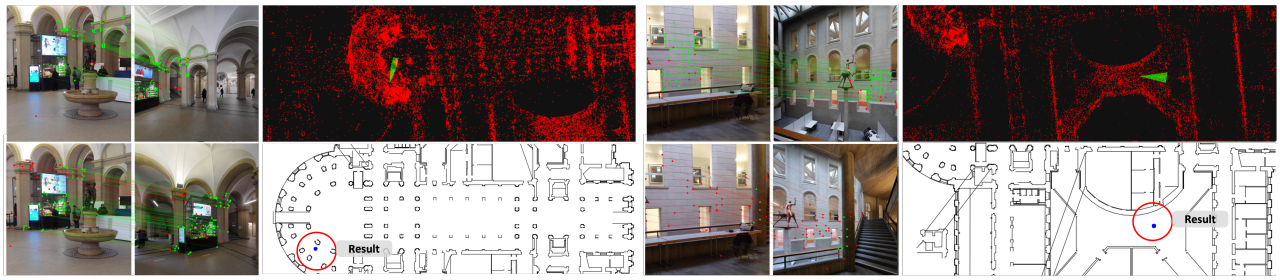
Fig. 4 shows two experiment results of the localization and navigation. In 4a, the trajectory on the floorplan minimap perfectly matches the real trajectory. The navigation function also successfully gives out the navigation path to the picked destination. When we return to the starting point, the trajectory also coincides with the origin.

We also tested the large-scale localization, and the APP also gives out a very reliable result as shown in figure 4b.

Details can be checked in the video.

4.3. Evaluation

Asychronous localization Figgure 6 shows the effect of asynchronous localization. In fig. 6a we can see some severe errors and drift. The pure local trajectory should be



(a) Success case 1. (b) Success case 2.

Figure 5. Success matching cases.

continuous. But in the red circles, we can see that the APP corrects all the trajectories after some timestamp by moving the trajectory away using the new correct transformation matrix, thus causing the gap. At the end (the upper red circle) it successfully corrects the trajectory and makes all the trajectories afterward more accurate.

Fig. 6b shows how the onboard SLAM compensates for the calculation speed of visual localization. Limited by GPU device, hloc algorithm requires about 60 seconds to process one image. In Fig. 6b, the right floating images are the images sent to the backend, which shows a large delay from the current location. As displayed on the floorplan (the left images) in Fig. 6b, the SLAM system update the location and compensates for the time delay.

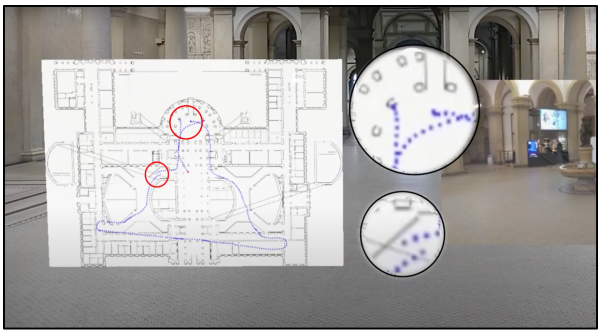
Visual Localization Fig. 5 shows hloc's successful matching results, which include enough key points for localization and correct pose in 3D point clouds and 2D maps. However, hloc sometimes fails due to motion blur, wrong matching, and lack of features. As demonstrated in Fig. 7a and Fig. 7b, motion blur and lack of features will directly lead to the failure of hloc. While the wrong matching is mainly caused by the challenging environment which contains many self-similarities and symmetric structures, as shown in Fig. 7c.

5. Conclusion

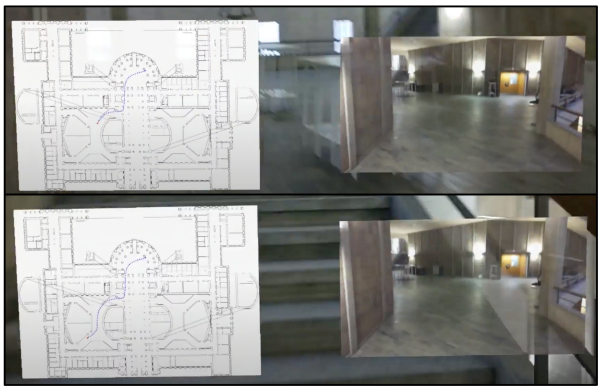
In this project, we developed an APP for indoor localization and navigation on Hololens, based on the computer-vision method. It provides a reliable solution to solve the problem of localization in indoor scenes without GPS signals. At the same time, it is very easy by rendering all the items in the real-world using mixed reality.

In the future, there is still some work we can do to improve the experience of this APP, e.g

- Improve the global localization inference speed by using the local localization information as priors.
- Show the navigation information in a mixed reality way.



(a) Trajectory correction. The global localization results are used to correct the error from local drift and the last localization result



(b) Real-time Update. The trajectory is updated even the global localization result is not returned (no new image is taken).

Figure 6. Asynchronous Localization.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016. 5

[2] ETH Zurich Computer Vision Group and Microsoft Mixed Reality & AI Lab Zurich. The ETH-Microsoft Localization Dataset. https://github.com/cvg/visloc-iccv2021, 2021. 4

[3] Patrick Hübner, Martin Weinmann, and Sven Wursthorn.

(a) Motion blur.

(b) No features.

(c) Wrong matching.

Figure 7. Failure cases.

Marker-based localization of the microsoft hololens in building models. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(1), 2018. 2

[4] Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2599–2606. IEEE, 2009. 2

[5] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-dof localization on mobile devices. In *European conference on computer vision*, pages 268–283. Springer, 2014. 2

[6] Reza Moezzi, David Krcmarik, Haythem Bahri, and Jaroslav Hlava. Autonomous vehicle control based on hololens technology and raspberry pi platform: An educational perspective. *IFAC-PapersOnLine*, 52(27):80–85, 2019. 2

[7] Reza Moezzi, David Krcmarik, Jaroslav Hlava, and Jindřich Cỳrus. Hybrid slam modelling of autonomous robot with augmented reality device. *Materials Today: Proceedings*, 32:103–107, 2020. 2

[8] Hyeonwoo Noh, Andre Araujo, Jack Sim, Tobias Weyand, and Bohyung Han. Large-scale image retrieval with attentive deep local features. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3465, 2017. 2

[9] Sebeom Park, Shokhrukh Bokijonov, and Yosoon Choi. Review of microsoft hololens applications over the past five years. *Applied Sciences*, 11(16):7259, 2021. 2

[10] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12716–12725, 2019. 2, 4

[11] Paul-Edouard Sarlin, Frédéric Debraine, Marcin Dymczyk, Roland Siegwart, and Cesar Cadena. Leveraging deep visual descriptors for hierarchical efficient localization. In *Conference on Robot Learning*, pages 456–465. PMLR, 2018. 2

[12] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020. 2, 4, 5

[13] Linus Svärm, Olof Enqvist, Fredrik Kahl, and Magnus Oskarsson. City-scale localization for cameras with known vertical direction. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1455–1461, 2016. 2

[14] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. Inloc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018. 2

[15] Akihiko Torii, Relja Arandjelovic, Josef Sivic, Masatoshi Okutomi, and Tomas Pajdla. 24/7 place recognition by view synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1808–1817, 2015. 2

[16] Dorin Ungureanu, Federica Bogo, Silvano Galliani, Pooja Sama, Xin Duan, Casey Meekhof, Jan Stühmer, Thomas J Cashman, Bugra Tekin, Johannes L Schönberger, et al. Hololens 2 research mode as a tool for computer vision research. *arXiv preprint arXiv:2008.11239*, 2020. 2

[17] Tobias Weyand, Ilya Kostrikov, and James Philbin. Planet-photo geolocation with convolutional neural networks. In *European Conference on Computer Vision*, pages 37–55. Springer, 2016. 2

[18] Mulun Wu, Shi-Lu Dai, and Chenguang Yang. Mixed reality enhanced user interactive path planning for omnidirectional mobile robot. *Applied Sciences*, 10(3):1135, 2020. 2