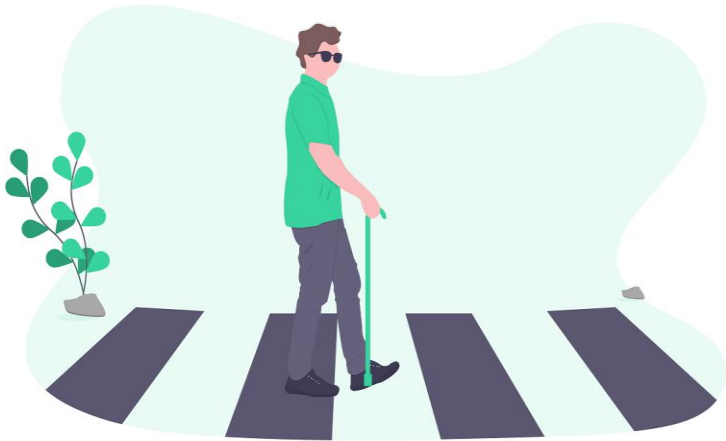


Image Based Indoor Localization

Assistech Lab, IIT Delhi

Motivation:



Blind people have difficulties in finding their way through unknown buildings. There being no special aid or instrument, whatsoever, available to them, their only ways of navigating in a new surrounding are their walking stick & constantly asking passersby for help. This deprives them of their sense of independence and freedom.

Our aim is to build a navigation system without adding much /any additional equipment to the infrastructure of the building, and unlike the systems that rely on beacons to find the location of user, thus increasing the feasibility and minimizing cost.

Approaches that didn't work (so well) !!

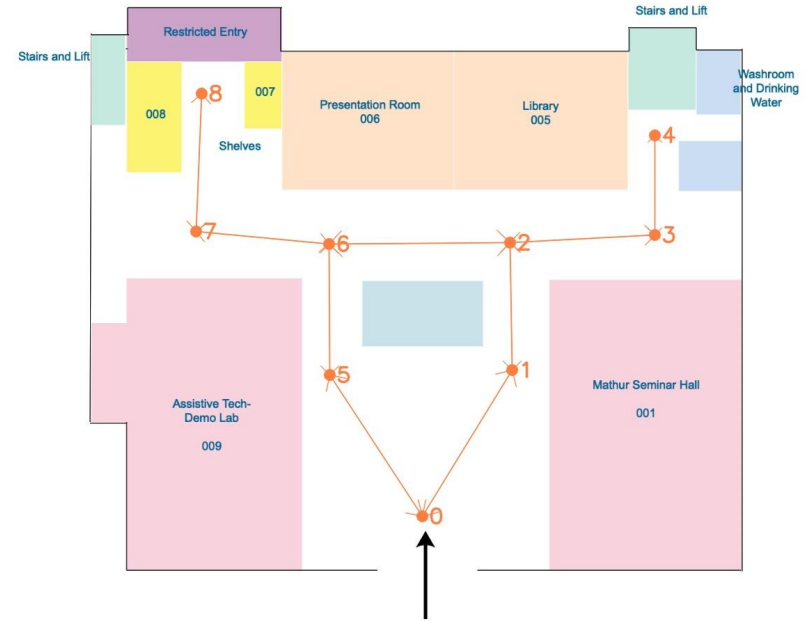
- SURF, SIFT with bf-matcher didn't give as accurate results as SURF with knn matcher.
- ASIFT which accounts for relative orientation of images being compared has very slow processing (to process a 30 sec video it took almost 4-5mins).
- KAZE with cosine distance instead of Lowe's ratio test.
- ORB with bf-matcher.
- We tried to make an efficient algorithm but it didn't work properly so we increased the search domain for a query frame



CURRENT APPROACH

Database creation

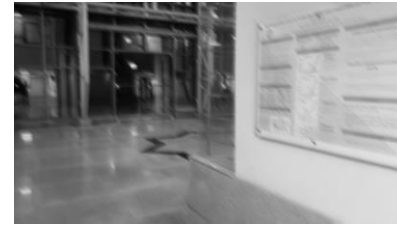
- A Graph program takes a 2D image map of a floor as input and enables the user to manually define and mark the nodes and edges on the map.
- We then create videos of each edge and store the information about the descriptors and key points of the distinct frames generated from the video.
- For 720p edge videos for SIT first floor, database size is around 20 MB.



Increasing efficiency for database creation

- Storing only distinct frames to reduce redundancy.
- Converting all images to grayscale to increase comparison speed and reduce memory usage.
- Disregarding blurry frames. They are found using variance of Laplacian. More the variance more the blurriness.
- Overlooking the frames which have features less than a particular threshold

Similar Images



Blurred Images

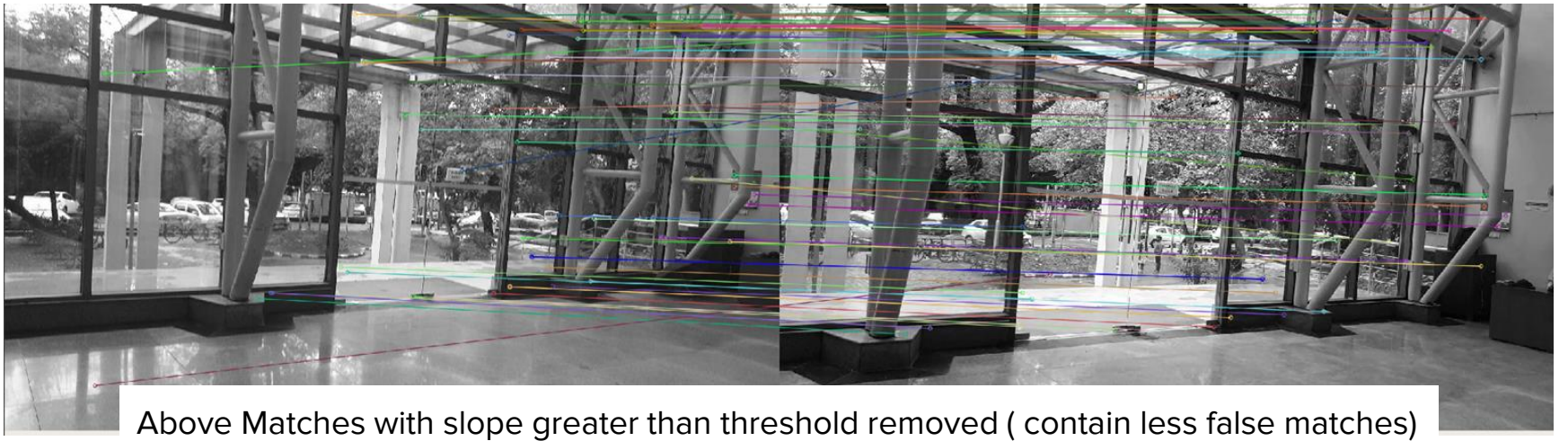
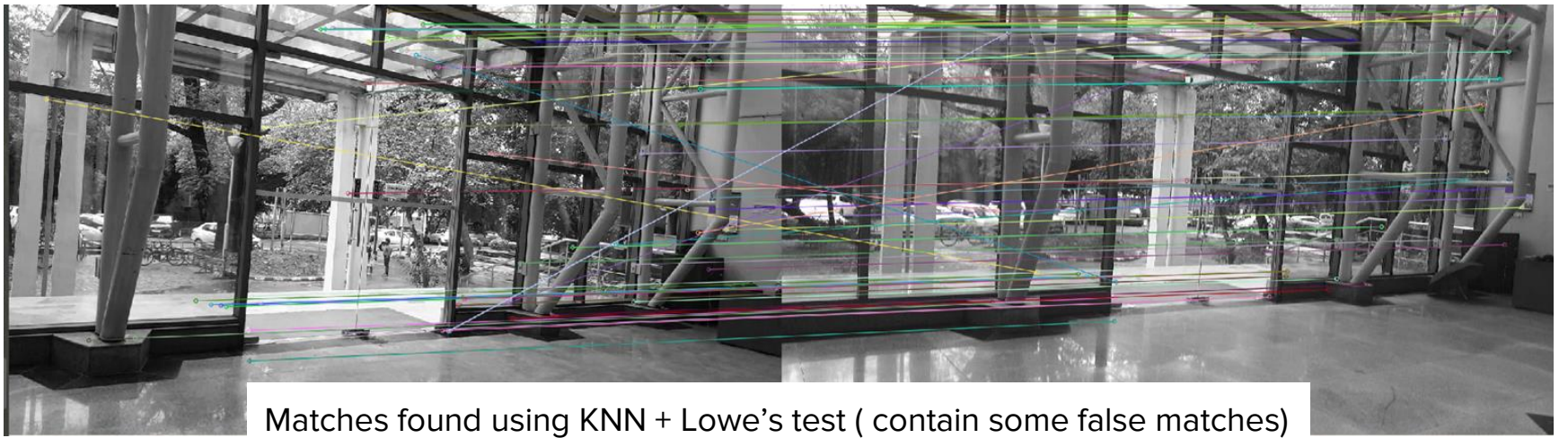


Less Feature Images

Image Comparison



- We use OpenCV's SURF to extract features from images.
- KNN matcher is used to match these features and good matches are filtered using Lowe's ratio test.
- Fraction match between two images is calculated based on no of features matched. When this fraction exceeds a certain threshold images are considered 'matched'.
- To further filter the good matches, the slope of the line connecting the corresponding features is calculated. If the slope exceeds a certain threshold, it is not a match.



Query and localisation (1)

- The user's device is used to capture a video stream and is transferred via wifi to the host system and the frame speed is taken to be 2fps.
- At each instant, a frame is captured from the video stream, converted to grayscale, checked for blurriness and features extracted.



Query and localisation (2)

- Our algorithm takes into account the best-found matches of the last 5 query frames for determining the location.
- For e.g., If the last 5 frames give matches as follows:

Query frame no.	Edge no. whose frame in database has matched	The frame no. of edge in database which has matched
60	2	1
61	2	1
62	3	5
63	None	None
64	2	1

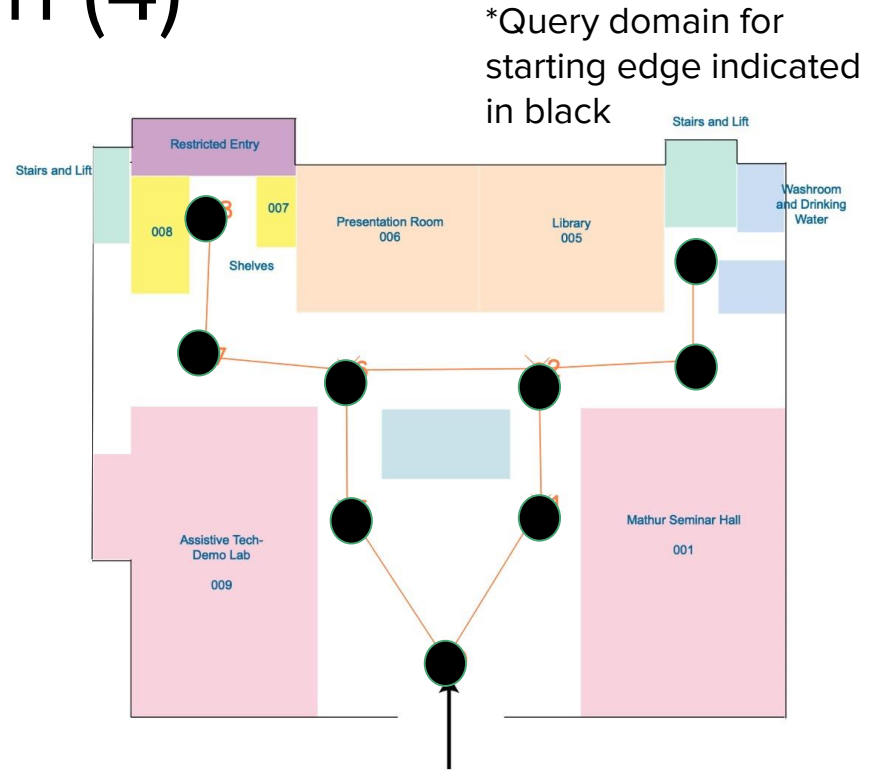
Then the current location will be declared as 1st frame in edge no. 2.

Query and localisation (3)

- Taking the most frequent matches ensures the accuracy of localisation in case :
 - there are false matches, or
 - if the user's camera is obstructed by random persons/objects for some time, or
 - if the person stops midway

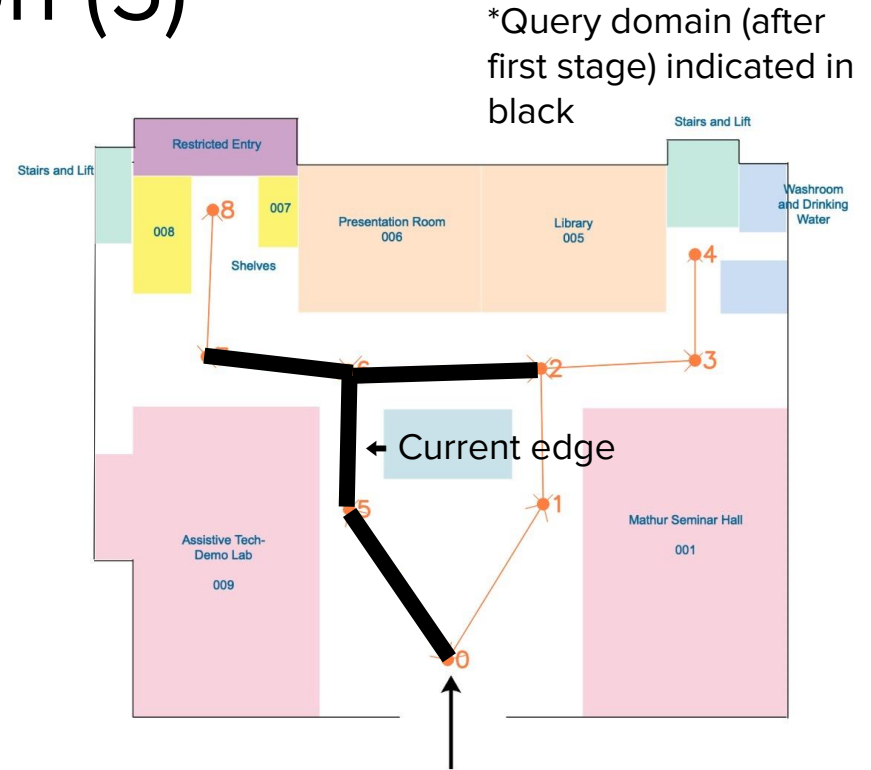
Query and localisation (4)

The query domain is determined at each stage based on the current location. For determination of the starting edge, the query frame is matched with the first frame of each edge.



Query and localisation (5)

Once the edge is found, the subsequent query frames are first compared with the frames of the found edge, and if no match is found (happens when the edge changes or the frame is random and doesn't correspond to any frame in the database), the frames of nearby edges (those sharing a common node with the current edge) are compared.



Query and localisation (6)

- The fraction of edge traversed is calculated using the timestamp of the best-matched frame (in the database) of the edge.
- For e.g., if the frames in an edge are as follows:

Frame no.	0	1	2	3	4
Time stamp	0	55	150	172	200

If the best matched frame is #2, then the fraction of edge traversed = $150 / 200 = 0.75$. This implies user is at $\frac{3}{4}$ th of the edge.

- This information of the current edge and fraction traversed is used to display the current location of the user on the map.

Activities ▾ Wed 00:26

ImageLocalisation3 [~/PycharmProjects/ImageLocalisation3] - .../localisation_final.py [ImageLocalisation3] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Numpy Horizontal Concat

Git: ↩ ✓ 🕒 🔍

---Max match for 76: (3, 0.5)

fx: 1211.v 281 ~ R:198 G:198 B:198

```

---Max match for 72: (2, '0_5')
0_5, 1
graph called
---Max match for 73: (3, '0_5')
0_5, 3
graph called
---Max match for 74: (3, '0_5')
0_5, 3
graph called
---Max match for 75: (4, '0_5')
0_5, 3
graph called

```

4: Run 6: TODO 9: Version Control Terminal Python Console

ScView Database

points, shape) of imgObj at query_index
elements()

ames

e_edges

he given query_index frame
(float).maxedoe: edge name(str)

IPv4: http://10.194.36.234:8080
IPv4: https://10.194.36.234:8080
IPv6: http://[2401:4900:ce2:72b3:0:4b:c6f::1]:8080
Video connections: 0, Audio: More...


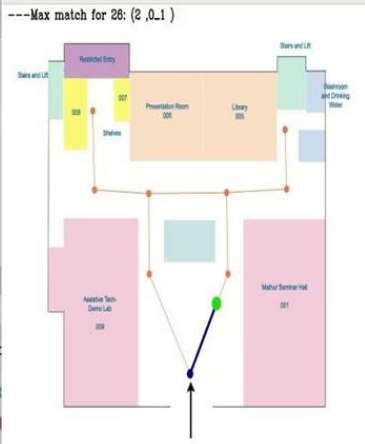
Sample Video 1

Activities Wed 00:20 ImageLocalisation3 [~/PycharmProjects/ImageLocalisation3] - .../localisation_final.py [ImageLocalisation3] - PyCharm

File Edit View Navigate Code Refactor Run Tools VCS Window Help Numpy Horizontal Concat

Git: [Icons]

---Max match for 26: (2, 0,1)



```
...points, shape ) of imgObj at query_index
elements()

ames

e_edges
```


1: Project

2: Favorites

7: Structure

```
(x: 354, y: 156) ~ R: 202 Q: 221 B: 244
---Max match for 22: (4, '0_1')
0_1, 4
graph called
---Max match for 23: (2, '0_1')
0_1, 4
graph called
---Max match for 24: (2, '0_1')
0_1, 2
graph called
---Max match for 25: (2, '0_1')
0_1, 2
graph called
```

4: Run 6: TODO 9: Version Control Terminal Python Console



IPV4: http://10.194.36.234:8080
IPV4: https://10.194.36.234:8080
IPV6: http://[2401:4900:ce2:72b3:0:4b:c6ff:a301]:8080
Video connections: 0, Audio: 0

More...

Sample Video 2

Limitations

1. The query domain is limited
2. Point localisation is not absolutely accurate
3. Lighting and background conditions
4. Using SURF for feature detection and limitation on walking speed.
5. High resolution and good quality camera is required
6. Doesn't work well in highly crowded places

1. The query domain is limited

- At each point, only the frames of current or adjoining edges can be queried due to time constraints.
- If large no of continuous frames is not matched due to irregular frames in the query video or the database, then the algorithm can catch up with the current location if the person is still on the last known current edge or its adjoining edges.
- However, if the current edge and its adjoining edge is missed completely because of no matches, then the algorithm fails because his current location will be out of the query domain.

2. Point localisation is not absolutely accurate

- Though it yields good results, the representation on the map is only indicative of the best-matched frame in the current edge.
- Its accuracy depends on the best-found match among the frames of the edge, and the density of the database.

3. Lighting and background conditions

- Because image matching does not yield good results in varied lighting conditions, localization is reasonable only if the lighting conditions are somewhat similar at the time of database creation, and at the time of the query. Also the working is best at night when lighting conditions are even throughout.
- In case of buildings like SIT, where sunlight plays a heavy role in lighting during the day, separate databases have to be created for the morning, afternoon and evening for good results.
- Also in SIT, the main gate adjacent to wall is completely made of glass due to which the background changes are very drastic in the edge facing the wall due to which the detection is not proper.



4. Using SURF for feature detection and limitation on walking speed

- Though SURF gives reasonable results, it is far from perfect.
- It is inconsistent in terms of features detected and gives very low percentage match for even very similar images
- The walking speed of the person must be limited to 0.5-0.7 of normal walking speed of average human being due to hardware constraint on processing.

5. High resolution and good quality camera is required

- Camera should be preferably more than 2 mp
- Preferred video resolution is 720p
- Camera with optical stabilization is required
- Camera with more dynamic range handle image matching better

6. Doesn't work well in highly crowded places

- Image matching is not good if there is a lot of crowd in the testing time
- This happens because the crowd covers the major section of the stable background and the matcher is not able to give good results.

References

1. Blur detection with OpenCV

<https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/>

2. More research in image-based localisation

<https://pdfs.semanticscholar.org/c695/642def74bbd772ab39a3f1e592937fd87a5d.pdf>

<https://www.sciencedirect.com/science/article/pii/S0167865515000744>

1. Asift matcher

<https://github.com/opencv/opencv/blob/master/samples/python/asift.py>

1. Converting image to bag of words using KMeans on Surf Descriptors and training svm to generate classes to group similar images.

<https://kushalvyas.github.io/BOV.html>

Future Improvements

- We can use pedometer and phone compass along with our algorithm to improve results.
 - We can also place QR codes on important locations to fine detect the location further and even improve accuracy of our algorithm
-

Thank you