



T21: Technocrats

Automated Testing of Virtual Reality applications

Team Members

- Lakshmi Priya B (Team Leader)
- Rithik Dutt
- Ayush Kumar Mehta



Problem Statement

To provide solutions for testing Virtual Reality applications in an automated manner in order to reduce the overall development lifecycle.

The VR application testing requires a physical testing process which is generally done manually. The process can be automated by using several programmed methodologies. The best way or proposed way would be mechanical by programming all the movements and tracks into it but again it would be a hectic task to program in such a way.



Solution

The solution proposed by us to automate VR testing is to capture the screenshot of the user viewable segment of virtual reality environment at initial instance before the stimulated mechanical arm movement and at a later instance after the stimulated mechanical arm movement and test this against the projected 2D-image of the virtual reality environment based on three main criteria:

- Distortion
- Movement
- Unexpected behaviour



Example



Final View



2D image of VR environment

Initial View

ACTION: Device movement performed by user

RESULT : Change in user's viewable area
Initial view -> Final view

SENSOR READINGS:

x: centroid displacement along x axis after movement
y: centroid displacement along y axis after movement

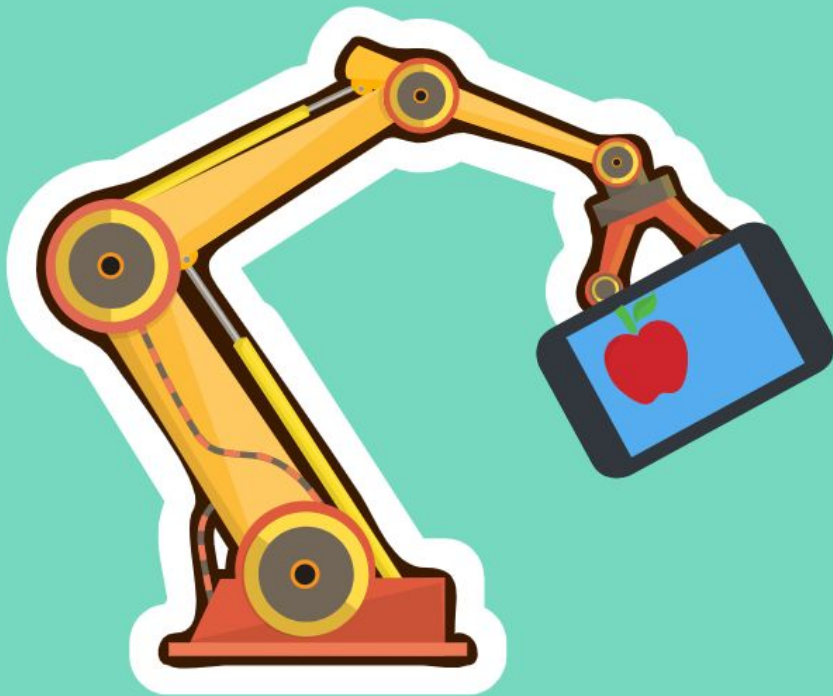
VALUES STORED:

- 2D image of Virtual Reality environment
- Initial view
- Final view
- Sensor readings

Testing based on 3 main criteria:

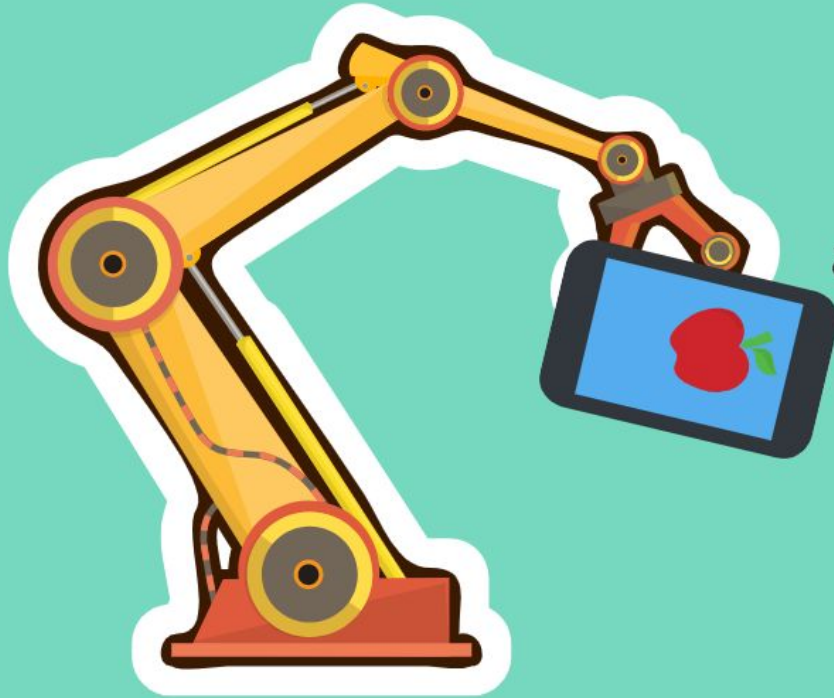
1. Distortion
2. Movement
3. Unexpected behaviour





Capture
Initial view
screenshot
before device
movement



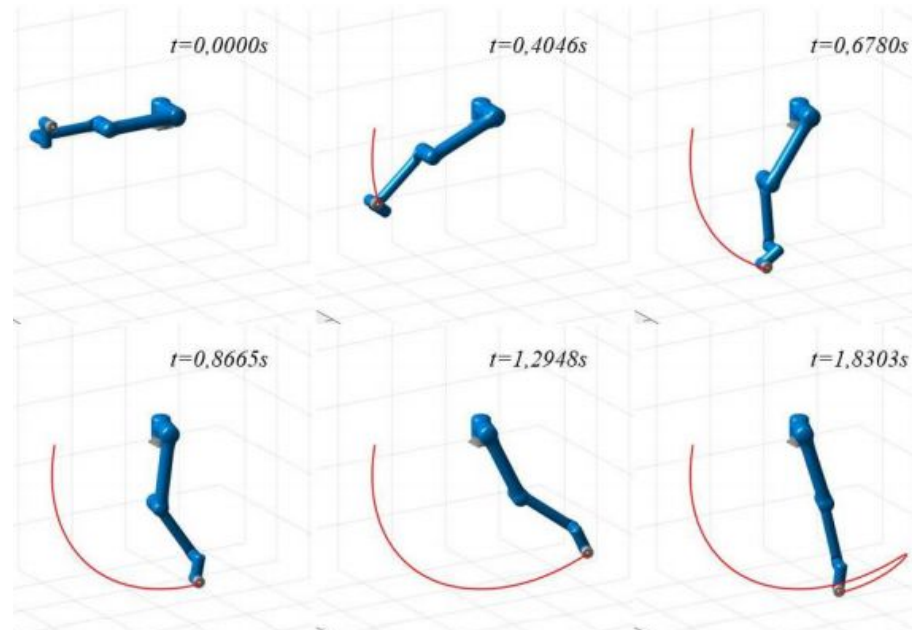


Capture Final view
screenshot and
displacement reading
using sensor
after device movement



Step 1

Automate Test case Generation

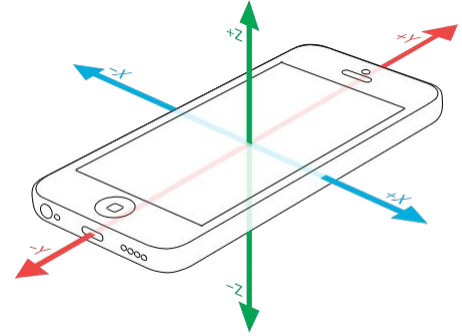


To automate test data generation:

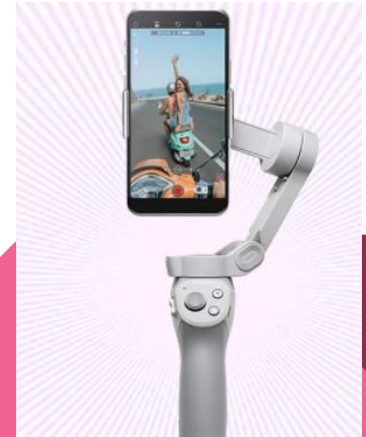
- Stimulate mechanical arm movement
- Store screenshot at instance t_{initial}
- Store sensor readings for distance moved in x direction and y direction
- Store screenshot after movement at instance $t_{\text{initial}} + t$

Virtual Reality user experience stimulation

A mechanical robotic arm with sensor attached can be used to stimulate user movement during virtual reality user experience stimulation




A sensor like gyroscope, accelerometer, etc is used for calculating centroid displacement from initial instance screenshot to final instance screenshot in x direction and y direction



Step 2

Preprocessing Generated Test Cases

 input - Notepad

```
File Edit Format View Help
vr2.jpg initial2.jpg final2.jpg -200 -20
vr2.jpg initial2_r.jpg final2_rotated.jpg 370 -10
vr2.jpg initial2.jpg final2.jpg 1 1
vr2.jpg initial2_distorted.jpg final2.jpg 100 100
vr2.jpg wrong.png final2_rotated.jpg 1 1
```

Store testcases in format:

2D_image_of_environment <space>
Initial_view_image_name <space>
Final_view_image_name <space>
x_displacement <space>
y_displacement <enter>

[Next entry...] in file

Step 3

Automated VR application
Testing



Approach towards automated testing

Approach 1: Template Matching

Approach 2: Multiscaling mechanism + Template Matching

Approach 3: Feature Matching + Homography

Approach 3 is selected as it is the most suitable APPROACH for our need



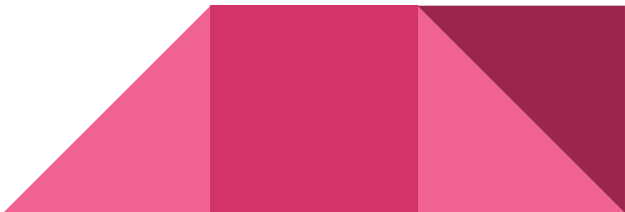
Reasons for choosing Approach 3

Template Matching (Approach 1) and Multiscaling mechanism + Template Matching (Approach 2) **does not work for rotated or scaled versions of the template** as a change in shape or size of object with respect to template will give a false match.

As rotations and scaled versions of the template is very common in case of Virtual Reality applications, **Feature Matching + Homography** (Approach 3) is used which **works for rotated and scaled versions of the template** thereby giving a correct match in these cases.



Feature Detection

- Mathematical representations of key areas in an image are the features
 - Features are the vector representations of the visual content from an image so that we can perform mathematical operations on them
 - Features from an image plays an important role in computer vision for variety of applications including object detection, motion estimation, segmentation, image alignment and a lot more
 - Features may include edges, corners or parts of an image
 - Feature detection algorithm SIFT (Scale-Invariant Feature Transform) can be used to detect features
- 

Scale-Invariant Feature Transform

- SIFT is both rotation as well as scale invariant
- SIFT provides key points and keypoint descriptors where keypoint descriptor describes the keypoint at a selected scale and rotation with image gradients
- The output image has circles depicting the key points/features, where size of the circle represents the strength of the key point and the line inside the circle denotes the orientation of the key point



Feature Matching

- Feature matching is used to match features in one image with others
- Feature matching between images can be done with:
 - Brute-Force matcher
 - FLANN based matcher



FLANN based matcher

- Fast Library for Approximate Nearest Neighbors (FLANN) is optimised to find the matches with search even with large datasets
- It is fast when compared to Brute-Force matcher
- It contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features



Step 4

Output Consolidated Testing Report

Test with valid and invalid testcases.
Output test result as any one of the following:

- "FAILED: Distortion/Unexpected Error"
- "FAILED: Movement Error"
- "PASSED: Success" for each of the testcase

Implementation

The automated testing is implemented using Python modules:

- cv2
- numpy
- matplotlib

Testing example



2D image of Virtual Reality Environment

Testing example



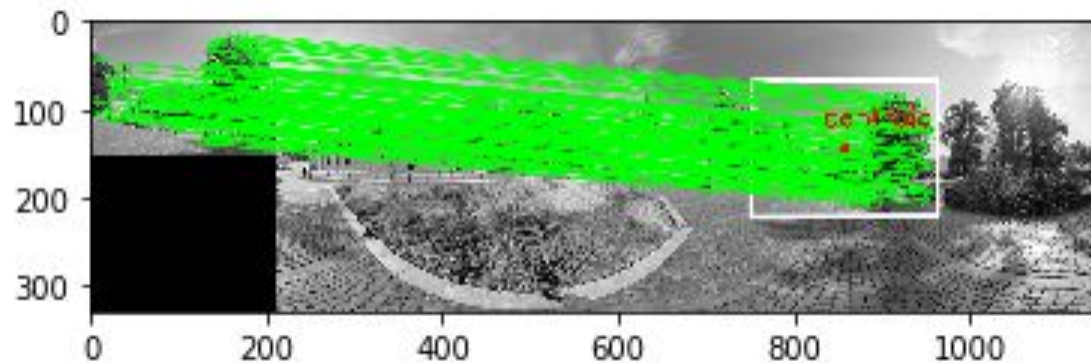
Initial view



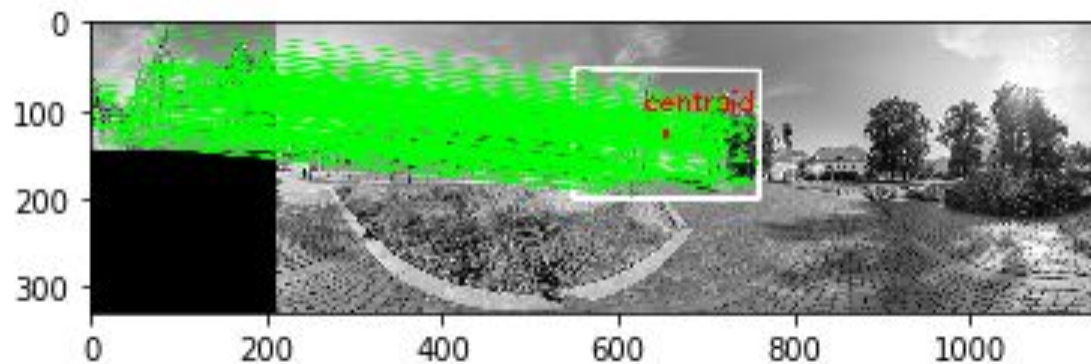
Final view

Distances moved in x and y directions to get final view from initial view:
(-200, -20)

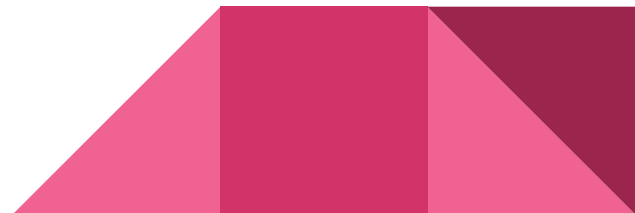
Testing example



Initial view matching in 2D image of VR environment



Final view matching in 2D image of VR environment



Test result of example

```
[[[540  68]]
```

```
[[539 221]]
```

```
[[751 220]]
```

```
[[751  68]]]
```

```
centroidX: 857
```

```
centroidY: 144
```

```
[[[337  54]]
```

```
[[337 199]]
```

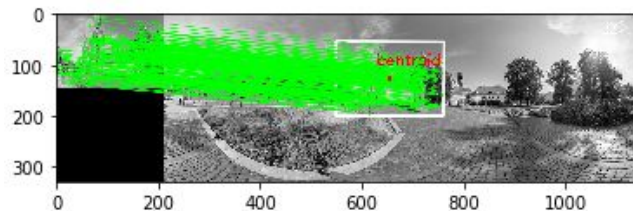
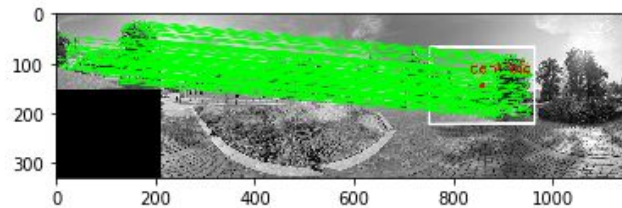
```
[[548 199]]
```

```
[[549  55]]]
```

```
centroidX: 654
```

```
centroidY: 126
```

TEST CASE PASSED!



Testing for Movement

```
[[[540  68]]]
```

```
[[539 221]]
```

```
[[751 220]]
```

```
[[751  68]]]
```

```
centroidX:  857  
centroidY:  144
```

```
[[[337  54]]]
```

```
[[337 199]]
```

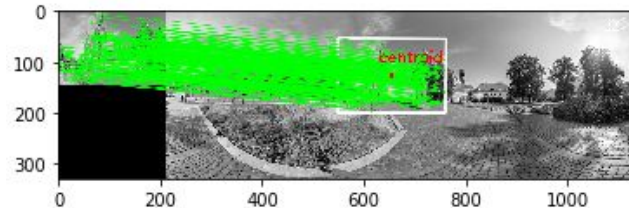
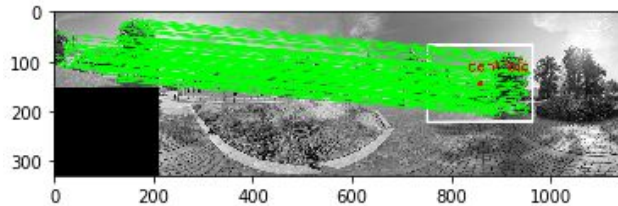
```
[[548 199]]
```

```
[[549  55]]]
```

```
centroidX:  654  
centroidY:  126
```

```
Movement error!!  
TEST CASE FAILED!!!!
```

Test case fails if the difference between centroid displacement calculated by testing module and that procured from sensors is greater than the defined acceptance margin



Testing for Movement

```
[[[ 14  91]]]
```

```
[[ 14 222]]
```

```
[[102 221]]
```

```
[[103  92]]]
```

```
centroidX: 147  
centroidY: 156
```

```
[[[307 160]]]
```

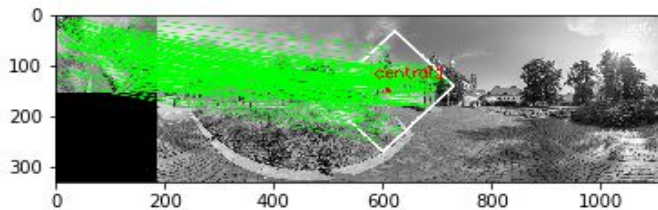
```
[[415 268]]
```

```
[[545 138]]
```

```
[[437  30]]]
```

```
centroidX: 610  
centroidY: 148
```

TEST CASE PASSED!



Testing for Distortion

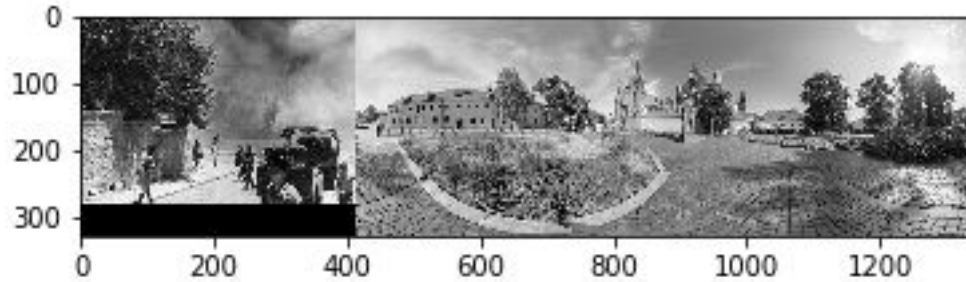
Not enough matches are found - 1/20



TEST CASE FAILED!!!!

Testing for Unexpected Behaviour

Not enough matches are found - 0/20



TEST CASE FAILED!!!!

Testing with invalid initial frame

Automated Testing - Consolidated Testing Report

TESTED WITH VALID AND INVALID TEST CASES

*****CONSOLIDATED TESTING REPORT*****

Testcase	0	:	PASSED: Success
Testcase	1	:	PASSED: Success
Testcase	2	:	FAILED: Movement Error
Testcase	3	:	FAILED: Distortion / Unexpected Error
Testcase	4	:	PASSED: Success
Testcase	5	:	FAILED: Distortion / Unexpected Error
Testcase	6	:	PASSED: Success



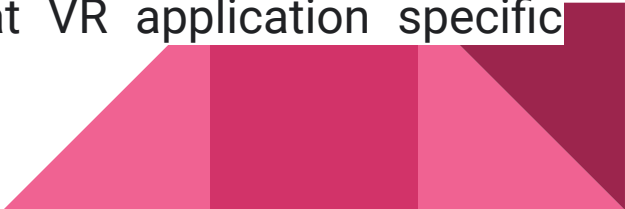
Infosys VR

- Investment in mixed reality (combination of AR & VR) is exploding across industries.
- Infosys has a team of experienced engineers, graphic artists, designers, and video animators that will work with user to explore this new wave of technology with a specific focus on user's needs and the best way to build a solution to solve user's problems.
- Infosys team has over 60 years of combined experience in feature films, large scale VR experiences, enterprise AR solutions simulators, and video game market. Their projects cover manufacturing, aerospace, retail, real estate, education, sports, healthcare, etc.
- Using Design Thinking techniques, Infosys works with business and subject matter experts to understand user's goals and challenges and define the problem statement.
- It also works on ideation, storyboarding, prototyping and user testing, engaging the user in every step of the way, to ensure that they deliver an immersive and compelling experience.
- **So, there is an urge to improve and automate the VR testing further in order to ensure customer satisfaction and thrive amongst all the other competitors.**

Support for proposed automated VR testing solution

- The virtual reality market is estimated at USD 6.1 billion in 2020 and is expected to reach USD 20.9 billion by 2025. Easy availability of affordable VR devices is the key factor driving the adoption of VR devices. Therefore, proposed automated testing solution is **desirable** as it saves cost, time and work expended in manual testing.
- The proposed automated testing solution is **feasible** to build with existing technologies like mechanical robotic arm, sensor, image processing models, etc. The amount spent for these technologies is recovered due to completely automated cheaper testing.
- The proposed automated testing solution is **viable** now and in the future as VR domain is in its developmental phase which requires intensive testing support now and in the future.


Scalability

- Other testing criteria can be added in addition to the 3 main criteria chosen. These new criteria can also be specific to the VR application being tested. Addition of valid criteria increases the efficiency of the model further.
 - Mechanical arm movements can be made to exactly replicate real world scenario so that testing process is in line with real world use of VR application.
 - The number of automated test cases generated can also be tuned as per the requirement. This in turn influences the testing duration and testing accuracy.
 - The solution proposed can be scaled up for complex VR environments by clearly deciding upon mechanical arm operations and that VR application specific testing criteria.
- 

Algorithm

1. To generate test case:
 - 1.1. Stimulate mechanical arm movement
 - 1.2. Store screenshot before movement
 - 1.3. Store sensor readings for the distance moved in x direction and y direction
 - 1.4. Store screenshot after movement
2. Take as many test cases as desired and store the details in the format:
Name_of_2D_Image_of_environment <space> Name_of_inital_view_image <space>
Name_of_final_view_image <space> x_displacement <space> y_displacement <enter>
[Next entry.....] in file "input.txt"



3. To test:
 - 3.1. Take the query image (screenshot of instance) and convert it to grayscale
 - 3.2. Initialize the SIFT detector
 - 3.3. Detect the keypoints in query image and scene with SIFT
 - 3.4. Compute the descriptors belonging to both the images with SIFT
 - 3.5. Match the keypoints using FLANN Based Matcher
 - 3.6. Store all the good matches as per Lowe's ratio test
 - 3.7. If the good matches > MIN_MATCH_COUNT:
 - 3.7.1. Compute the centroid of query image
 - 3.7.2. Show the matched images with marked centroid
 - 3.7.3. Return the computed values
 - 3.8. Else:
 - 3.8.1. Declare that "TEST CASE FAILS" as this is the case of DISTORTION or UNEXPECTED BEHAVIOUR and exit
- 

4. To perform automated testing:
 - 4.1. Extract the test cases from “input.txt” file and store each test case as a list entry
 - 4.2. For each test case:
 - 4.2.1. Perform step 3 with the initial frame before movement and VR environment image as input to get ix, iy which denotes the centroid of the matched initial frame in VR environment image
 - 4.2.2. Perform step 3 with the final frame after movement and VR environment image as input to get fx, fy which denotes the centroid of the matched final frame in VR environment image
 - 4.2.3. Use the detected sensor movement distances x, y to check the condition:
 $|fx - ix - x| \leq \text{ACCEPT_MARGIN}$ AND $|fy - iy - y| \leq \text{ACCEPT_MARGIN}$
 - 4.2.4. If the condition is satisfied, then the “TEST CASE PASSES”
 - 4.2.5. Else, “TEST CASE FAILS” as this is the case of INVALID MOVEMENT
 - 4.3. Print consolidated testing report
5. The various tunable parameters like MIN_MATCH_COUNT, ACCEPT_MARGIN, etc. can be altered based on the application which is being tested

Overview of main module

1. Detect Keypoints in image and template
2. Create Descriptors from the Keypoints
3. Matches the descriptors of image and template
4. Keep good Keypoints from the matches
5. Find object from the good Keypoints



Submissions

Video:


<https://drive.google.com/file/d/1jY454wbTX1cj-q6ohU5u4k2SjgNIYycu/view?usp=sharing> (public access link)

Presentation:

<https://docs.google.com/presentation/d/12KYolEQGQeyihLltz3u-3hcydnGYStwLbgd5AVSDawo/edit?usp=sharing> (public access link)

Program:

<https://drive.google.com/drive/folders/1C32dQ8AfrSb7FcIL2DUFDCTNU6yL36sN?usp=sharing> (request access)



Thank you

