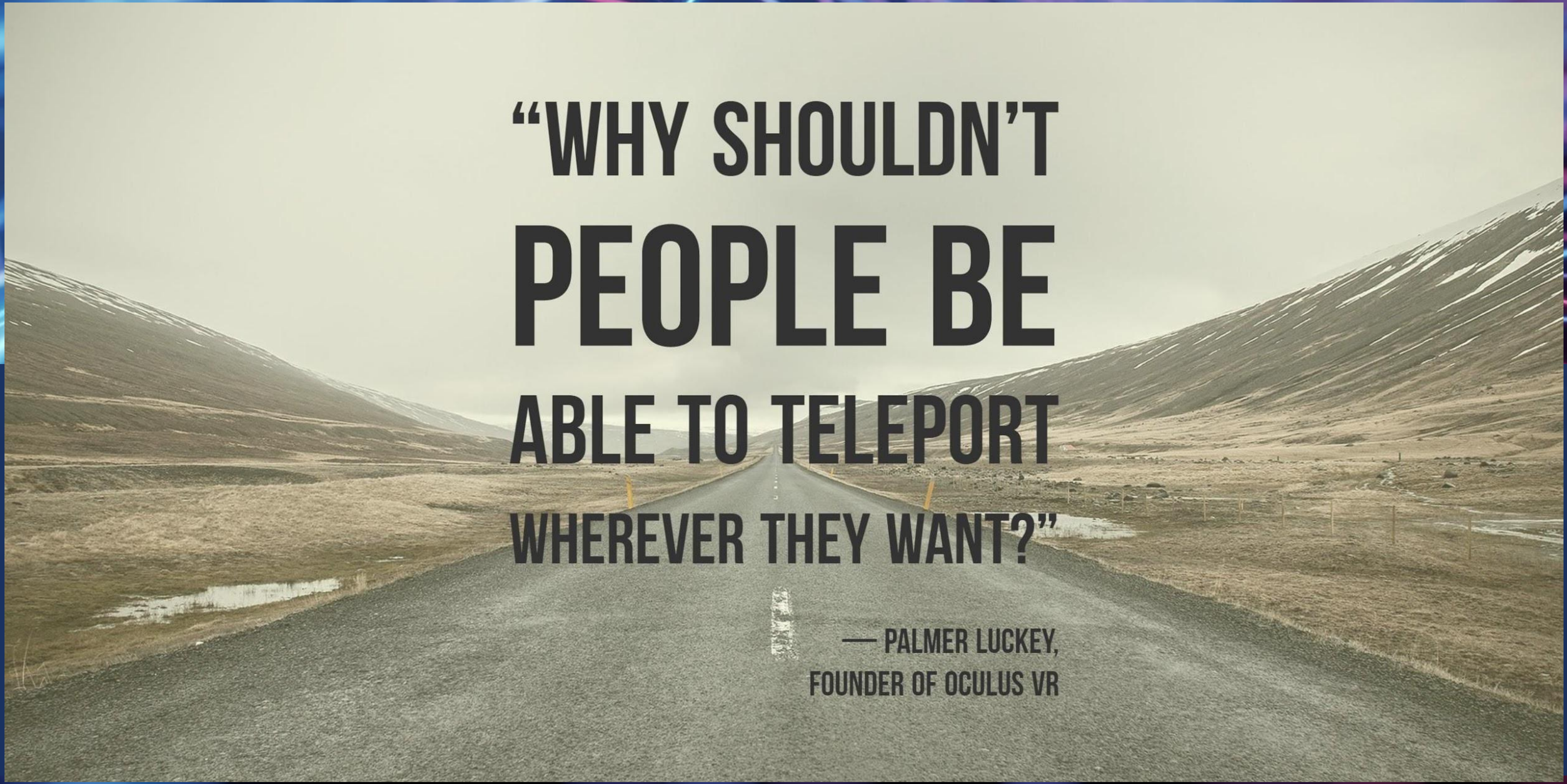
IN-HOUSE VISUALIZATION OF ORBIT USING AUGMENTED REALITY



"WHY SHOULDN'T PEOPLE BE ABLE TO TELEPORT WHEREVER THEY WANT?"

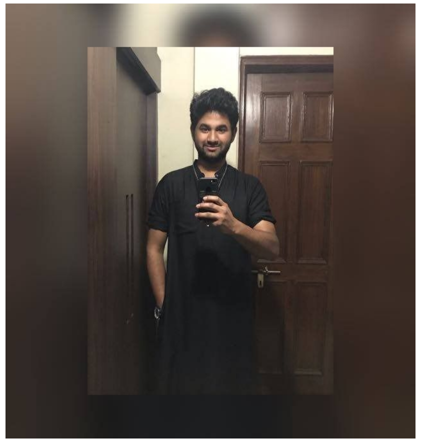
— PALMER LUCKEY, FOUNDER OF OCULUS VR

TEAM

Mentor : Sumit Maheshwari

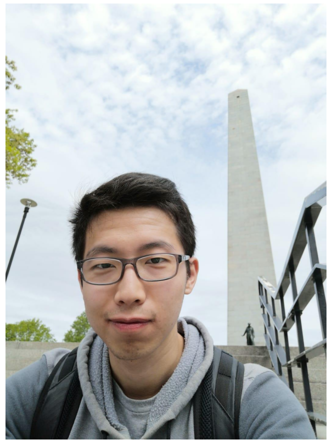
Phd researcher, WINLAB
Advisor: Prof. Dipankar Raychaudhari

Team Member 1: Ajinkya Patankar



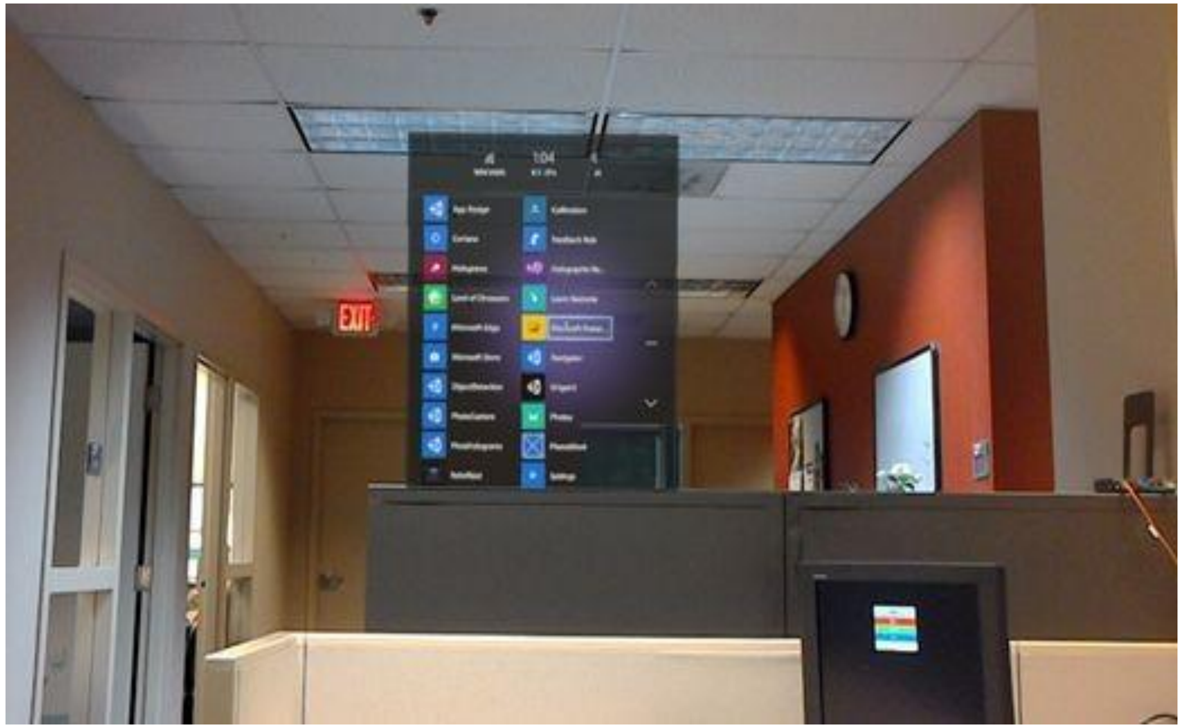
Rising Grad student in CS department at Rutgers

Team Member 2 : Weiping Jiao



Rising under-grad student in ECE department at Rutgers

IDEA



To enable a user to visualise the information about objects or a group of objects in the orbit labs using Object Detection and Augmented Reality

OVERVIEW

Aim

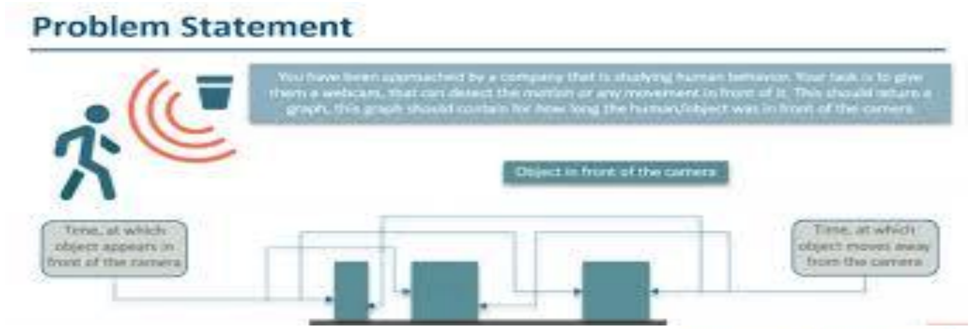
- To provide in-house navigation facility.
- To enable the user to navigate the interiors of the building without actually having to walk around.

Motivation

- We implement an in-house navigation system with the help of object detection and memory mapping to provide meaningful information to the user.

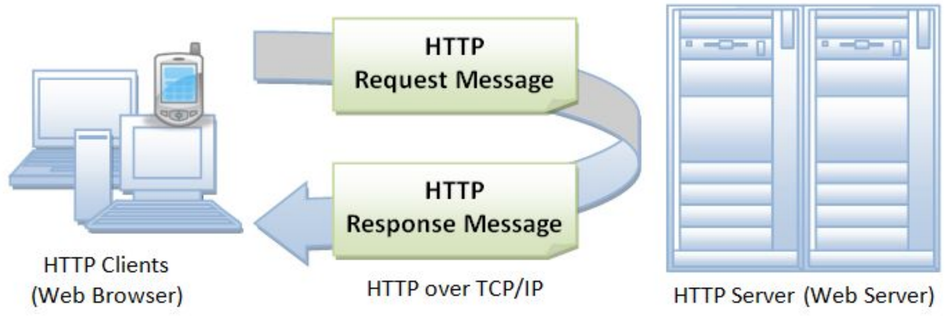
COMPONENTS AND TECHNOLOGIES

- Hololens : HoloLens provides a state of the art technique to achieveAR functionalities by its unique features such as frame ofreferences (stationary and attached), spatial anchoring andspatial mapping.
- OpenCV Library : OpenCV (*Open source computer vision*) is a library of programming functions mainly aimed at real-time computer vision. Its applications include gesture recognition, motion tracking, etc. We use only the 2-D feature toolkits for our project.
- Http Protocol : Protocol used by the World Wide Web defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.


Problem Statement

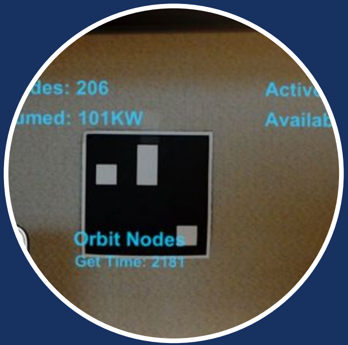
You have been approached by a company that is studying human behavior. Your task is to give them a webcam, that can detect the motion or any movement in front of it. This should return a graph, this graph should contain for how long the human/object was in front of the camera.

Object in front of the camera
Time, at which object appears in front of the camera
Time, at which object moves away from the camera



PROJECT COMPONENTS

User-Interface



Back-end

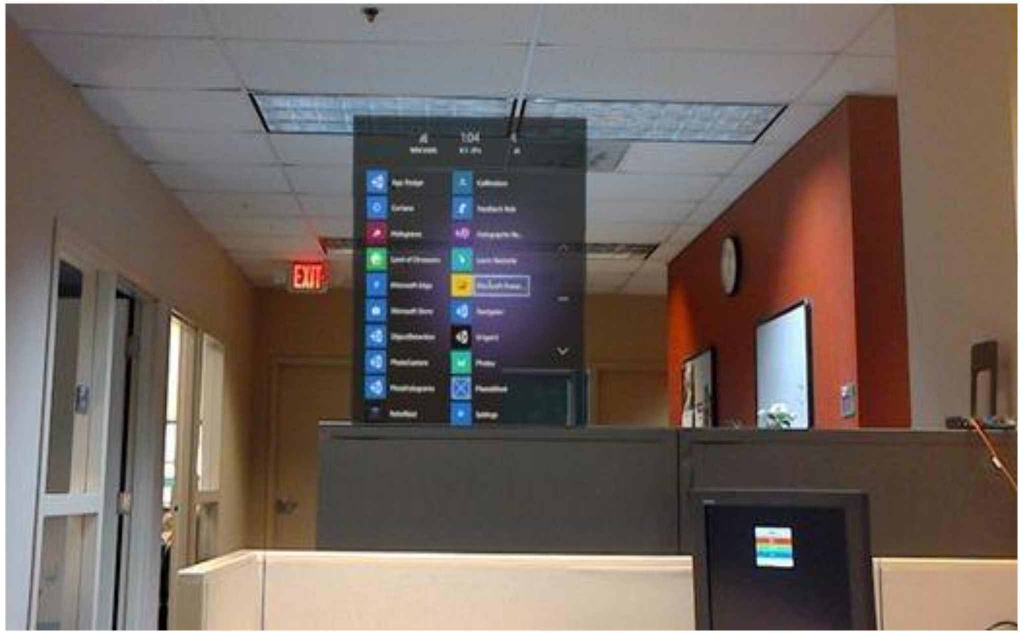
```
(edges.copy(), cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)[-2:]

ours
eight) / 50
 in contours if len(contour) > min_contour_length]

ay(((0, 0),
     (warped_size - 1, 0),
     (warped_size - 1, warped_size - 1),
     (0, warped_size - 1)),
    dtype='float32')



:
.approxPolyDP(contour, len(contour) * 0.01, True)
_curve) == 4 and cv2.isContourConvex(approx_curve)):


ay(cv2.convexHull(approx_curve, clockwise=False),
   dtype='float32')
.getPerspectiveTransform(sorted_curve, canonical_marker_coords)
arpPerspective(img, persp_transf, (warped_size, warped_size))
vtColor(warped_img, cv2.COLOR_BGR2GRAY)

threshold(warped_gray, 127, 255, cv2.THRESH_BINARY)
reshape(
ped_size / MARKER_SIZE, MARKER_SIZE, warped_size / MARKER_SIZE]

xis=3).mean(axis=1)



rn(marker)
ming_code(marker)
ing_code), 2)
d)
r(id=marker_id, contours=approx_curve))
```

Database

FRONT-END : USER-INTERFACE

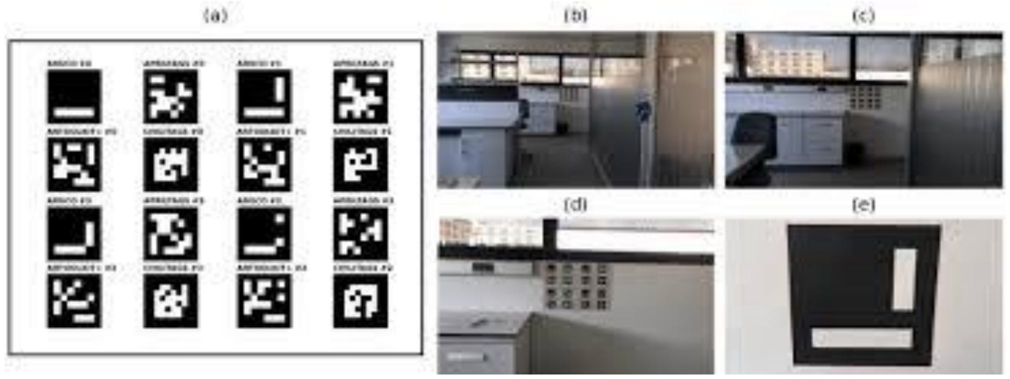
User-Interface is the primary interaction of the user with the application. It is what the user realizes his aim through.

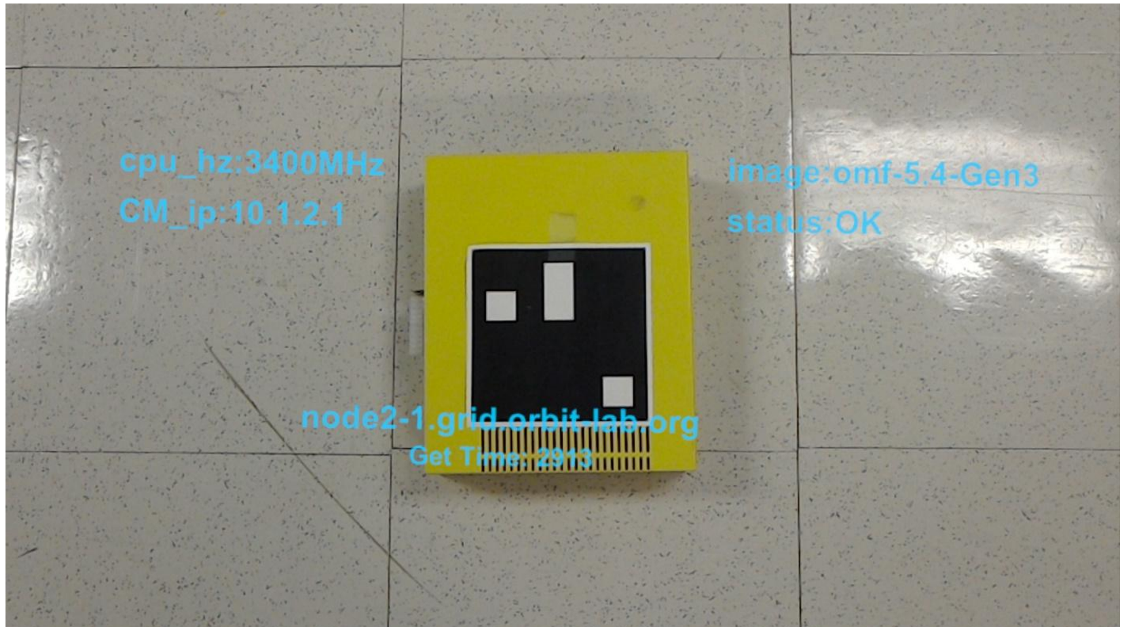
- Developed an app for the user-interface using Unity IDE, and the Visual Studio. The Unity helps us in creating structure of the application.
- Once the outlook is done, its compiled into a solution in the visual studio which provides the computer interpreted part.
- Our UI consists of a simple Unity 3-D object recognition. application.



Virtual Reality as seen in the physical world

BACK-END : OBJECT DETECTION





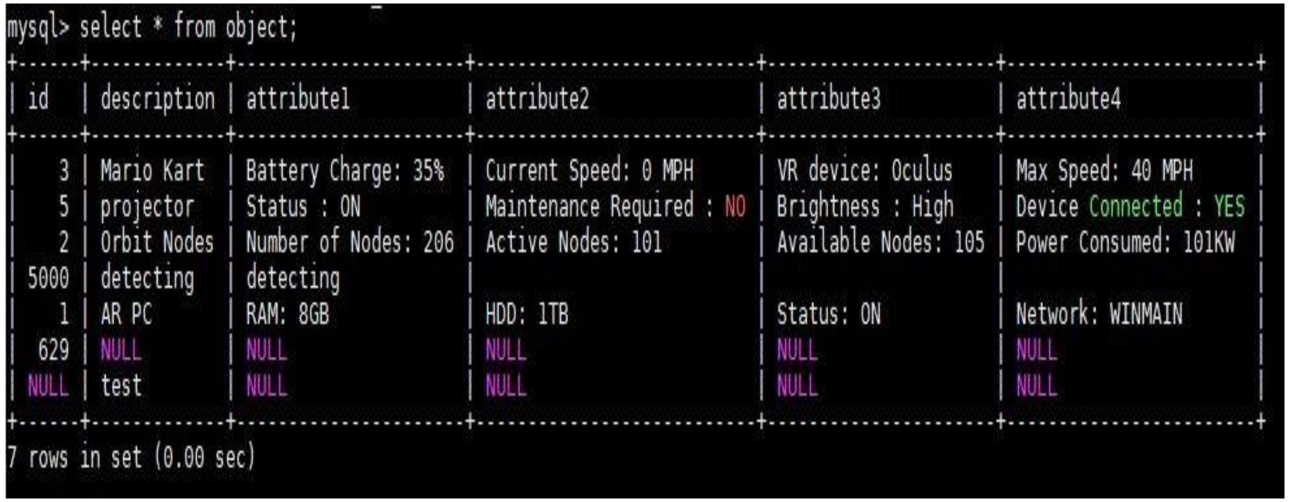
Back-end is the background operations cluster of the application, which performs requests of the user and displays them on front-end

- The object detection component forms the bulk of our project. It uses OpenCV library to detect chilitags (barcodes), each associated with an user specific information, to display relevant information on the Hololens.
- For this purpose, we generated chilitags initially by using random binary digits spaced in a 2-D matrix and then converting the 0's as blacks and 1's as whites.
- These 2-D formations are unique and are associated with a unique identifier, called UID. These UID's are used to fetch information about the objects.

DATABASE AND API REQUEST

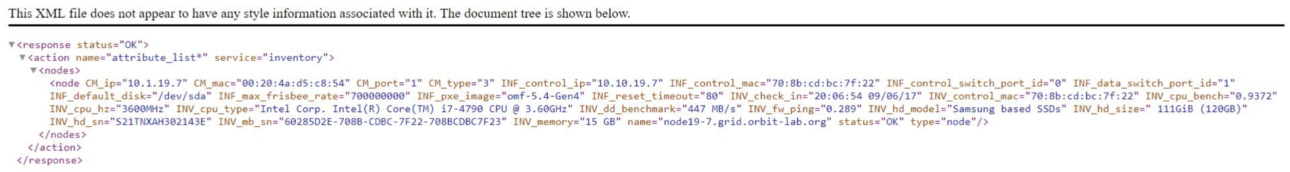
Databases are also a part of the back-end and work accordingly by sending the data requested by the front-end to the back-end for processing

- The database of the project has been developed in the MySQL.
- It consists of 6 parameters: UIDs, object name and 4 other parameters. The first two parameters are mandatory, while the other 4 are not and can be customised according to the needs of the system.
- The UIDs is the functioning link between the developed chilitags and the objects, and it is through these that object detection app fetches information and display on the screen.
- **Orbit nodes API** to fetch dynamic information of ORBIT nodes using the API request and response query.



```
mysql> select * from object;
+------+-------------+--------------------+-----------------------------+--------------------+----------------------+
| id   | description | attribute1         | attribute2                  | attribute3         | attribute4           |
+------+-------------+--------------------+-----------------------------+--------------------+----------------------+
|    3 | Mario Kart  | Battery Charge: 35% | Current Speed: 0 MPH       | VR device: Oculus  | Max Speed: 40 MPH    |
|    5 | projector   | Status : ON        | Maintenance Required : NO   | Brightness : High  | Device Connected : YES |
|    2 | Orbit Nodes | Number of Nodes: 206 | Active Nodes: 101         | Available Nodes: 105 | Power Consumed: 101KW |
| 5000 | detecting   | detecting          |                             |                    |                      |
|    1 | AR PC       | RAM: 8GB           | HDD: 1TB                    | Status: ON         | Network: WINMAIN     |
|  629 | NULL        | NULL               | NULL                        | NULL               | NULL                 |
| NULL | test        | NULL               | NULL                        | NULL               | NULL                 |
+------+-------------+--------------------+-----------------------------+--------------------+----------------------+
7 rows in set (0.00 sec)
```

Static database with a few entries

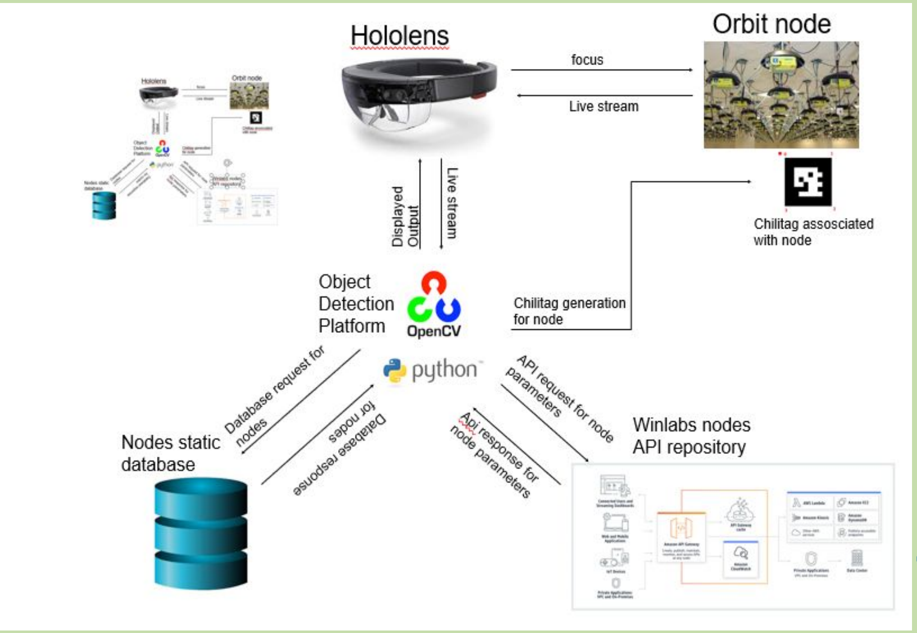


This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<response status="OK">
  ▼<action name="attribute_list*" service="inventory">
    ▼<nodes>
       <node CM_ip="10.1.19.7" CM_mac="00:20:4a:d5:c8:54" CM_port="1" CM_type="3" INF_control_ip="10.10.19.7" INF_control_mac="70:8b:cd:bc:7f:22" INF_control_switch_port_id="0" INF_data_switch_port_id="1"
       INF_default_disk="/dev/sda" INF_max_frisbee_rate="700000000" INF_pxe_image="omf-5.4-Gen4" INF_reset_timeout="80" INV_check_in="20:06:54 09/06/17" INV_control_mac="70:8b:cd:bc:7f:22" INV_cpu_bench="0.9372"
       INV_cpu_hz="3600MHz" INV_cpu_type="Intel Corp. Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz" INV_dd_benchmark="447 MB/s" INV_fw_ping="0.289" INV_hd_model="Samsung based SSDs" INV_hd_size=" 111GiB (120GB)"
       INV_hd_sn="S21TNXAH302143E" INV_mb_sn="60285D2E-708B-CDBC-7F22-708BCDBC7F23" INV_memory="15 GB" name="node19-7.grid.orbit-lab.org" status="OK" type="node"/>
     </nodes>
   </action>
 </response>
```

Otbit-Nodes API Response

PROJECT ARCHITECTURE AND DATA FLOW


Hololens
Orbit node
focus
Live stream
Displayed Output
Live stream
Chilitag assosciated with node
Object Detection Platform
OpenCV
Chilitag generation for node
python
Database request for nodes
Database response for nodes
API request for node parameters
Api response for node parameters
Winlabs nodes API repository
Nodes static database

Data Flow Diagram

1. **Application:** Once the application starts, the hololens focuses on the nodes and detaches the live-stream. This is achieved by establishing a socket connection between the hololens as the client and PC as the server.
2. **OpenCV platform:** That is the library which is used initially to generate chilitags associated with each orbit-node and also decodes the chilitags collected by live-stream to generate a unique identifier.
3. **Object Detection:** The live-stream is 30 frames/second and decodes the chilitag to get the unique identifier. The app then sends an API request to the Orbit-nodes API repository. It also sends database requests to the database to fetch details related to the orbit node.
4. **Database requests and reponses:** The decoded unique identifier requests the node number from the database and then appends it to the API request command. It also fetches other parameters as requested by the database query and sends them back to the Hololens.
5. **ORBIT node API repository:** A python code requests the parameters of the nodes by using the API request command

CONCLUSIONS

- The output displayed on the Hololens screen consists of the focussed node and 5 parameters.
- The first 2 parameters are name and IP address, received from the API repository.
- The remaining 3 parameters are the information about the nodes received from the database.
- The API information is dynamic in nature, i.e. it fetches the fresh information of the node while the database information is static in nature i.e. it fetches the information stored in the database by the administrator.

FUTURE WORK

- Visualizing other aspects of the ORBIT nodes like wavelength, frequency, memory usage in 3-D models, by using D3.js.
- Incorporate Artificial Intelligence model to enable the Hololens guide the user from outside the building.

Any Questions, Suggestions or Recommendations?

THANK YOU