Title

Tourist Eyes

Organization

vEyes

Experiment Description

Blind people encounter huge difficulties in visiting unknown places, even if they have traditional assistive supports like a white stick and a guide dog, or some Tactile Ground Surface Indicators are installed in the visited places. Moreover, a guide dog may often be not available since it may be not immediately assigned to a blind person abroad, or because the person can be allergic to animals. A possible solution that has been adopted in the last few years is to run a GPS-based application on the smartphone of the blind person; however, this does not give a sufficiently high-precision to guarantee safety for the blind person when he/she moves in the city and might not work outdoor. On top of that, orientation and point-of-interest identification prove to be very difficult operations with the currently available technologies.

The objective of this experiment, named *Tourist Eyes*, is to provide blind tourists, visiting a smart city, with a framework for supporting their activities. *Tourist Eyes* is proposed by vEyes [1], a non-profit organization whose mission is to realize assistive technologies for blind people. The main idea is porting the testbed Poseidon 2.0 [2, 3], developed by vEyes and already tested for blind swimmers, over a 5G infrastructure, in order to apply Poseidon 2.0 to more complex scenarios like a smart city.

The following pages list and explain in detail the software components needed to carry out the experiment. Afterwards, the preparation and implementation steps needed to set-up the experiment are shown, distinguishing those carried out remotely (e.g the upload of the descriptors in the 5GinFIRE [4] Portal) from those performed at Millennium Square (Bristol).

The application scenario leverages on the facilities provided by the University of Bristol 5G Testbed.

1 Software components

In this section we provide a brief description of the Tourist Eyes components that have been either used or appropriately designed and implemented for the experiment.

1.1 .NET Framework

.NET Framework is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It is required for both the Tourist Eyes Manager and Tourist Eyes Console to function.

1.2 SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft. The database is used by the Tourist Eyes system to manage places and related points of interest, cameras, routes, and all the other parameters needed to track the user.

1.3 Web Services

Web services have been designed to support interoperability between the app in the smartphone and the database. Specifically, they are used by the Tourist Eyes app to retrieve from the SQL Server the list of available places, the relevant points of interest and to change the color of the hat/umbrella used by the user in order to be traced.

1.4 Tourist Eyes Manager

The Tourist Eyes Manager (briefly Manager) is the core of the entire system. After the system boot, the Manager listens for connections on a certain port. Through the Tourist Eyes App the user sends a request to the Manager to be tracked and guided to safely arrive at the desired destination. After selecting the starting point and the ending point from the Tourist Eyes app, a connection request is sent to the Manager. The Manager then executes a query in the database to retrieve the appropriate route (for each starting point and ending point there is only one route). Afterwards, the first two cameras of the route begin to send video streams to the Manager, as shown in Figure 1: the tracking algorithm is performed on the first camera, while the second one is ready to welcome the blind user.

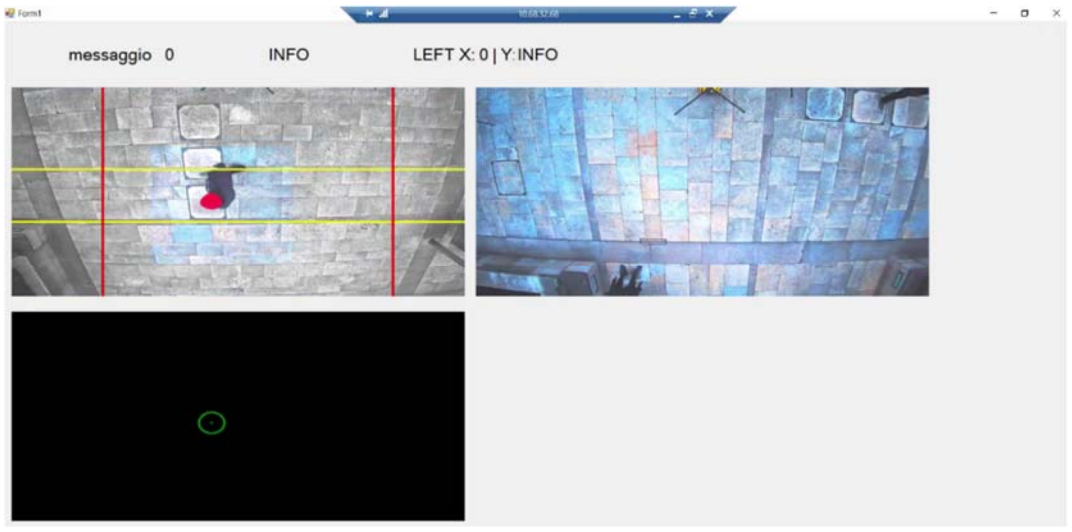


Figure 1 Tourist Eyes Manager

Figure 2 Console Camera Icon

This way, while the user leaves the first cam, the second one is already sending its video stream to the Manager, which then "shifts" its tracking algorithm from the video stream of the first camera to the stream of the second one while guaranteeing a soft handover procedure.

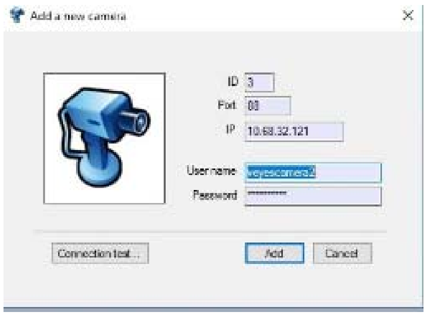
When the blind user leaves the first camera and enters the second one, the first camera stops sending video streams and the third camera simultaneously starts its transmission; in the meantime, the user is being tracked by the second cam. The tracking algorithm searches for a pseudo-spherical object of colour previously chosen by the user through the app; moreover, to reduce interference from external objects of the same colour in the surrounding environment, a greyscale filter is applied to each frame of the video frames, grayscaling everything but a squared area near the user. Another filter can be applied to enhance certain colours, so that the System will be able to recognize the hat more easily. Tracking is carried out by specific guidelines that are set using the Console during the set-up of the experiment. When the user crosses one of these guidelines, the Manager sends a message to the app. This message is decoded by the app and transduced into an acoustic signal. The user, after receiving the acoustic beep, will change direction to re-enter the route. Once at the destination, another acoustic beep is sent to the user, and the Manager starts listening for new connections. The Manager also has a graphical interface that has been used for debugging purposes during the execution of the experiment.

1.5 Tourist Eyes Console

The Virtual Eyes console (briefly Console) is implemented as a C# application. The Console aims at simplifying the set-up process of the experiment by providing an intuitive graphical user interface that enables the creation of new routes and the configuration of the route parameters. Therefore, the console drastically reduces the set-up time of the experiment while enabling the management of the system set-up without directly interacting with the database. Below, we describe the operations that must be performed to create and set the route parameters.

Insertion of a new camera

The first operation to be carried out is the addition of a new camera to the list of the system cameras.



Add a new camera

ID: 3
Port: 80
IP: 10.68.32.121
User name: veyescamera2
Password: **********

Figure 3 Add a new Camera

A camera can be used by the system only after it has been added to the list of available cameras. This is done by clicking on the camera icon (Figure 2), and then, entering the camera ID, port, IP address, username and password (Figure 3). By clicking on connection test, a window is shown where, if the connection is successful, the camera streaming is shown.

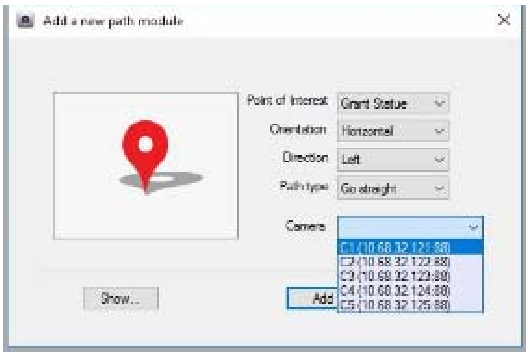
Route creation and parameters setup

The creation of a new route is done by clicking the icon next to the Camera icon at the top right corner of the window, already shown in Figure 2. A route is made up of several modules; each module is associated with one cam. First of all, select the starting Point of Interest from which the user will be tracked. Then you need to configure the orientation, the direction and the message type of the Starting Module. The orientation can be both horizontal or vertical, the direction depends on the orientation and can be both right or left for horizontal modules and up or down for vertical modules. Finally, the camera associated to the module must be selected, as shown in Figure 4. The message type represents the audio signal that will be sent to the user when he/she crosses a specific guideline. There are two types of guidelines:

- Yellow Guidelines
- Red Guidelines

The yellow lines delimit the virtual track in which the blind user must walk: when the user crosses one of these two lines, a message is sent from the Manager to prompt the user (e.g make a step on the side) to re-enter the route. The red lines define the prompted message that the Manager will send to the user when he/she is approaching the next camera. In this way, the user is prompted to make a right /left turn or just to keep going straight. The next step is to set the position of the red and yellow lines. This is done by right-clicking on the module that have just been created and by selecting "show". This way, the console shows the real time video streamed by the camera, thus facilitating the setting of guidelines position. Finally, you need to select the radius of the hat, which needs to be reconfigured for each camera, since each camera can be set at different heights w.r.t. the ground. It is also possible to set the radius for the umbrella, since it can be used as an option to the hat. This last step concludes the configuration of the starting module. As already said before, a route is made up of several modules; two of them must be the starting and the arrival module. A new module can be added to an existing route by clicking on one of the already-created modules and by selecting "add new module".

The operations to be performed are those just described, with the only addition of having to specify whether the module just added is the last of the route (the arrival) or not. It is important to note that the outward and return journeys are created in two different routes since there are touristic sites like museums where the routes are unidirectional.



Point of Interest: Grant Statue
Orientation: Horizontal
Direction: Left
Path type: Go straight
Camera:
C1 (10.68.32.121:88)
C2 (10.68.32.122:88)
C3 (10.68.32.123:88)
C4 (10.68.32.124:88)
C5 (10.68.32.125:88)

Figure 4 Add a new module

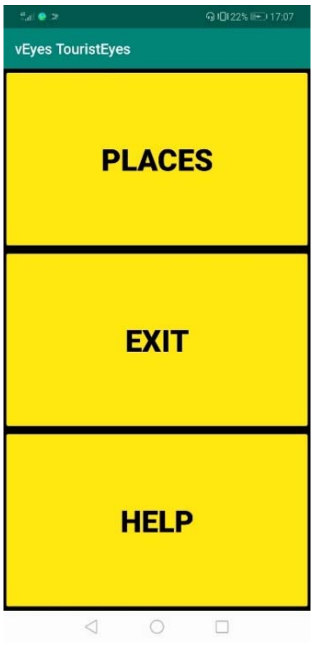
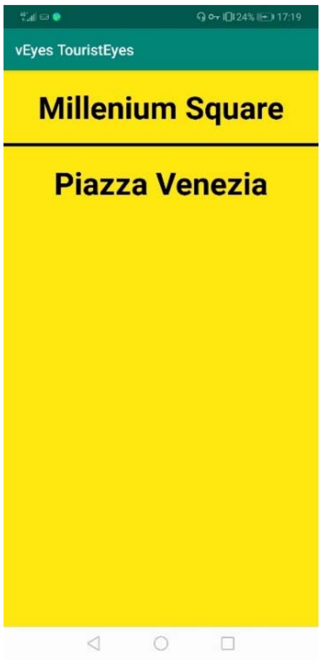


Figure 5 Tourist Eyes App Main Screen

1.6 Tourist Eyes App

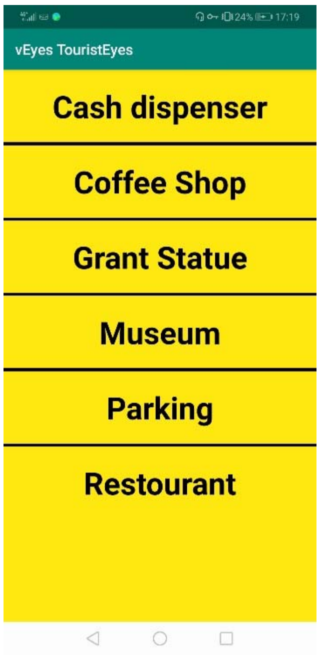
The Tourist Eyes app is the only software component that each user of the system must have installed on his/her smartphone. Through the app, the user sends a request to the Manager to be tracked and guided to safely arrive at the desired destination. The blind user can easily interact with the application thanks to the support of the smartphone's voice assistant and the Tourist Eyes Training App. From the main screen of the app (shown in Figure 5), select "Places" to display the list of places where the Tourist Eyes system is active, as shown in Figure 6.


vEyes TouristEyes

Millenium Square

Piazza Venezia

Figure 6 Tourist Eyes App List of Places


vEyes TouristEyes

Cash dispenser

Coffee Shop

Grant Statue

Museum

Parking

Restourant

Figure 7 Tourist Eyes App Points of Interest

By selecting, for example, Millennium Square, a list of points of interest is displayed (Figure 7); after selecting the starting point, a second list shows the possible points of arrival. Note that the retrieval of all this information is done via Web Services, which are used by the Tourist Eyes app to retrieve the lists of available places and relevant points of interest from the SQL Server. After selecting the point of departure and arrival, a request is sent to the Manager. Afterwards, the user will only have to click GO! (Figure 8). The Manager will now begin to send messages to the app; these messages will be transduced by the app into acoustic beeps that the user will receive through his/her headphones. When the user arrives at the destination, tracking ends and new connections can be accepted by the Manager.

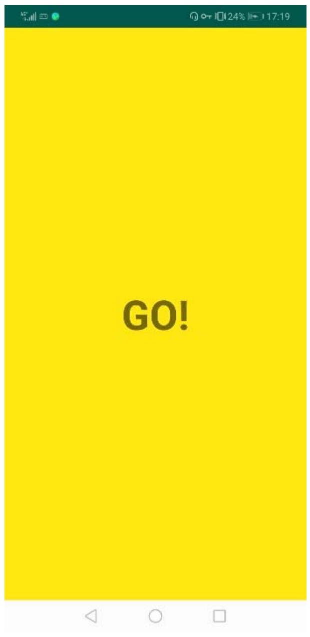


Figure 8 Tourist Eyes App "GO" Screen

1.7 Tourist Eyes Training App

The Tourist Eyes Training App, whose snapshot is shown in Figure 9, is used to "train" the user to recognize the actions to be performed for each acoustic signal. The "Sound SX" and "Sound DX" signals tell the user to take a sidestep respectively to the right and to the left. The "Turn to SX" and "Turn to DX" signals tell the user to make respectively a 90° leftwards or rightwards rotation. When the user arrives at destination, he/she will be notified with the "Destination Hit" sound. The "Lost tracking" sound indicates instead that the user is currently not being monitored by any camera. This last signal has been particularly useful to successfully set up both cameras and guidelines.

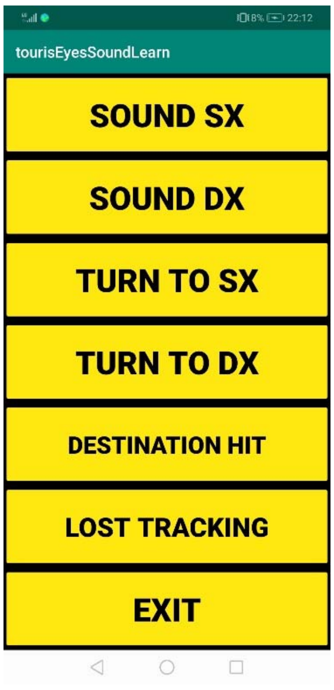
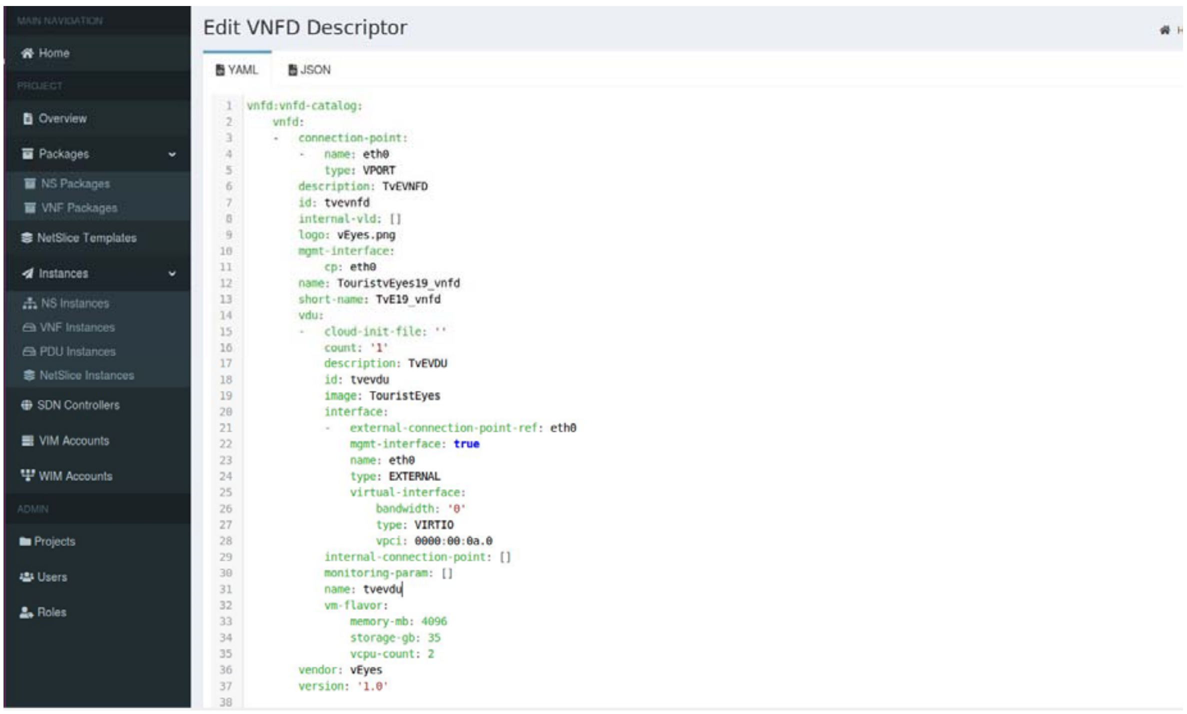


Figure 9 Tourist Eyes Training App Main Screen



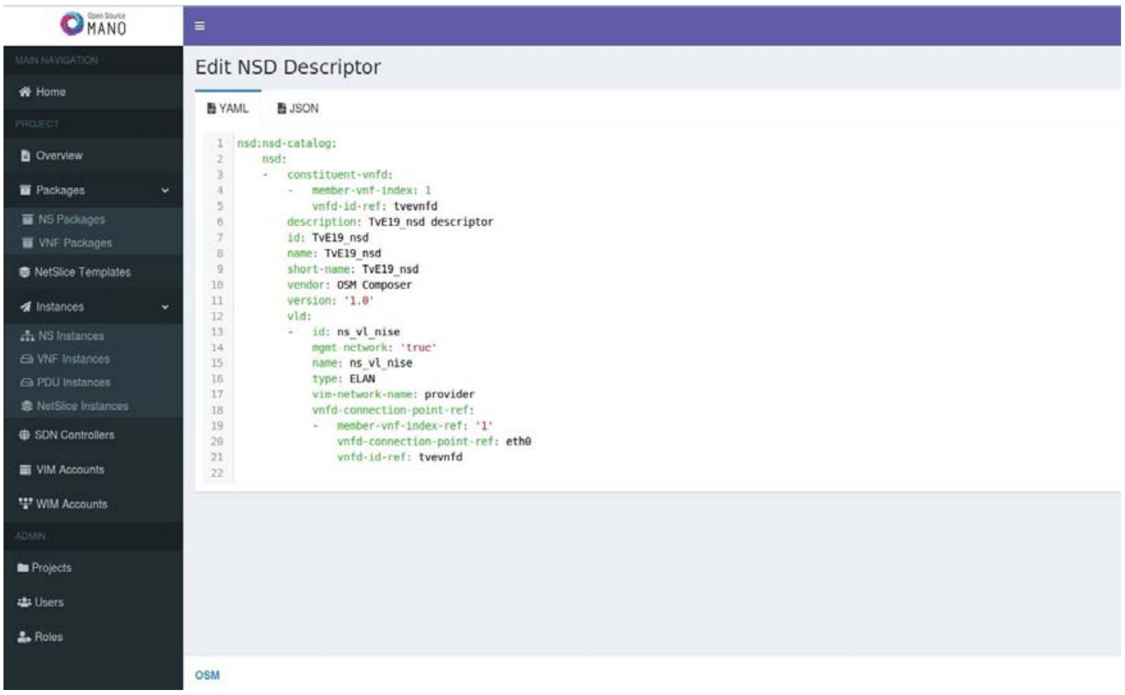
```yaml
vnfd:vnfd-catalog:
    vnfd:
    -   connection-point:
        -   name: eth0
            type: VPORT
        description: TvEVNFD
        id: tvevnfd
        internal-vld: []
        logo: vEyes.png
        mgmt-interface:
            cp: eth0
        name: TouristvEyes19_vnfd
        short-name: TvE19_vnfd
        vdu:
        -   cloud-init-file: ''
            count: '1'
            description: TvEVDU
            id: tvevdu
            image: TouristEyes
            interface:
            -   external-connection-point-ref: eth0
                mgmt-interface: true
                name: eth0
                type: EXTERNAL
                virtual-interface:
                    bandwidth: '0'
                    type: VIRTIO
                    vpci: 0000:00:0a.0
            internal-connection-point: []
            monitoring-param: []
            name: tvevdu
            vm-flavor:
                memory-mb: 4096
                storage-gb: 35
                vcpu-count: 2
        vendor: vEyes
        version: '1.0'
```

Figure 10 VNFD

2 Experiment Preparation and Implementation

2.1 Set-up

During the first part of the set-up, the image containing all the software components required to execute the experiment was uploaded in the 5GinFIRE portal. The VNFD, shown in Figure 9, and NSD, shown in Figure 10, were also uploaded.



```yaml
nsd:nsd-catalog:
    nsd:
    -   constituent-vnfd:
        -   member-vnf-index: 1
            vnfd-id-ref: tvevnfd
        description: TvE19_nsd descriptor
        id: TvE19_nsd
        name: TvE19_nsd
        short-name: TvE19_nsd
        vendor: OSM Composer
        version: '1.0'
        vld:
        -   id: ns_vl_nise
            mgmt-network: 'true'
            name: ns_vl_nise
            type: ELAN
            vim-network-name: provider
            vnfd-connection-point-ref:
            -   member-vnf-index-ref: '1'
                vnfd-connection-point-ref: eth0
                vnfd-id-ref: tvevnfd
```

Figure 11 NSD

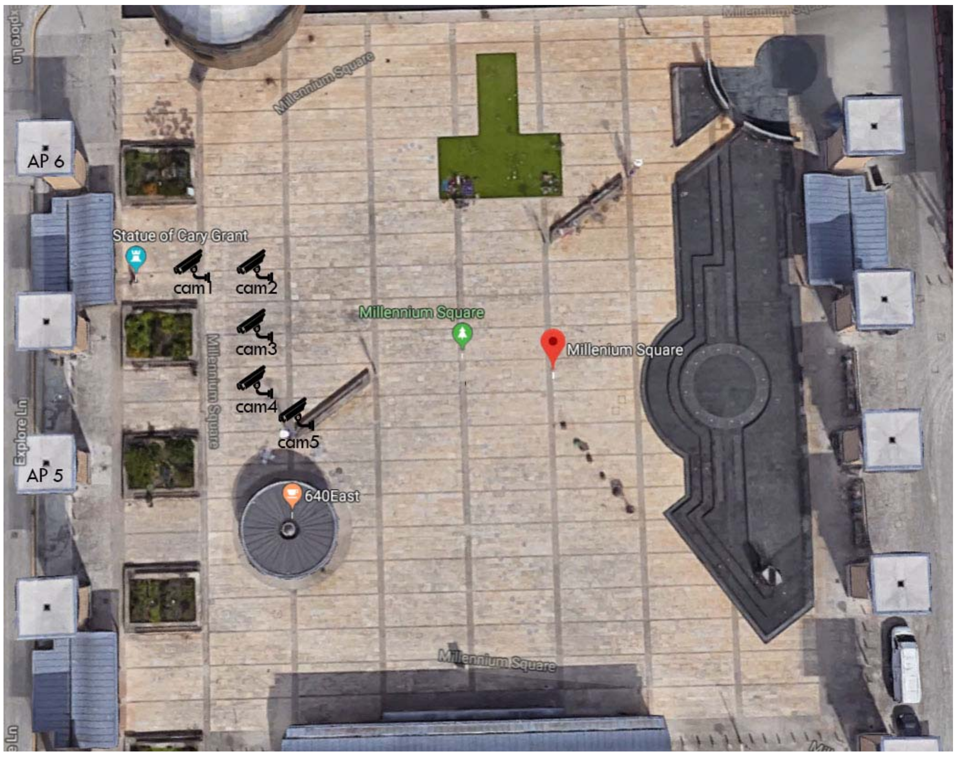


Figure 12 Camera and Access Point position at Millennium Square

The second phase of the setup of the experiment was carried out at Millennium Square, where IP cameras have been installed. A preliminary check involved the test of the connectivity between the IP cameras and the VNF. Every IP camera that has been used for the experiment could only support the 2.4 GHz band option. Table 1 shows the distance of each IP camera from the access point. There were 2 access points at Millennium Square to get connected to the Tourist Eyes network. The first AP was fixed on top of “Tower 5” of Millennium Square, while the second one was deployed on top of “Tower 6”. Please refer to Figure 12 to have a better understanding of the position of the cameras and the access points at Millennium Square. The installation and configuration of the cameras was carried out through the Console following the steps described in detail later. Finally, the Console allowed us to create the routes from Grant Statue and 640East and configure the 5 modules of which it is composed. The creation of the forward route (from the Grant Statue to 640East) and the return route (from the 640East to the Grant Statue) concluded the experiment set-up phase. All the data have been recorded and processed in accordance with the DSA and will be legally compliant and in line with GDPR principles and requirements.

Table 1 Distance between IP Cams and Access Points

IP Cam	Access Point	Distance
1	6	12.3 m
2	6	16.5 m
3	6	19 m
4	5	14.4 m
5	5	14.7 m

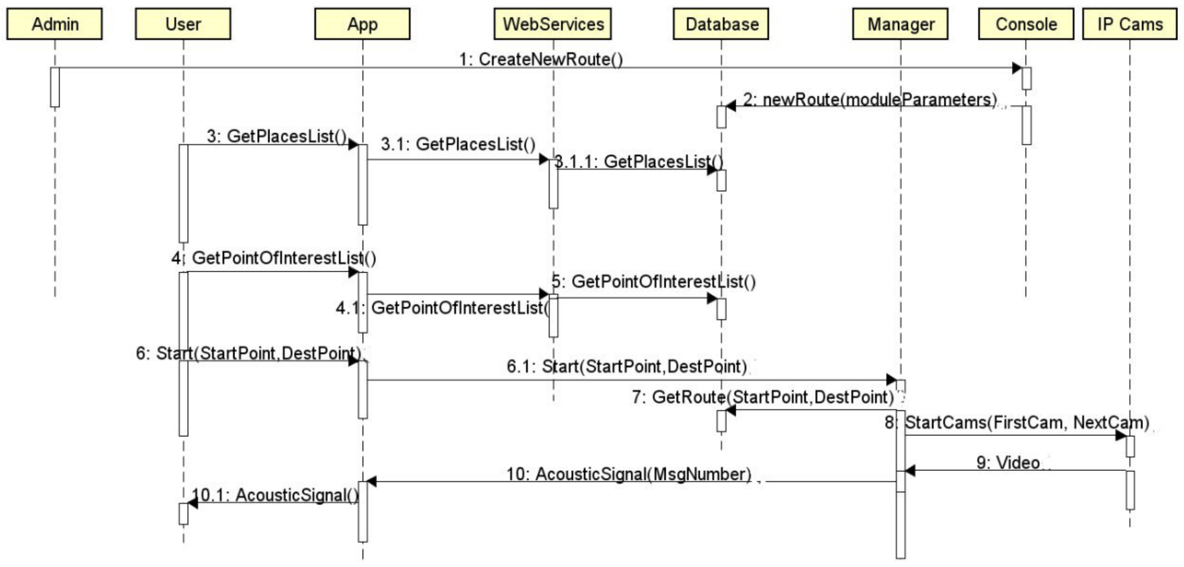
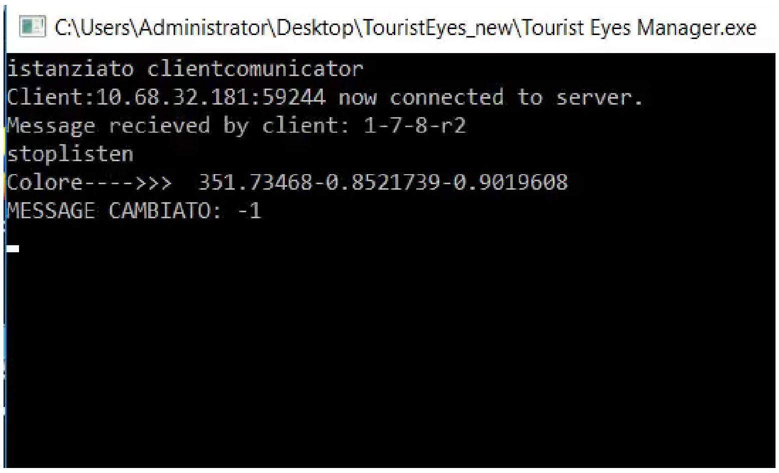

Admin
User
App
WebServices
Database
Manager
Console
IP Cams
1: CreateNewRoute()
2: newRoute(moduleParameters)
3: GetPlacesList()
3.1: GetPlacesList()
3.1.1: GetPlacesList()
4: GetPointOfInterestList()
5: GetPointOfInterestList()
4.1: GetPointOfInterestList()
6: Start(StartPoint,DestPoint)
6.1: Start(StartPoint,DestPoint)
7: GetRoute(StartPoint,DestPoint)
8: StartCams(FirstCam, NextCam)
9: Video
10: AcousticSignal(MsgNumber)
10.1: AcousticSignal()

Figure 13 Data Flows

2.2 Data flows

This section describes the data flows between the main components of the Tourist Eyes system. The SQL database contains the points of interest and the routes that allow the user to move from one point of interest to another. This database is managed by the admin of the system, which has the ability to create a new path. The creation of a new path consists in the configuration of all the parameters required for the correct functioning of the system. This operation is carried out via the Tourist Eyes Console. The user, assisted by the smartphone voice assistant, selects the list of available places and points of interest using the Tourist Eyes App. All these data needs to be retrieved from the database; to do so, the app uses the webservices to collect from the database the list of places and the relevant points of interest. After selecting the point of interest of departure and arrival, the app sends a connection request to the Tourist Eyes Manager, including the point of departure and arrival. The Manager will query the database to retrieve the route that connects the starting point to the destination point, the modules of the route, and the list of cameras associated with the modules. Afterwards, the Manager will send a video request to the first and second camera of the route; the tracking algorithm will be applied initially to the first camera, and when the user "leaves" the first camera, the algorithm will be applied to the second camera and at the same time the video will be requested from the third camera of the route (and so on until the user arrives at destination). During the tracking algorithm, the Manager will send data to the app. The data transmitted is then going to be encoded by the app and translated into an acoustic signal, which will be received by the user through his/her headphones; the user will then perform a specific movement: if no acoustic signals are received, the user keeps walking straight.



```
C:\Users\Administrator\Desktop\TouristEyes_new\Tourist Eyes Manager.exe
istanziato clientcomunicator
Client:10.68.32.181:59244 now connected to server.
Message recieved by client: 1-7-8-r2
stoplisten
Colore---->>>  351.73468-0.8521739-0.9019608
MESSAGE CAMBIATO: -1
```

Figure 14 Tourist Eyes Manager receives a request from the app

3 Experiment Execution

The execution of the experiment consists of two phases.

PHASE 0: Service Activation by a blind tourist

The blind tourist, assisted by the smartphone voice assistant, activates the *Tourist Eye* app on his/her smartphone and selects the point of departure and arrival from the list of available places and points of interest (Figure 7). In our case, the user selected Grant Statue as the departure and Coffee Shop as the arrival. Then, the app sends a connection request to the Tourist Eyes Manager (Figure 14), including the point of departure and arrival. Afterwards, the Manager will query the database to retrieve the route that connects the starting point to the destination point (highlighted in red in Figure 15), the modules of the route (Figure 16), and the list of cameras associated with the modules. Finally, the Manager sends a video request to the first and second camera of the route. This concludes the system service activation.

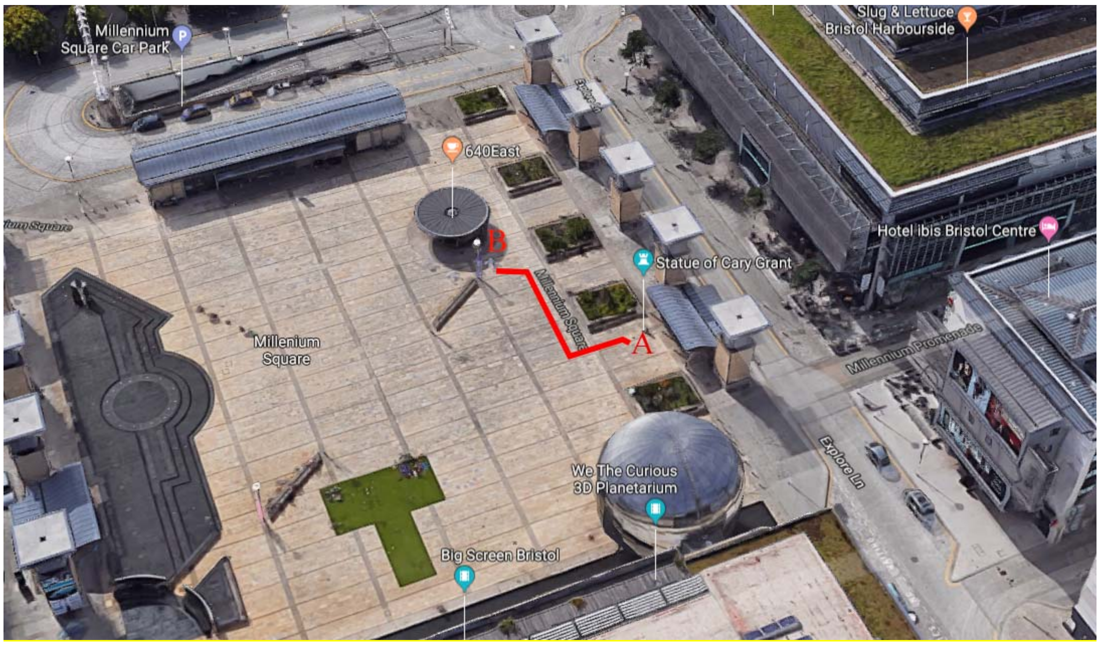

Millennium Square Car Park
Slug & Lettuce Bristol Harbourside
640East
B
Statue of Cary Grant
Hotel ibis Bristol Centre
Millennium Square
Millenium Square
A
Millennium Promenade
Explore Ln
We The Curious 3D Planetarium
Big Screen Bristol

Figure 15 Experiment Execution

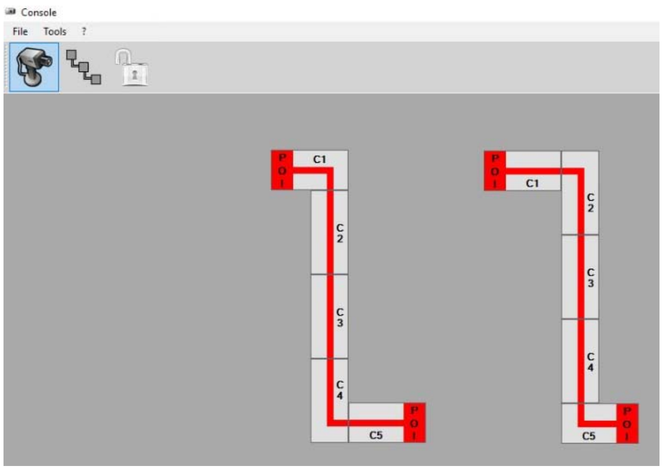


Figure 16 Routes and modules

<u>PHASE 1: Walk from the Grant Statue (A) to the 640East (B)</u>

The blind tourist walks from the Grant Statue through the path highlighted in Figure 15 to reach the 640 East Coffee Shop. When the user starts walking, the Manager begins to track the user through the first camera of the route (shown in Figure 17). When the user makes a 90° rightwards rotation, the tracking algorithm is shifted from first camera to the second one. The user then continues to walk straight for about ten meters (the tracking is shifted from the second camera to the third one and then from the third one to the fourth one – each camera is approximately 4m away from the others); afterwards the user makes a 90° leftwards rotation (the tracking algorithm is shifted to the fifth camera. After completing this rotation, the user will only need to take a few steps to get to the coffee shop. The reverse path, from the 640East (B) to the Grant Statue (A), has been crossed as well. In both paths, the user has successfully managed to reach their destination. During the experiment, the Manager sends data to the app. The data transmitted is encoded by the app and translated into an acoustic signal, which is received by the user through his/her headphones; the user will then perform a specific movement: if no acoustic signals are received, the user keeps walking straight. The user is able to perform the appropriate movement only if he/she has been appropriately trained, using the *Tourist Eyes Training* app, to recognize the acoustic signals.

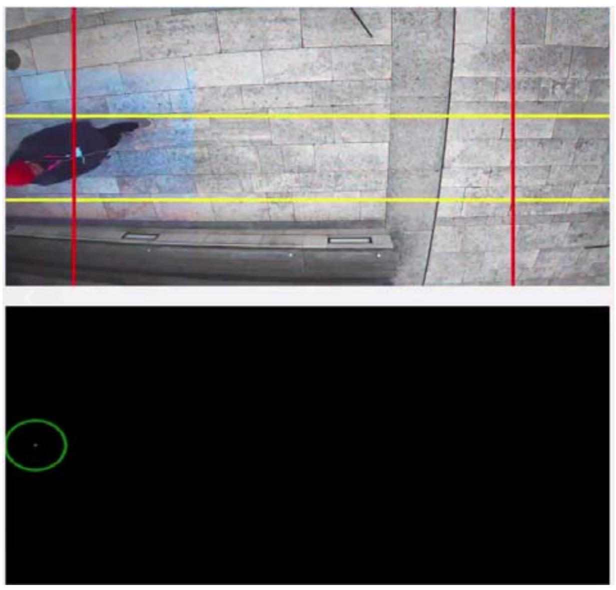


Figure 17 Tracking Algorithm after Service Activation

4 Measurements

During the experiment, we identified three different working regions, as reported in the following:

1) Time Interval P1: Tuesday, October 22nd, from 10:45 to 11.55 am
2) Time Interval P2: Wednesday, October 23rd, from 12:45 to 1.45 pm
3) Time Interval P3: Wednesday, October 23rd, from 4:45 to 5.45 pm

Specifically:

- in P1 the system experienced high quality performance, since the video was flowing smoothly through the system.
- in P2 the system performed poorly in terms of quality of service. The system worked very poorly, with recurring image freezing events and consequent system non-responsiveness, as detected by the user. Even though the blind user was diverting from his path, the system gave no feedback to correct the trajectory, or gave feedback with a very long (unacceptable) delay.
- in P3 the system worked with rather good quality. All the experiments during this time interval were successful, the user was able to receive all the expected feedback and complete his/her path. In fact, when his/her diversions from the established path resulted in deviations from the planned trajectory, the feedback worked properly, although with short but still acceptable delay.

Note that the distinction of these working regions comes from problems which were external to the Tourist Eyes experiment and mainly related to traffic and users accessing the network, considering that the Millennium Square area in Bristol was very crowded at certain day time and many users at the time of the experiment execution were using their own applications and accessing the Internet. Also the weather conditions were very variable and unstable, and thus our experiment suffered of the bad propagation conditions.

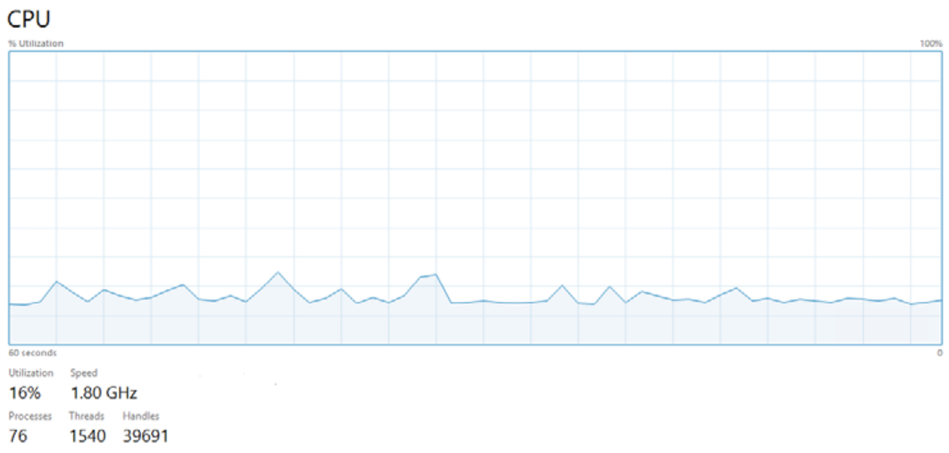
In the following sections, we analyse, for each of the three scenarios, the following parameters:

- VM status, in terms of CPU and memory usage

- Overall bitrate of IPCAM video flows entering the VM
- Ping Delay
- Ping message loss percentage

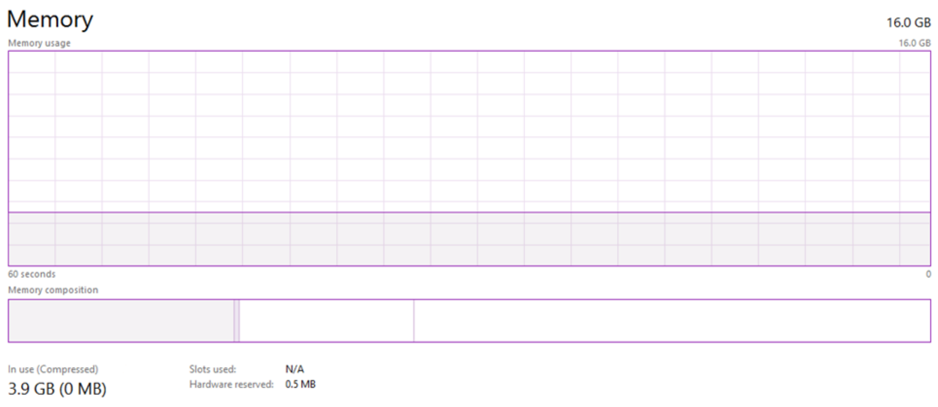
In order to represent what happened at the application level, we also add some figures reporting the console screenshots for the different areas where the user moves as served by the cameras.

Before showing the status of the VM for each scenario, let us show the CPU usage (Figure 18), and the memory usage (Figure 19) when the VM is not loaded.



Utilization 16%
Speed 1.80 GHz
Processes 76
Threads 1540
Handles 39691

Figure 18. CPU - VM not loaded



In use (Compressed) 3.9 GB (0 MB)
Slots used: N/A
Hardware reserved: 0.5 MB

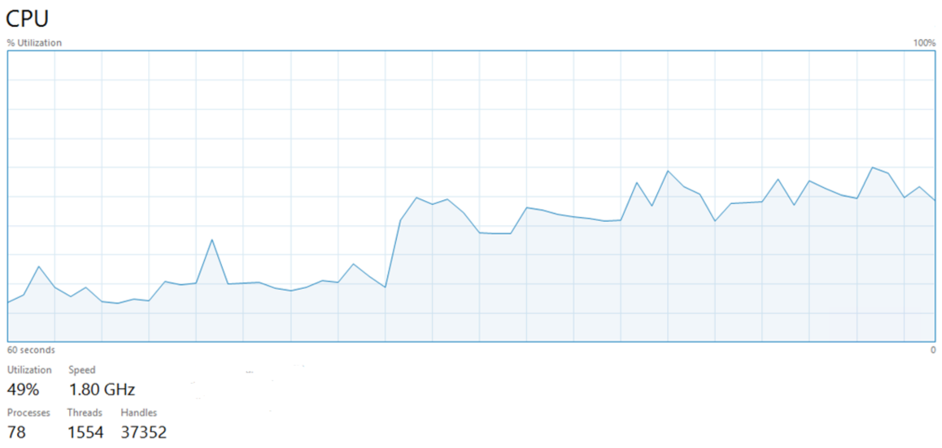
Figure 19. Memory Usage - VM not loaded

For each of the 3 scenarios, also screenshots taken through the Tourist Eyes Console are shown. In particular, for each scenario, the following three images are presented:

- In the first image, the user is shown while moving along the path when visible to CAM3.
- In the second image, the passage of the user from CAM3 to CAM4 is shown. In these images the tracking algorithm is applied only to the video stream sent from CAM3 (while CAM4 is already transmitting videos so that the "shift" of the tracking algorithm from CAM3 to CAM4 is carried out in the shortest possible amount of time).
- In the last image of each scenario we show how the tracking algorithm is finally applied to CAM4 (this does not apply to the "very poor quality" scenario, since the system freezes before the algorithm is shifted).

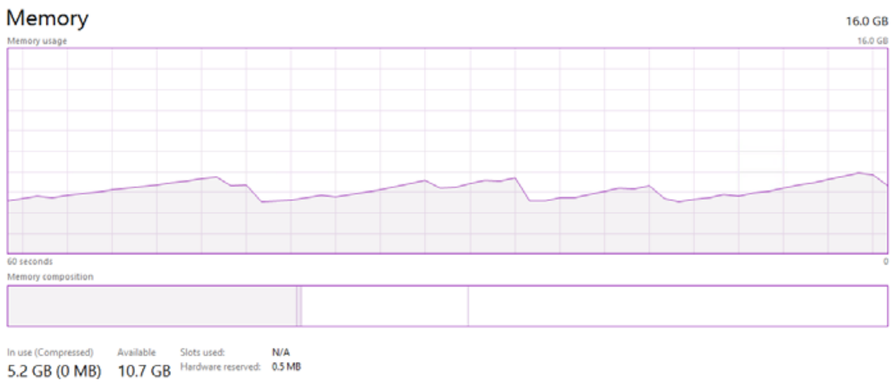
1.1 System Evaluation in a high quality scenario (scenario P1)

First, we show the CPU usage (Figure 20) and the memory usage (Figure 21) of the VM.



Utilization 49%
Speed 1.80 GHz
Processes 78
Threads 1554
Handles 37352

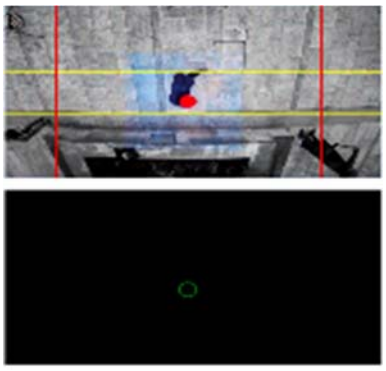
Figure 20. CPU – High Quality Scenario

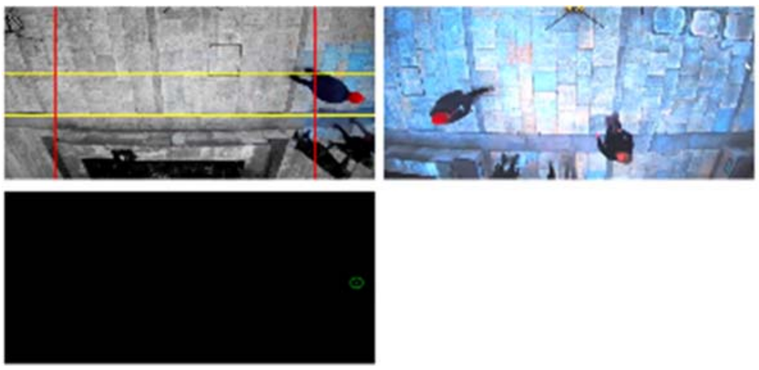


In use (Compressed) 5.2 GB (0 MB)
Available 10.7 GB
Slots used: N/A
Hardware reserved: 0.5 MB

Figure 21**. Memory – High Quality Scenario**

Let us have a look at the position of the user in Figure 22: at the beginning, see Figure 22(a), the user is halfway through the path section visible to CAM3. The user keeps on walking; the user leaves (see Figure 22(b)) the section visible through CAM3 and enters the section of the path under coverage of CAM4; finally, the user is captured by CAM4 (see Figure 22(c)). By observing the circle in the black rectangle representing the user as tracked by the system, we note that the system perfectly works and allows promptly making handover between different cameras.


(a) CAM3


(b) User leaving CAM3

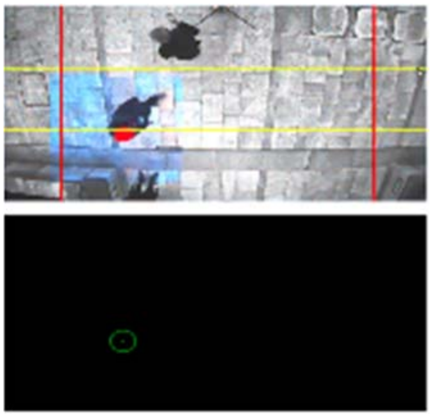
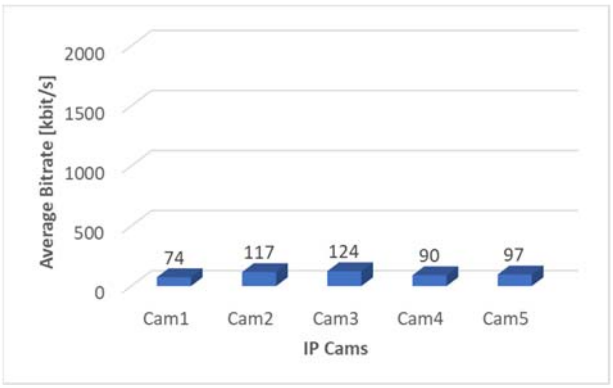
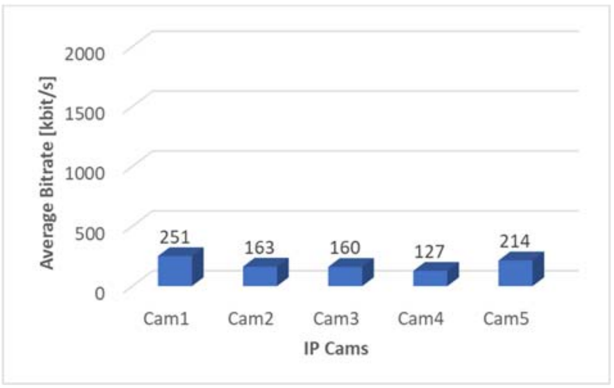

(c) User entering CAM4

Figure 22. Tourist Eyes Console Screenshots – High Quality Scenario

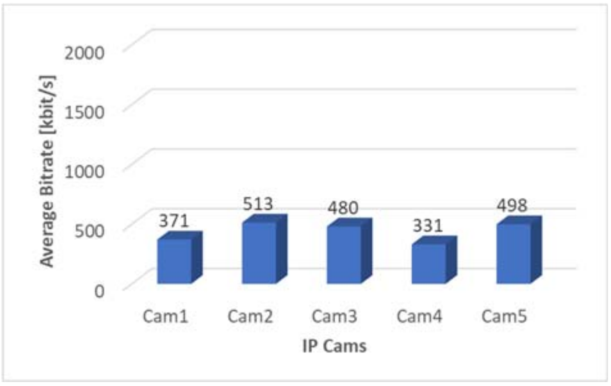
For each of the considered scenarios we also estimated the average bitrate achieved by the 5 cameras upon varying the encoding bit rate. As shown in Figure 23, upon increasing the encoding bit rate, as expected, a higher bit rate is measured. In particular, when the encoding rate is not more than 512 kbit/s, the bit rate is very close to the encoding target. Instead, when the target is too high, i.e. 2 Mbit/s, the maximum bitrate is generated, which is not higher than 87% of the target encoding rate.



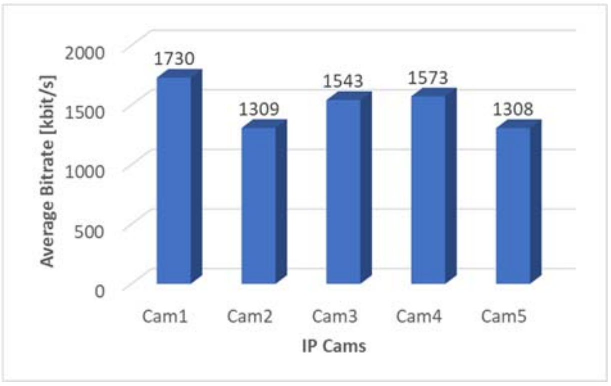
(a) Encoding bitrate: 128 kbit/s



(b) Encoding bitrate: 256 kbit/s



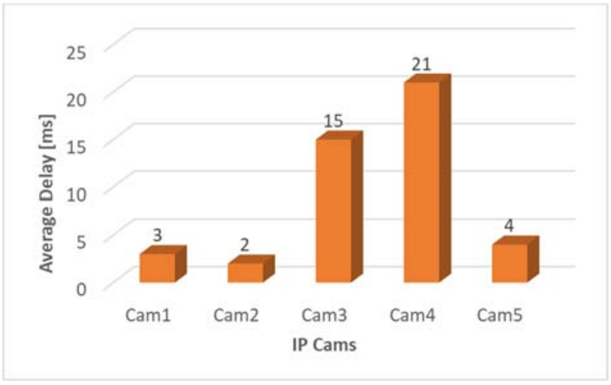
(c) Encoding bitrate: 512 kbit/s



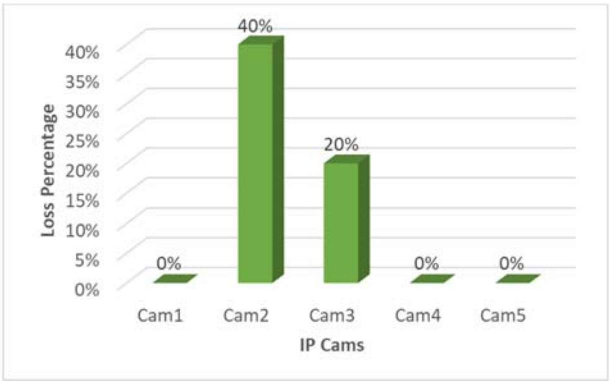
(d) Encoding bitrate: 2 Mbit/s

Figure 23. Average transmission bitrate from the IP Cams

Moreover, two highly significant parameters like the average delay and the loss percentage are shown in Figure 24. In particular, note that in Figure 24(a) the average delay of CAM3 and CAM4 are significantly higher than the average delay of all the other cams. This specific behavior of CAM3 and CAM4 was also detected in the loss percentage graph (Figure 24(b)). Thanks to the help of the Millennium Square Network Staff, we discovered that this behavior was due to a poor quality of the wireless link between these cameras and the WiFi access points.



(a) Average Delay



(b) Loss Percentage

Figure 24. Delay and Loss Percentage in P1

1.2 System Evaluation in a poor quality scenario (scenario P2)

In this section we similarly show the system performance exhibited in case of a poor quality scenario. First of all, in Figures 25 and 26 we report the CPU usage and the memory usage of the VM. We can note that, as expected, the load of the VM is not considerably influenced by the quality of the radio access link. In Figure 25, in the middle of the time axis, we have a rapid increase; this is due to the activation of the service. Nevertheless, also in this case, CPU does not result overloaded.

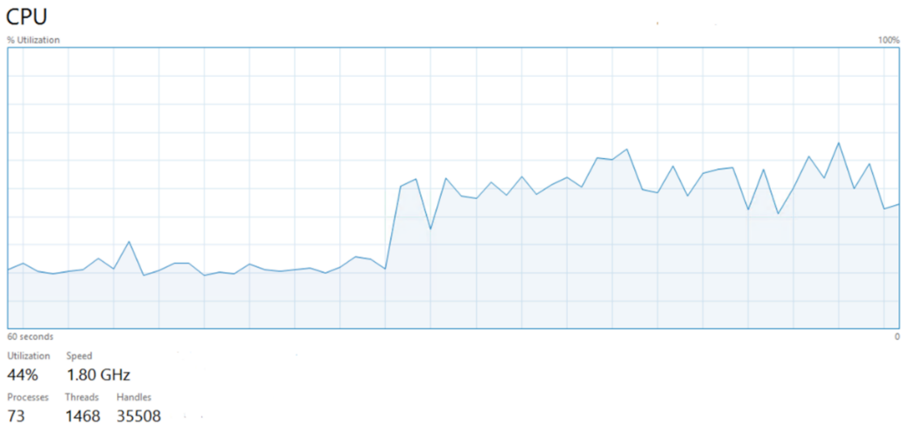


Figure 25. CPU –Poor Quality Scenario

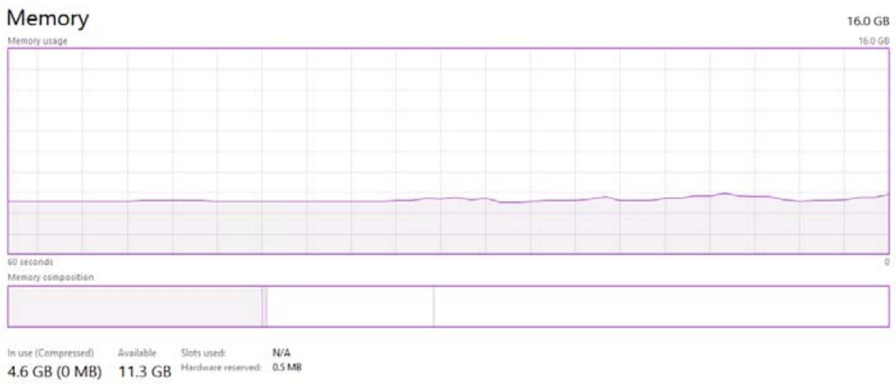
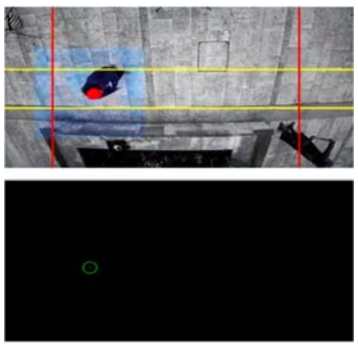
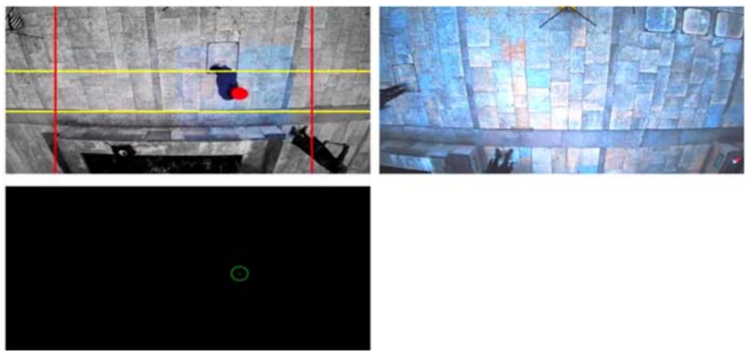


Figure 26. Memory – Poor Quality Scenario

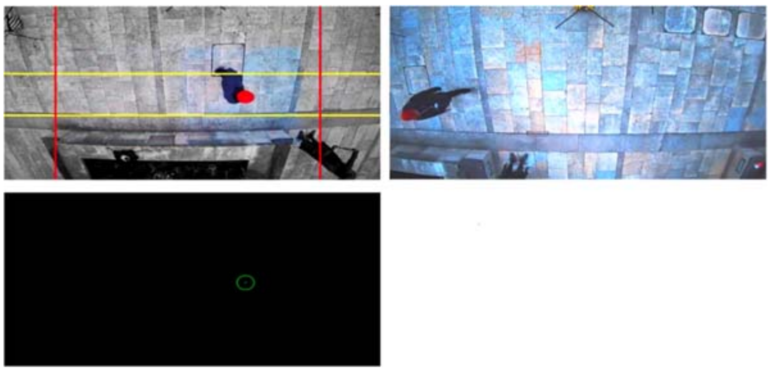
Concerning the monitoring of user's movement in Figure 27, we note that: at the beginning (Figure 27(a)), the user is halfway through the path section visible to CAM3. The user keeps on walking but suddenly the video freezes (b). Therefore, the system perceives the user as if he/she has stopped. At the same time the video from CAM4 flows regularly (see Figure 27(c)). Since the algorithm is applied to CAM3 (freezed), the user receives a false feedback, because, according to the algorithm, he stopped, while he is still moving as shown in CAM4 (Figure 27(d)). The reason for the algorithm still relaying to CAM3 is that the user has not entirely left CAM3 but CAM3 is freezed; therefore, the system has not realized that the user left CAM3 and entered the region served by CAM4.



(a) CAM3



(b) CAM3 video freezes – CAM4 video flows regularly


(c) CAM3 video still frozen – User keeps walking

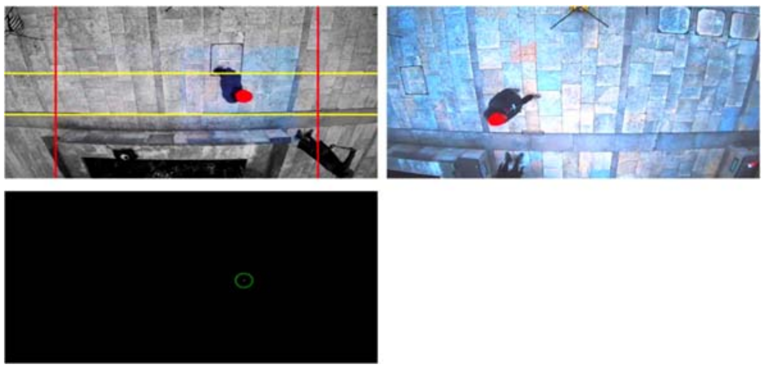
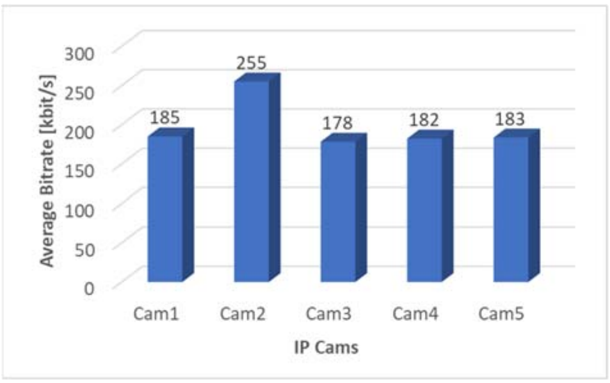
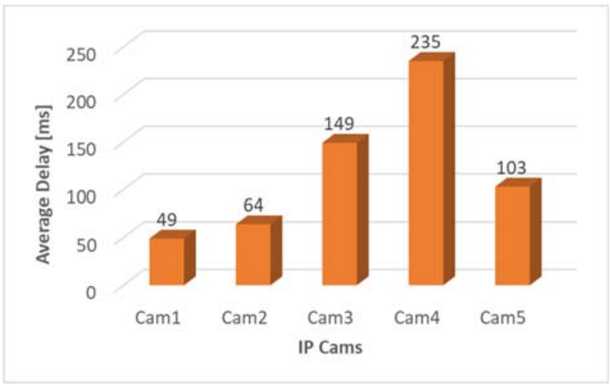

(d) CAM3 video still frozen – User still walking

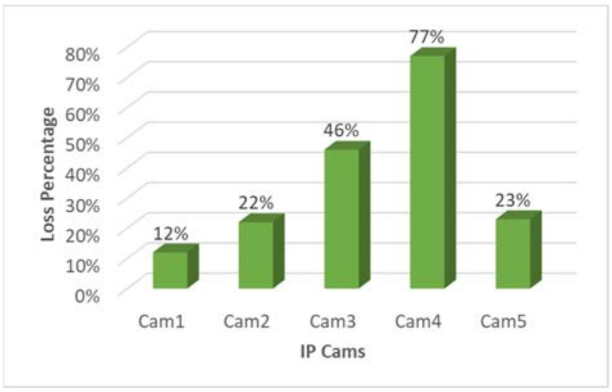
Figure 27. Tourist Eyes Console Screenshots – Poor Quality Scenario

The average transmission bitrate, the average delay and the loss percentage graphs are reported in Figure 28. Also in this case we can observe that each IP camera presents a different average transmission rate, that depends on the current scene it is focusing, the movements of the user in the covered area, and the presence of other persons in the same area. As regards average delay and loss percentage, we can observe a similar behavior of the previous scenario. In fact, although with higher values, CAM3 and CAM4 present the worst values, again due to the badness of the wireless link used for connection to the structured network. The too high value of both delay and loss percentage, especially for CAM3 and CAM4, motivate why the experiment is not working in this case.



(a) Average transmission bitrate





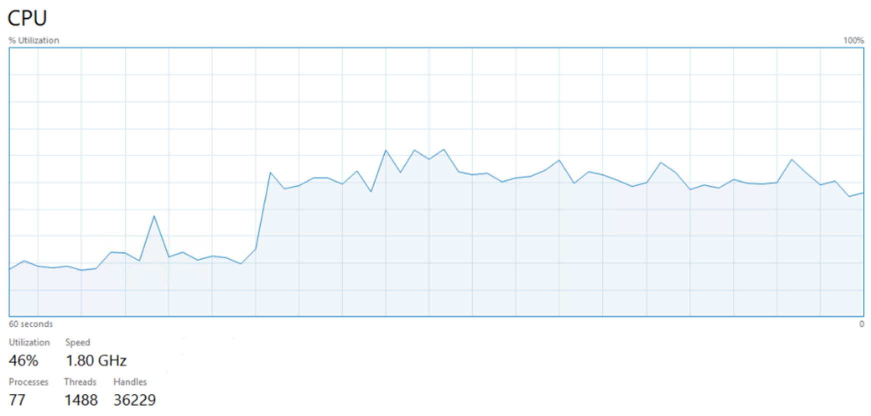
(b) Average Delay **(c) Loss Percentage**

Figure 28. Measurements - Poor Quality Scenario

1.3 System Evaluation in a good quality scenario (scenario P3)

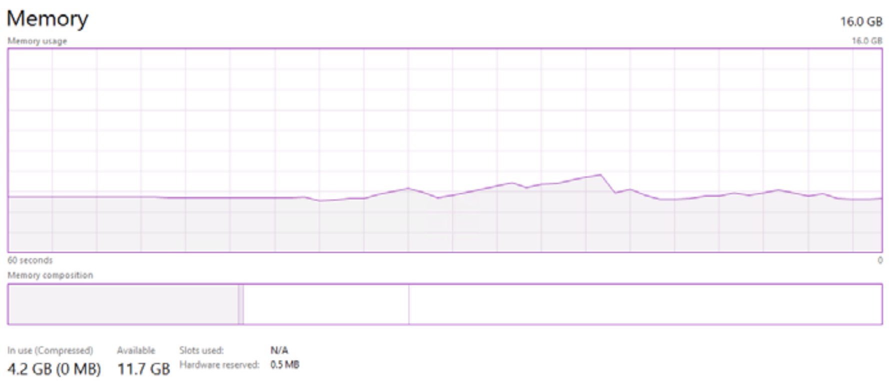
In this section we report performance results obtained in a medium quality scenario.

In Figures 29 and 30, similarly to what discussed in the previous scenarios, we report the CPU usage and the memory usage of the VM. These figures confirm that behavior of both CPU and RAM performance is not influenced by the particular scenario we are considering.



Utilization 46%
Speed 1.80 GHz
Processes 77
Threads 1488
Handles 36229

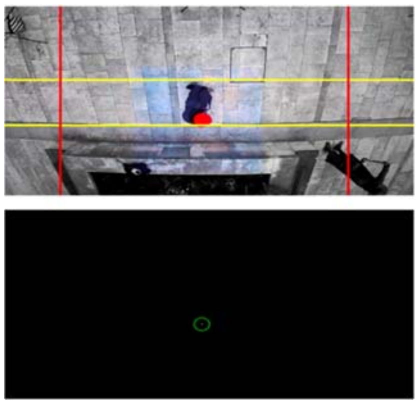
Figure 29. CPU – Good Quality Scenario

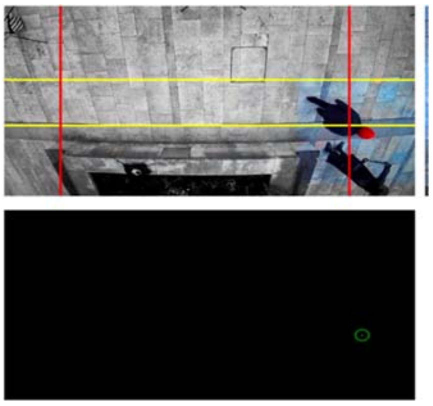


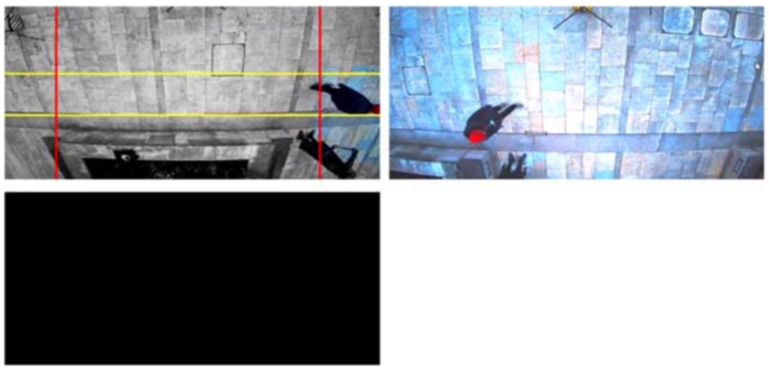
In use (Compressed) 4.2 GB (0 MB)
Available 11.7 GB
Slots used: N/A
Hardware reserved: 0.5 MB

Figure 30. Memory – Good Quality Scenario

Figure 31 shows a user moving from the area served by CAM3 to the one served by CAM4 in a good quality scenario. Let's have a look at the position of the user in Figure 31: at the beginning (Figure 31(a)), the user is halfway through the path section visible to CAM3. The user keeps on walking; the user leaves (Figure 31(b)) the section visible by CAM3 and enters the section of the path under coverage of CAM4. The network delay affects the commutation process from CAM3 to CAM4; as evident in Figure 31(d) unfortunately some delay is experienced from when the user entered the area served by CAM4 and when his new position was identified.


(a) CAM3


(b) User leaving CAM3


(c) Algorithm still applied to CAM3

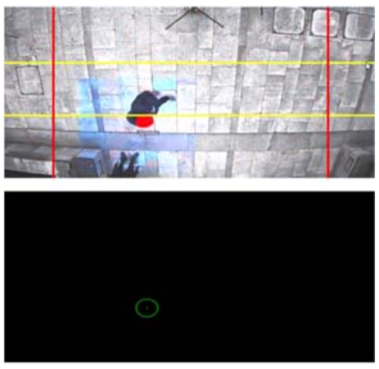
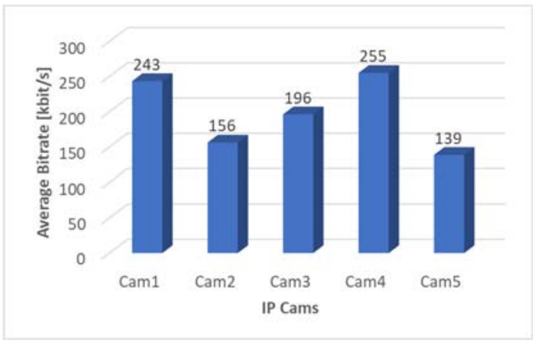

(d) User entering CAM4

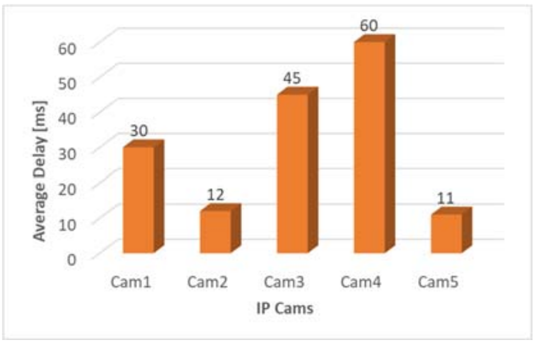
Figure 31. Tourist Eyes Console Screenshots – Good Quality Scenario

The average bitrate generated by each CAM is presented in Fig. 32(a). Here, again, we can observe that each camera has a different behavior, and Cam4 is the one more stressing the network.

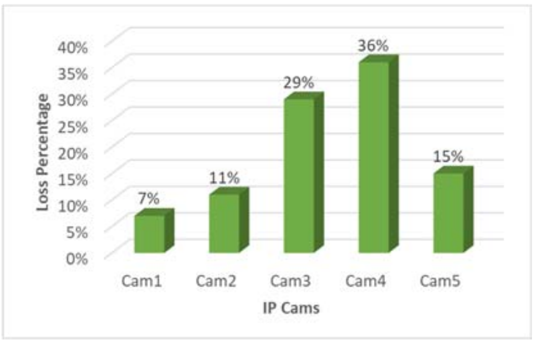
The average delay and the loss percentage are shown in Figures 32(b) and 32(c), respectively. In Figure 32(b) the average delays of CAM3 and CAM4 are significantly higher than the average delay of CAMs 1, 2 and 5. This specific behavior of CAM3 and CAM4 compared to the behavior of CAM1, CAM2 and CAM5 was also detected in the loss percentage graph shown in Figure 32(c).



(a) Average transmission bitrate



(b) Average Delay



(c) Loss Percentage

Figure 32. Measurements - Good Quality Scenario

The results carried out during the experimentation show that there are a number of critical and unexpected issues which need to be taken into account and that are related to traffic conditions, delay, weather conditions, possible temporal mis-functioning of certain cameras which strongly impact on the performance of the system. However, the Tourist Eyes system is very robust and reliable and showed to well absorb these unexpected features. The 5GINFIRE platform also revealed to be very reliable, flexible and modular in such a way to be able to well support our application.

5 Conclusions

In this document, we delivered the results including a comprehensive report on the 5GinFIRE Tourist Eyes experiment preparation, implementation and execution under three scenarios – high quality, poor quality and good quality - in collaboration with the Bristol testbed team. We developed a testing campaign on filed in October 2019 in Bristol. The experimental activity was very successful showing the effectiveness of the proposed approach and the possibility to support reliably and flexibly our application in the 5GINFIRE platform.

References

[1] http://www.veyes.it

[2] http://www.veyes.it/poseidon-2-0/

[3] https://www.youtube.com/watch?v=bwSC-A7oQaY

[4] http://5GINFIRE-5g.eu/ © 5GINFIRE consortium 2017