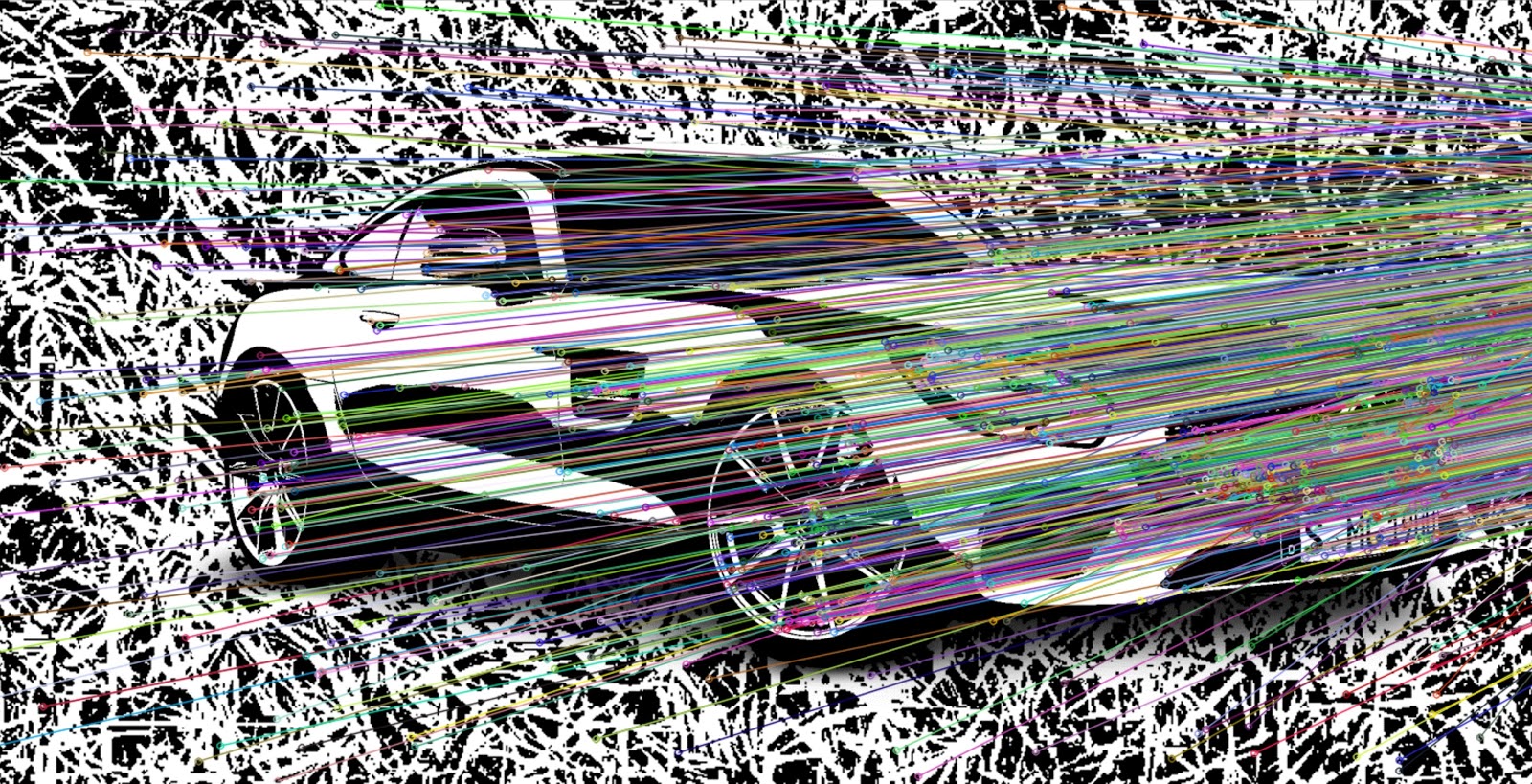
**Final Project Book**

**The Features of The Future**



Meitar Cohen

Omri Sigal

Boris Binaev

Ziv Sofrin

Guy Ohayon

Supervisor:

Dr. Shai Horowitz

1/08/2019

**Table of Contents**

**Project introduction** 3

**Our solution** 6

**Proof of concept** 6

**Academic Justifications** 7

**System description- by phases, code walkthrough up next** 8

**Our system, code walkthrough** 10

**Sequence diagram** 11

**The algorithms we used** 12

**Examples** 13

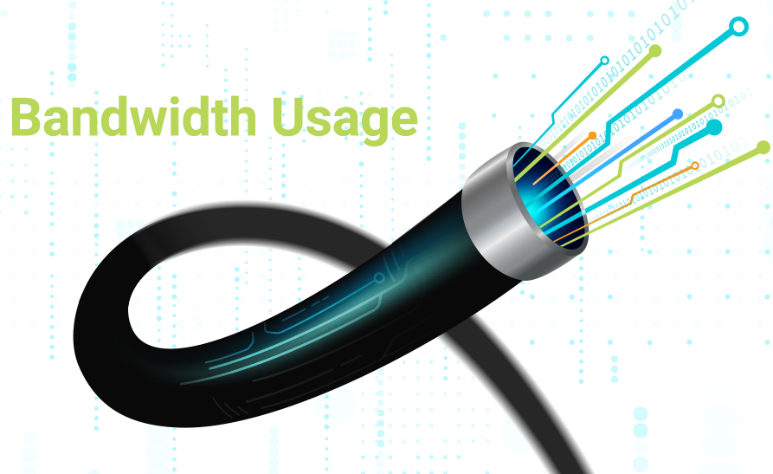
**Project outputs and results** 16

**Conclusions** 17

**Bibliography** 18

**Project introduction**

Our project is not a regular project. We are making an academic study which is on the verge of becoming an academic article. This includes a research of a never tried before method for reducing bandwidth usage.



There has been and still are many attempts to “save” data going through the internet by compressing it.



We decided to research the future\present problem of the extreme amount of data being sent at autonomous cars network:

Autonomous cars are all connected in one network, all of the cars are connected to one mainframe.

At the moment, the cars are constantly sending information to the server, information like its GPS location and\or the road that the car sees through its camera.

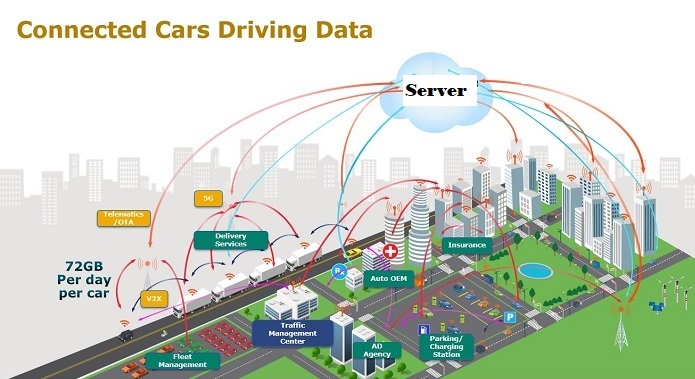
That is an enormous amount of data, that is one of the reasons we chose to focus at this field of work.

We did not know where our research will take us. We hoped for good results, but we were ready to just cling on every benefit that we will find on the way. And even- fail.

Eventually- we succeeded, and we wish to write an academic article.

The World’s Problem

Imagine a network with many nodes(millions), all are connected, sending data to each other. consider the data to be video transmission. That is a lot of data. Is there a way to relieve the usage of the bandwidth?



\*The numbers are just for example.

Our research, which focus on Video\Image transmission cost reduction, will add another layer of reduction, not just by using compression algorithms but by harnessing computer vision and machine learning- design a mechanism that waver insignificant data, thus reducing the bandwidth usage.

First, relieving the usage of the bandwidth takes us straight to a term we all know - “Compression”.

If we compress the information, we will relieve the bandwidth, right?

Right, but that is not all we can do-

What about not sending insignificant data?

That is the point we took off from, not being satisfied with compression, but adding another layer by not sending all the data, instead, sorting out the insignificant and not sending it from the first place.

Our data is video transmission. Video consists of images. by transmitting 24 images in one second, the human eye connects the images to one flowing picture.

In our work we will first dismantle the video to images then process the images separately.

Our server will know the roads, will know what to expect from each GPS location, we will teach it that by using test cars and machine learning. By learning the roads only one time, we will save a lot data in the future.

We will use image recognition and image processing algorithm to compare images.

The server will tell the client which parts he does expect to receive, and the client will know the insignificant part to not send back to the server.

How Others Solved It

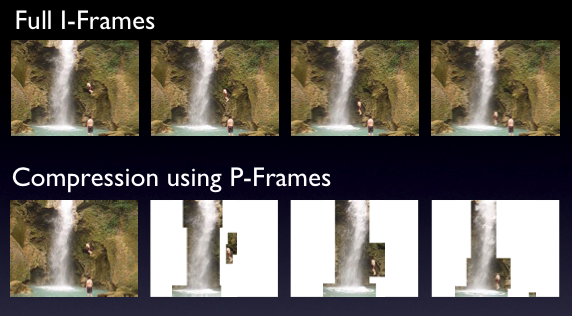
Other type of video transmission relieves use prior knowledge of a different kind then ours- they use previous frames of the same video in order to predict what “needs” to be sent next.

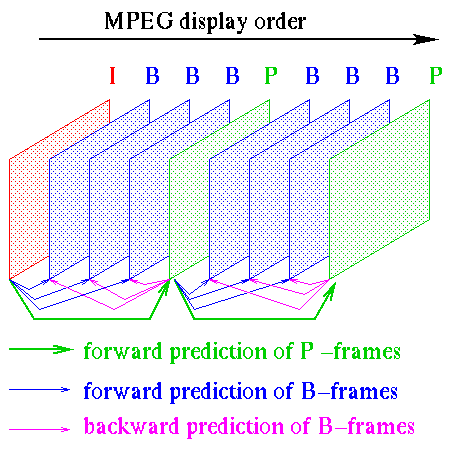
The algorithm examines moving parts of the frames and transmit only them and not the static parts of the image, assuming they will remain the same, then concatenate the old static ones with the newly transmitted moving parts, as you can see at the picture below.

These are great algorithms, but regarding our world’s problem, they come in short.

The roads are constantly changing while driving, mentioning turns and different angles.

This is the reason we have come up with our system, that will be displayed here next.





# **Our solution**

Instead of finding a way to compress the information, we use a different approach. using prior knowledge still but different, and by pre-process that information, including divide the video to frames and work on each frame separately, we will save data across the network.

We have achieved much better data saving by learning the roads themselves before-hand, so we know what image to expect.

# **Proof of concept**

We have built a system that simulates a self-driving car, sending footage information to a remote server. The server, which we taught the way beforehand. The car will send to the server only parts that had changed in each frame.

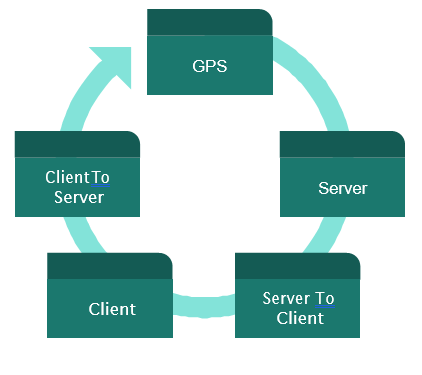
Part A- learning the roads:



Our systems use object detection algorithm and keypoints detection in order to detect static objects that does not varies for the same GPS location. The ones that does not varies are being saved in a very compressed way in the server under that specific GPS location as value in the DB, ready to be transmitted to a client going through this location.

\*the data saved in the server are Keypoints, Objects that describe an image part and not the image part itself. Keypoints can be compared.

Next, a continuous cycle, Part B:



The client send its GPS location to server which receive that information and look it up in its DB, it takes the appropriate value and send it back to the client, the client makes it’s computations and send the results back to the server, and the cycle continues with the next GPS location.

# **Academic Justifications**

As mentioned before our project is an academic research, we had an idea of what we want to achieve but searched ways of how to implement it.

We start of by trying to compare images, we knew this will be the bases of our work.

As students of “Data science” we decided 2 things:

1. Using python
2. Search for algorithm of machine learning to compare images.

We started by taking 2 images, slicing them to parts and comparing parts, saving the results in DataFrames and making many processing on these DataFrames.

So far, we used python Pandas, Numpy, OpenCV.

We understood significant problem in our system so far and we pivoted our direction-

* Simultaneously, we researched about LSH(Locality sensitive-hashing), an algorithm that hashes the data so in turn, when you wish to compare it, you won’t need it to be exactly the same in order to get positive result, like in normal hashing. This too was not a success and we chose to take another path.

We then started Studying SIFT\SURF algorithms, these algorithms being used to find keypoints in images. We worked with the python’s OpenCV version of them.

They find the key points, describe them as comparable descriptors, and by using KNN, compared them, it worked very well.

KeyPoint also consist of the x,y location of the point- and by using DBSCAN we then clustered the points. By clustering the points together, we could better compare if that part of the image is in another image.

We then finally improved our KeyPoints search by adding another layer of Object Recognition using YOLO object recognition algorithm.

For pasting the images together, we used OpenCV algorithm and image processing.

We also needed to detect objects that not being found, by other object- we then build an algorithm that uses triangulation in order to detect that object.

We finally tested our result by comparing the JPEG size of the output images and the Keypoint and objects number between the original image and the pasted image.

# **System description- by phases, code walkthrough up next**

## 

## 

## 

## 

## 

# **Our system, code walkthrough:**

Our system is a Server-Client system.

The server is connected to multiple client while the clients are connected to the server only.

Eventually the link between them will be based on the already existing connection that automobile companies currently have.

Our DB will be Cassandra.

1. Preprocessing at server side:

1.1 For each image with same GPS, make arrays of similar images(there can be images in the same GPS location which are completely different, for example, bridge)

1.2 for each array search for objects and key point.

1.3 make an intersect of them all, those who are find in every image of the array.

1.4 save them at the database under that GPS location, the data being save is in KeyPoint format.

## 2. client driving- send his GPS location to the server

3. The server search in the DB for the GPS location and send the result (KeyPoint) back to the client.

4. Client side-

4.1 search for objects and key points on the image the driver’s camera sees.

4.2.1 Compare the result from 4.1 with the data received from the server.

4.2.2 remove similar parts that was detected in 4.2.1, save the cropped image for later.

5. Client Side

5.1 on the cropped image search for objects and key points, cluster them and remove them as well.

5.2 make new image from the cropped image from 4.2.2 , minus the image from 5.1, we will be left only with the part we wish to send to the server. Parts that has been changed.

5.3 send the leftover image back to the server. With the information about the parts that has been discovered as similar.

6. Server Side – receive that image and the information about the parts and paste them together.

# **Sequence diagram**

**Phase A – Pre proccessing**



Client

proccess

Server DB

Key-GPS

Return Value

# **The algorithms we used:**

**Sift/Surf**

The **scale-invariant feature transform** (**SIFT**) is a [feature detection](https://en.wikipedia.org/wiki/Feature_detection_(computer_vision)) algorithm in [computer vision](https://en.wikipedia.org/wiki/Computer_vision) to detect and describe local features in images.

**SIFT** can robustly identify objects even among clutter and under partial occlusion, because the SIFT feature descriptor is invariant to [uniform scaling](https://en.wikipedia.org/wiki/Scaling_(geometry)), [orientation](https://en.wikipedia.org/wiki/Orientation_(geometry)), illumination changes, and partially invariant to [affine distortion](https://en.wikipedia.org/wiki/Affine_transformation).

**SURF** (**speeded up robust features** ) is inspired by the [scale-invariant feature transform](https://en.wikipedia.org/wiki/Scale-invariant_feature_transform) (SIFT) descriptor. The standard version of SURF is several times faster than SIFT and claimed by its authors to be more robust against different image transformations than SIFT, but scaling.

### **Object Detection**

A computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos. We specifically used the YOLO (You only look once) regression algorithm**,** commonly used for real-time object.

### **DBSCAN**

Density-Based Spatial Clustering of Applications with Noise It is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together, the black dots are marking as outliers points that lie alone in low-density regions.

**Epsilon variable** - The maximum distance between two samples for one to be considered as in the neighborhood of the other. This is not a maximum bound on the distances of points within a cluster. This is the most important DBSCAN parameter to choose appropriately for your data set and distance function.

**Min\_samples variable** - The number of samples (or total weight) in a neighborhood for a point to be considered as a core point. This includes the point itself.

### **OpenCV**

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision.

### **KNN**

In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), the ***k*-nearest neighbors algorithm** (***k*-NN**) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). In both cases, the input consists of the *k* closest training examples in the [feature space](https://en.wikipedia.org/wiki/Feature_space). The output depends on whether *k*-NN is used for classification or regression:

-In *k-NN classification*, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its *k* nearest neighbors (*k* is a positive [integer](https://en.wikipedia.org/wiki/Integer), typically small). If *k* = 1, then the object is simply assigned to the class of that single nearest neighbor.

-In *k-NN regression*, the output is the property value for the object. This value is the average of the values of *k* nearest neighbors.

# **Examples**

## **What do we see here?**

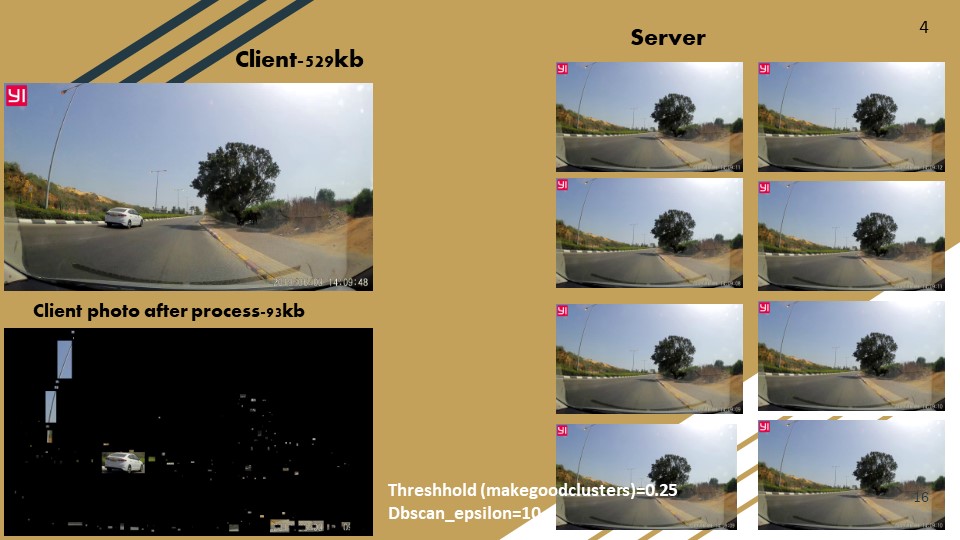
On the next few examples we will present how our project work. On the right the images which has been used in phase A in order to learn the road.

On the upper left is the image that the client sees.

On the bottom right is the output of the client’s calculations which will be sent to the server.

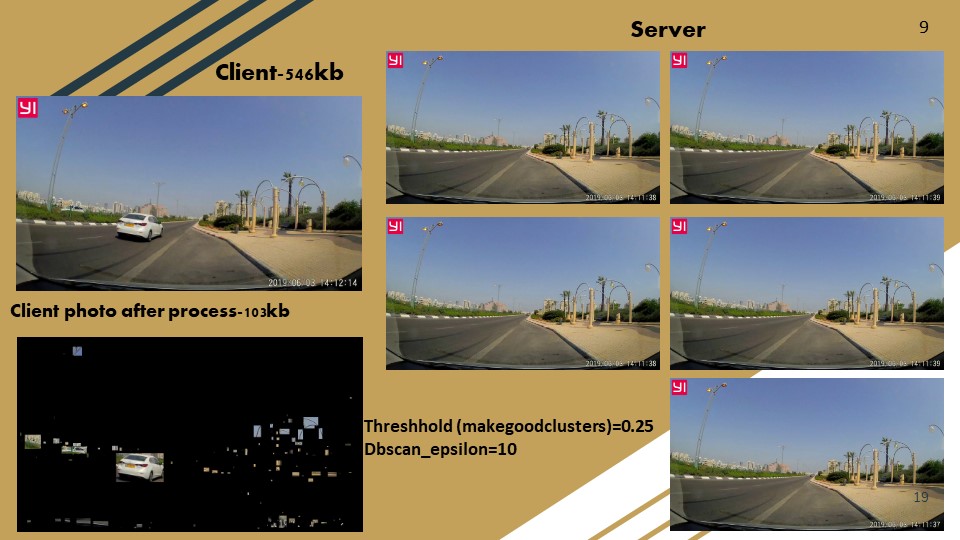








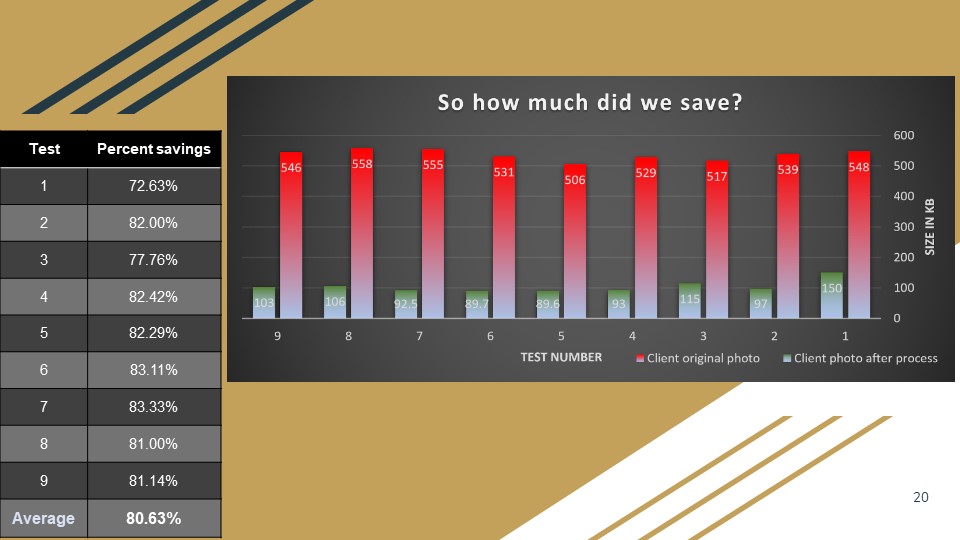




# **Project outputs and results**

## Performance graph

### Results



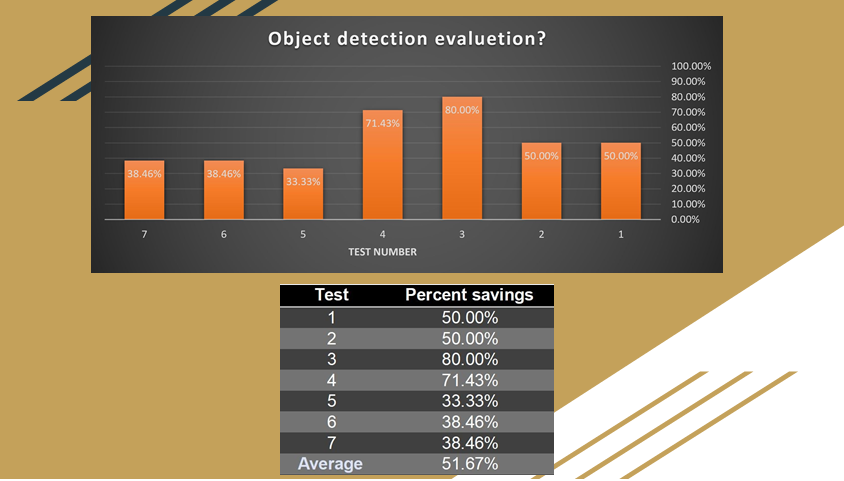
# **Conclusions**

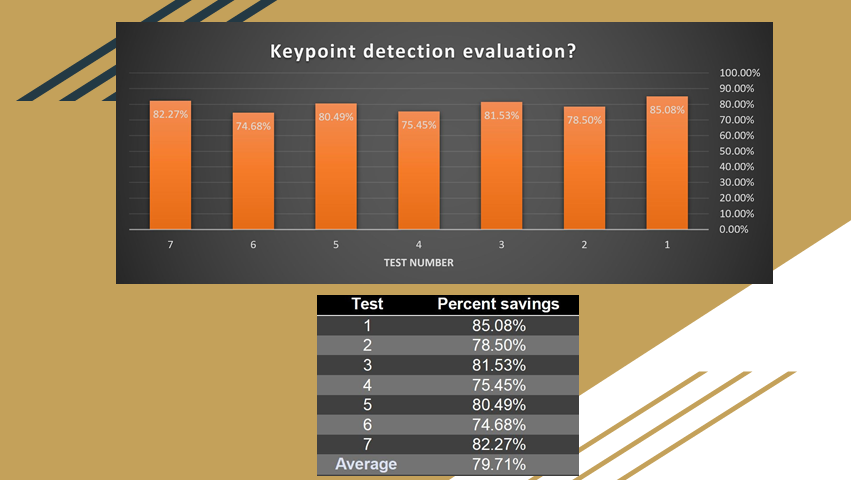
The result are a success.

In term of savings, as we can see from the graph, we averaged 80% saving.

Which mean that the data that will be send over the network will be 80 % smaller.

**Correctness evaluation:**





**Conclusion from the correctness graphs:**

These graphs show how much we did well by sending the important areas of the photos.

Although there is room for improvement certainly at the object detection phase, we see great outcome in the later, more important phase of the key point detection.

# **Bibliography**

***Image manipulation***

resize- <https://www.smartdraw.com/uml-diagram/>

Split - [https://stackoverflow.com/questions/43565275/split-and-join-images-in-python#](https://stackoverflow.com/questions/43565275/split-and-join-images-in-python)

Stiching - <https://www.pyimagesearch.com/2018/12/17/image-stitching-with-opencv-and-python/>

***OpenCV***

Distance between objects in an image - <https://www.smartdraw.com/uml-diagram/>

Delaunay Triangulation – helped forming the triangulation algorithm - <https://www.learnopencv.com/delaunay-triangulation-and-voronoi-diagram-using-opencv-c-python/>

***Sift \ Surf***

<https://www.researchgate.net/publication/314285930_Comparison_of_Feature_Detection_and_Matching_Approaches_SIFT_and_SURF>

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html>

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html>

SIFT Lecture - <https://www.youtube.com/watch?v=NPcMS49V5hg>

Sift and Surf comparison article - <https://www.researchgate.net/publication/314285930_Comparison_of_Feature_Detection_and_Matching_Approaches_SIFT_and_SURF>

***DBSCAN***

<https://scikit-learn.org/stable/auto_examples/cluster/plot_dbscan.html>

***Object Detection***

<https://www.youtube.com/watch?v=4eIBisqx9_g&t=67s>

***LSH***

Article - <http://www.scs.stanford.edu/16wi-cs240h/projects/douglass.pdf>

Implementation explanation -<https://engineering.purdue.edu/kak/distLSH/LocalitySensitiveHashing-1.0.1.html>

Python implementation - <https://github.com/mattilyra/LSH>

***General***

Impact of generation 5 over data transfer - <https://www.themarker.com/technation/tech-terms/1.5908199>

What will follow:

We had great results in our tests as the graphs above show, in order to be closer to the real world we will do -**more challenging tests**.

We need to keep an open mind about the many possibilities and field we can contribute at, we will -**consider add another field of work beyond autonomous cars network,** as suggest also by our supervisors.

Although we search thoroughly about similar works which try to gain the same end results as our, or which are using our method, we have not found, yet. That does mean that there are none. We will continue and do- **more checks about similar works**.

As our supervisor suggested, cooperation is KEY, there is another group working on trying to save data, in a different matter then us. We will -**check combine work with different project of data usage saving.**

In our code we used many open source codes, as long as we keep our algorithm just for academic purposes that is fine but if we will want to publish and even make a product out of it, we need to -**check for patents and rights**.

Finally, as we started off with, we wish to continue studying the subject and develop our work and eventually -**write an academic article**.