**Advanced Topics on Computer Vision**

# 3D Head Pose Estimation in Real Time Classical Approach

**Short Project**

May $30^{th}$ 2024

**Autors:** Arias Mitjà, Marc Zoel
Navarro Soler, Yeray


Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona

# Contents

## Abstract

This project presents a real time 3D head pose estimation system to assist physically impaired individuals to select objects from kitchen shelves. The system uses classical approaches for head pose estimation, to determine which direction the human head is facing and projects into the shelves a user-driven dot for visual feedback. Developed entirely in Python, two approaches were explored: a simple 2D approximation of face landmarks to compute some ratios, and computing 3D perspective transformation solving the Perspective-n-Point problem.

The *First Approach* provides simplicity and low computational cost, performing really good in the central YAW and PITCH angles, but underperforming in outer ones. The *Second Approach* implies more computational load and shows high sensitivity in the central angles, whereas achieves a high precision in the extremes ones. Mediapipe's Face Landmarker facilitates real-time detection, with performance enhanced through filtering techniques and attractors to facilitate object selection.

Although the system has potential, its efficacy is restricted by specific constraints, such as camera positioning and specific calibration. Future work can be directed towards improving the adaptability of the system to variable environment conditions and combining both methods appropriately to obtain a more robust system.

# 1 Introduction

## 1.1 Description of the problem to be solved

There is a current project in the *Departament d'Enginyeria de Sistemes, Automàtica i Informàtica Industrial* which consists on implementing an assistance system for helping physically impaired people to reach objects in the kitchen shelves. After the user selects an object, a robot arm will retrieve it and give it to the user. In order to select the object that the user wants to retrieve, a vision system wants to be implemented, by determining which is the object that the user is looking to. Therefore, a head pose estimation system that enables to select the different objects on the shelves needs to be created.

Although this project will be focused in head pose estimation, its main objective will not be to obtain a minimum error in the estimation of the real angles of the head's orientation, but to make a system that works well to the purpose previously presented.

## 1.2 State of the art and proposed solution

Nowadays, the head pose estimation task can be done basically by two different approaches. The first one consists on tracking facial landmarks, whereas the second one consists on using deep learning algorithms, getting an end-to-end solution.

**Classical approach:** The classical approach consists mainly on tracking facial landmarks. This approach is highly conditioned by the acquired landmarks, therefore by the landmark detection performance. Also, it depends on the approach that is selected in order to compute the head's rotations from these landmarks. Also, the camera internal parameters might be needed.

**Deep Learning:** The newer approach involves the use of a deep neural network to detect the head pose directly from images. Nowadays, there are already some models, for example WHENet [1] or Hopenet which has three neural network branches, one for each Euler angle and has been trained with three loss functions, one for each branch [2].

The classical approach will be used for this project, and different rotation's computation methods will be studied and compared. This approach consists on three different parts, which more detailed explanation on how it works is presented below [3, 4, 5]:

1. **Face detection:** to find the regions of interest.

2. **Facial landmarks detection:** The landmarks are facial keypoints such as eyes, eyebrows, nose, lips, etc. Extracting and scaling these coordinates appropriately is a pivotal aspect of the process.

3. **Head pose estimation:** Once the landmarks are extracted, the head's rotation must be computed. To do so, different methods can be applied, which two of them will be studied in this project.

To summarize, this project will focus on creating a 3D head pose estimation system in real time. To be able to receive feedback from where the user is looking to, and the object to retrieve, a projector will be displaying into the shelves a user-driven point, that has the information about the head's orientation. Furthermore, for this project, the camera will always be considered to be perpendicular and centered with respect to both the user and the scene.

# 2 Explanation and implementation of the selected method

## 2.1 Mediapipe

Mediapipe is an open-source framework developed by Google, which enables to build pipelines to perform artificial intelligence (AI) and machine learning (ML) techniques [6]. There is a big variety of solutions offered including audio, text and computer vision tasks [7]. The solutions can be introduced into the user applications and can be customized to fulfill the specific needs of the task in hand. Mediapipe also offers pretrained models for almost all the tasks, that can be used across multiple development platforms.

For this project, the *Face Landmarker* [8, 9] of Mediapipe will be used . This task allows us to detect the faces in an image and compute the landmarks of all of them. It also includes identification of human facial expressions. The *Face Landmarker*, which input can be images, videos or live video feed, outputs 3-dimensional face landmarks and facial expressions scores.

As the purpose of the project is to create a real-time video system, the input used will be live video feed directly from the computer webcam. In our case, the only relevant output of the *Face Landmarker* are the landmark's coordinates.

The extracted landmarks form a face mesh that can be represented graphically (see *Figure* 1). Also, the landmarks are always in the same specific order, but this order was not known, so the first thing was to determine the order of the landmarks (see *Annex* 5.1). The selection of the different landmarks used will be explained in the following sections, since not all landmarks will be used for each approach, but only some of them.

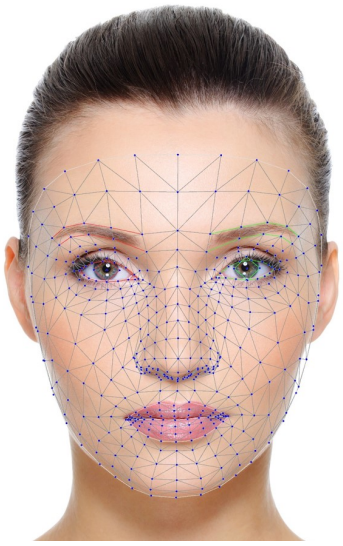


Figure 1: Face mesh

## 2.2 First Approach

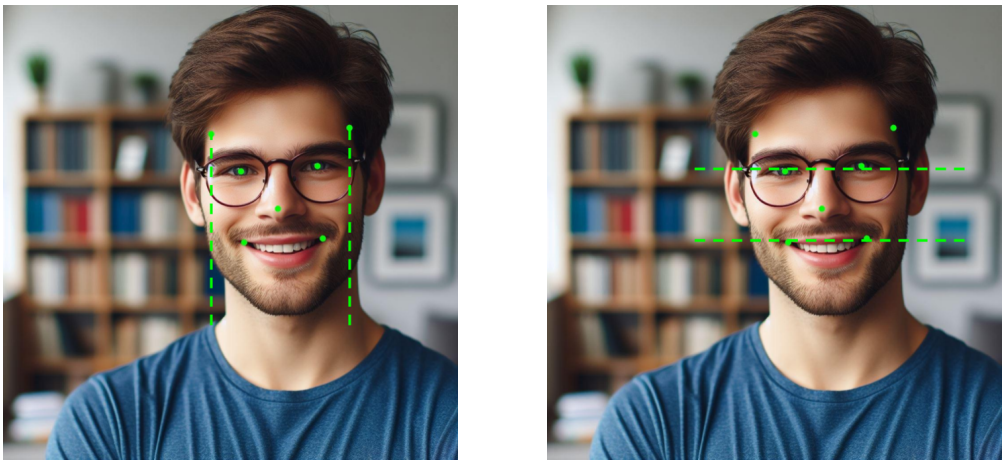
This method consists in estimating the face 3D pose by means of the projection of the most relevant landmarks into the 2D plane. Once these are projected, some ratios are computed in order to estimate the head orientation. This is a simple approach, with a low computational cost and quite acceptable results, but it also has some inaccuracies due to the lack of information, as we are considering only the X-Y location with respect to the camera and ignoring the Z distance.

First of all, we selected the most relevant landmarks obtained by means of Mediapipe, according to the 2.1 section. The main challenge here is to find those keypoints in the human face that are located in the same regions for any human face, regardless of its physiognomy. It is also important to find those landmarks which its position variation with respect to the others remains linear from frame to frame, as ratios will be computed taking this information.

(a) Original image (b) Marked image

Figure 2: Relevant landmarks extraction

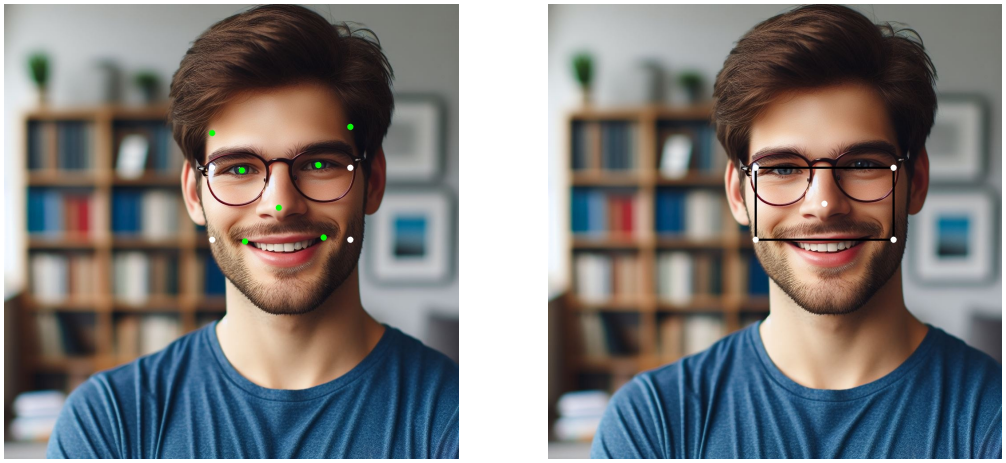
To decide the landmarks to be used, checking for the best performance have been done in an iterative way, trying several landmarks, but at the end, a total of 7 keypoints are considered, which are shown in *Figure 2b*. These are used to obtain the corners of the rectangle that will be used to estimate the face 3D pose. Concretely, the forehead landmarks will be projected vertically to fix the X components of the corners of the rectangle, i.e. the amplitude of the rectangle. Then, the mean of the Y component of both eyes and mouth landmarks will be computed. These values will be projected as to fix the Y components of the corners of the rectangle, i.e. the altitude of the rectangle. This procedure is shown in the figure below.



(a) Vertical (b) Horizontal

Figure 3: Landmark projections

The coordinates of the corners of he rectangle are determined by the intersections of the green dashed lines of *Figure 3*.



(a) Corners (b) Rectangle

Figure 4: Rectangle definition

Now, we dispose of a reference rectangle which will be updated for each frame. The location of the nose landmark will be compared with the center of the rectangle in order to obtain a relative distance and estimate the YAW and PITCH rotation of the head. Notice that ROLL angle is not being considered, that is the reason why the rectangle is forced to be always aligned with the X-Y axis, having its orientation fixed even if its size changes.

The rectangle and its magnitudes are also normalized, in order to obtain distances from 0 to 1. Then, this relative distance is converted to degrees by means of the following formula:

$$YAW = \frac{nose\_coordX - centre\_coordX}{0.5 \cdot rectangle\_width} \cdot yaw\_range \tag{1}$$

$$PITCH = \frac{centre\_coordY - nose\_coordY}{0.5 \cdot rectangle\_height} \cdot pitch\_range \tag{2}$$

Where $nose\_coord = (nose\_coordX, nose\_coordY)$ is the nose landmark and acts as a mobile point, $centre\_coord = (centre\_coordX, centre\_coordY)$ is the center of the rectangle and acts as the reference point, $rectangle\_width$ and $rectangle\_height$ are the amplitude and altitude values of the rectangle, respectively.

The $yaw\_range$ and $pitch\_range$ are constant values and correspond to the considered ranges of degrees that have been calculated empirically, analyzing the situation when the nose reaches the limit of the rectangle, i.e. the maximum considered angle. This analysis was performed over an own video in which the head rotates from 0º to 90º in YAW following a MCU trajectory. Rotation from 0º to 45º in PITCH was recorded as well, although it was more difficult to be precise in this axis. We decided to set the $yaw\_range$ to 41.94º and $pitch\_range$ to 22.5º, which are consistent with the natural or comfortable range of vision of a person in the kitchen and seem to minimize the error when comparing the obtained angles with labeled images from a dataset on the Internet, as detailed in future sections.

Since we normalized the rectangle magnitudes and distances, it remains invariant to the distance between the person and the camera. We realized that the rectangle was changing its dimensions frame to frame due to the Mediapipe landmarks noise. Then, it was decided to fix the rectangle -as it can be observed in the upper-left gray rectangle in *Figure 5*-, but a lot

of noise appeared on the nose landmark and, consequently, in the head pose estimation. This inconvenience will be addressed in future sections.

Regarding the landmarks used for this approach, both landmarks of the mouth can be substituted by a single landmark in the upper lift or between this point and the nose, but they have been discarded to have a more symmetrical range in the vertical movement with respect to the nose.

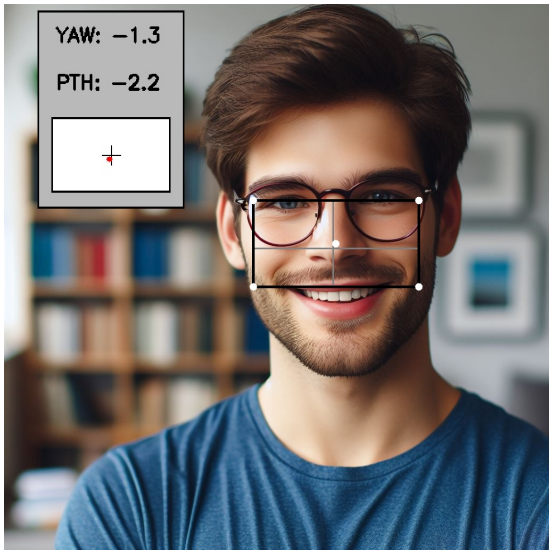
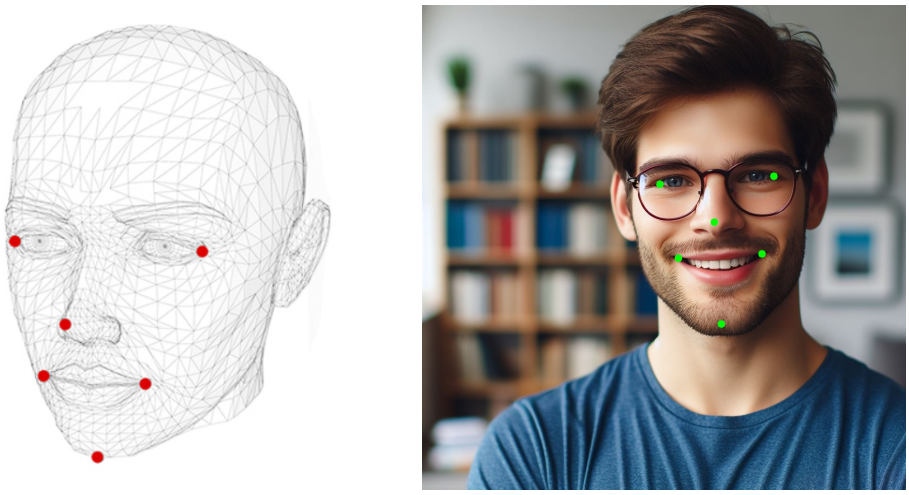



Figure 5: *First Approach*

### 2.3 Second Approach

This method consists in estimating the face 3D pose by solving the Perspective-n-Point (PnP) problem, which identifies the rotation matrix and translation vector between the real-world and the camera coordinates. With both rotation and translation, the movement of a 3D object can be completely described, and therefore, the head's orientation can be obtained. This is a more complex approach than the previous one, with more computational cost. The results are also quite acceptable, but have some inaccuracies as well.

The PnP problem consists on computing the translation and rotation that minimizes the reprojection error from 3D-2D point correspondences [10]. We will compute the head pose by taking into account 6 relevant points of the face, and making their correspondences with a generic 3D head model. The selected points are: tip of the nose, chin, right and left corners of eyes and mouth (see *Figure* 6a). Their 3D coordinates in the World Coordinates are known, and the corresponding landmarks can be retrieved (see *Figure* 6b).



(a) Head 3D generic model (b) Landmarks of the interest points

Figure 6: Points used for the PnP problem

The intrinsic parameters of the camera are also needed to solve the PnP problem, which are the focal length, optical center, and radial distortion of the camera. These parameters can be computed by calibrating the camera [11], or they can be approximated as follows: the focal length can be approximated by the width of the image in pixels, the optical center can be approximated by the center of the image and the distortion can be assumed to be null for the webcam of the computer.

Once having the 3D-2D pairs of points and the intrinsic parameters, the function `solvePnP` of the OpenCV library has been used. This function returns the rotation and translation vectors that allow transforming a 3D point expressed in the World Coordinates into the Camera Coordinates. After that, the rotation vector must be transformed into a rotation matrix, which has been done using the `Rodrigues` function of the same library. Then, this rotation matrix is decomposed into the rotated angles by using the `RQDecomp3x3` function.

Finally, the angles of yaw, ptch, roll of the face are obtained, and can be used for following computations. Even though this method is using a 3D generic model and solving the PnP problem, the values of the angles have a lot of noise and tend to change rapidly inside a little range. This can probably be caused because of the rapidly changing landmarks estimation done by the *Face Landmarker*. As commented before, this inconvenience will be addressed in following sections.

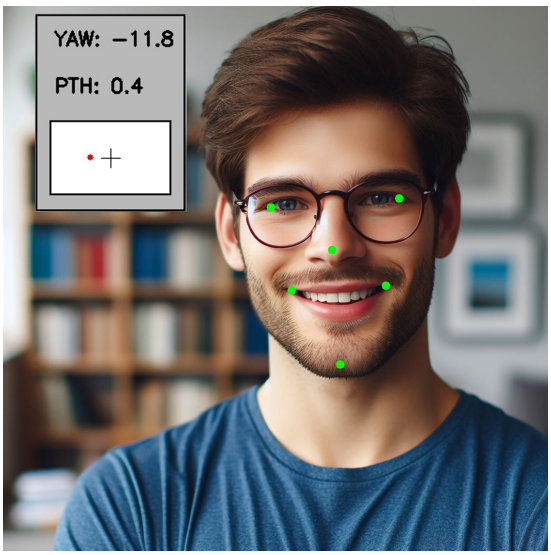



Figure 7: *Second Approach*

## 2.4 Homography

Once explained both considered approaches, it is time to project the solution into a real scenario, which is composed by a kitchen with a shelve and a projector plotting a dot on it. As it can be expected, the obtained solution in the PC screen will not correspond to the projected solution. Since perspectives does not match, so many transformations are needed, which are called *homography*.

The projector is located such that the image projected is allocated between the vertical plane of the shelves and the horizontal plane of the table next to the shelves. As only the shelve is considered, just the part of the projection that stays in the vertical plane will be used.

In order to compute the homography between the computer screen and the projected image, first, we measured the corners of the projection in the shelve in pixels by means of a python code. Once knowing these coordinates, we found out the proportion given by the ratio $height/width$, which was around 0.425. Then, we kept this proportion at the time of defining the available rectangular region in the PC: considering the whole width, we took $height = 0.425 \cdot width$.

We obtained the *homography* matrix $H$ by means of the *OpenCV* function:

$$H = cv2.findHomography(PC\_corners, Projector\_corners) \tag{3}$$

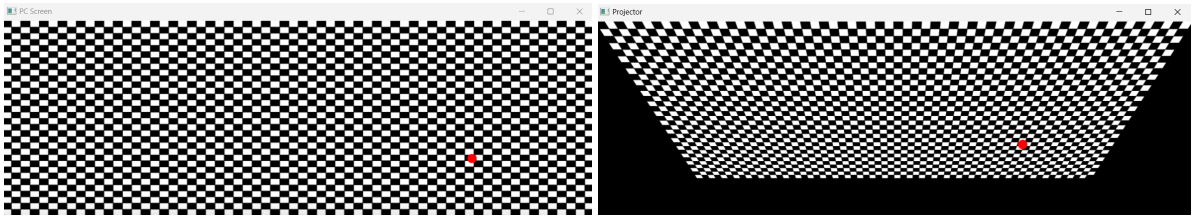
Where:

$$PC\_corners = [[0,0],[1920,0],[1920,816],[0,816]] \tag{4}$$
$$Projector\_corners = [[53,0],[1908,0],[1908,775],[39,808] \tag{5}$$

Both in pixels. Then, the obtained matrix $H$ is:

$$H = \begin{bmatrix} 1.008 & -0.018 & 53 \\ 0 & 0.983 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

The change of perspectives looks like:



(a) PC screen (b) Projector

Figure 8: Homography between the laptop and the projection

Note that the images presented before don't correspond to the homography matrix obtained, which is really similar to just a translation. The image has been exaggerated in order to be able to observe the transformation of the image conceptually.

### 2.5 Filtering

As it was commented in sections 2.2 and 2.3, the landmarks are affected by some noise due to Mediapipe key points fluctuation. In this section we try to solve this inconvenience by means of *recursive* filtering, which is defined as a weighted sum of the information extracted of the current frame with the one obtained on the previous frames.

To not apply this filter for all relevant landmarks, as this would take more computing power, it is directly applied on the computed head's rotations. The displayed rotations $R(n)$ are given by:

$$R(n) = \beta \cdot S(n) + (1-\beta) \cdot R(n-1) \tag{7}$$

Where $n$ denotes the frame number, $S(n)$ is the current measured signal and $\beta$ is a value from 0 to 1. The smaller the numerical value of $\beta$ is, the greater the weight from previous frames and the noise reduction is; thus, there is a greater potential for lag. The parameter $\beta$ is sometimes referred to as the "forget factor". The closer $\beta$ is to one, the more quickly the filter forgets the old input [12]. This means that we must find a compromise between smoothness -or noise reduction- and speed of tracking.

After many iterations, we observed that $\beta = 0.05$ provided a clean nose landmark evolution from frame to frame on which noise is totally attenuated. As it was expected, lag appeared in

quick head rotations: the nose landmark tracking was not enough fast as to follow the actual pose. This was solved by improving the *recursive* filter adding to it what we called the *dynamic* factor. If the angular velocity of the head is higher than a threshold equal to 2 or 6 degrees/frame for each approach respectively, the mode is set to *high*. Otherwise, the mode is set to *low*.

As we observed that modes were continuously switching, we also added the condition to register 5 consecutive frames in the same mode in order to carry out the switch. Notice that, as we have approximately 30 FPS, the switching verification is not affecting the real time response.

On one hand, *high* mode means that the head is moving at a certain velocity, which means that fast tracking is needed and accuracy is not relevant -some noise can be allowed-, so high $\beta$ values will fulfill these requirements. On the other hand, *low* mode reflects that the head is not moving i.e. is pointing to some object, what requires of a high accuracy -high noise reduction- while fast tracking is not relevant here -since the head is not moving-; low $\beta$ values will be required. This value must not differ so much between modes in order to ensure smooth switching between them and realistic results.

The values of $\beta$ were empirically determined and, the ones that give better results for each method and velocity mode, are summarized in the table below.

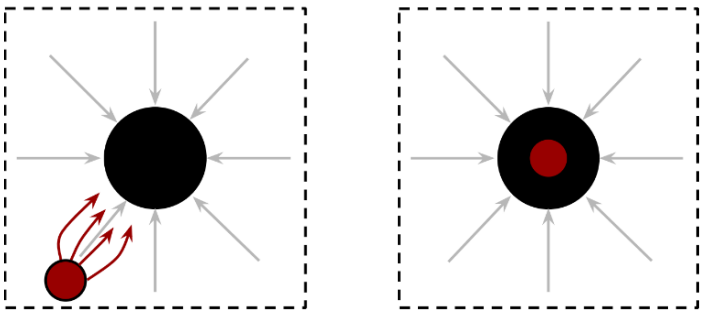
| | **High** | **Low** |
|---|---|---|
| **First Approach** | 0.4 | 0.05 |
| **Second Approach** | 0.3 | 0.05 |

Table 1: $\beta$ chosen values according to its approach and velocity mode

### 2.6 Attractors

This project is about creating a 3D head pose estimation system that enables a person to select the different objects on the shelves of a kitchen, as it was previously explained in section 1.1. Even having mitigated noise by means of the *recursive* filter, the user may not always feel enough accuracy when pointing to a concrete object on the shelve as we are admitting some error on the pose estimation. This is solved by means of *attractors*.

On this section we are defining regions around some concrete points -which are supposed to be in the shelves objects- in order to drive the nose landmark plot to its goal. This means that if the user is intentionally pointing closer to some object -in *low* velocity mode-, the system will correct the head pose estimation in order to select the desired object. Otherwise, this application may be so tough for the user if an extreme accuracy is required.



(a) Closer landmark (b) Attracted landmark

Figure 9: Attractor behaviour

The previous *Figure 9* represents the attraction effect applied to a landmark -red point- when it is near to an object -black point-.

The considered margins are 4 degrees in the X axis -YAW-, and 3 degrees in the Y axis -PITCH-. These values are fixed according to the $yaw\_range$ and $pitch\_range$ defined in section 2.2. The tolerance on the X axis is higher since the considered $yaw\_range$ is bigger, so its precision at pointing will be harder and it requires of some extra help.

# 3 Evaluation

## 3.1 Results and Analysis

Both approaches have been confronted in order to evaluate their relative efficacy, identify their strengths and weaknesses, and see how they perform under the same experimental conditions. This has been done by using both approaches at the same time with the same code. Two examples of the resulting images obtained in the computer screen can be found below.


(a) Confrontation 1 (b) Confrontation 2

Figure 10: Approaches comparison

Also, a code that enables switching between the two approaches has been created. By using it, the user can choose when to use each approach, and the differences between them can also be tested in a different way than explained before.

By confronting both approaches -as seen in *Figure 10*-, it was observed that both methods differ a bit and behave different in some regions. These differences are given by the essence of the code and are not caused by the deformation of the projector.

Concretely, the *First Approach* behaves smoothly and seems to be governed by a linear function, providing good results especially in central regions. Its sensitivity remains constant for any orientation along the screen, but it is hard to reach extreme YAW and PITCH values, i.e. the screen corners, with a proper accuracy. On the other hand, the *Second Approach* seems to act as determined by nonlinear functions and, unlike the previous method, is more effective at extreme YAW and PITCH values, i.e. the screen corners. It presents high sensitivity at central regions causing difficulty in precision in these regions. This could also been influenced by the applied filters, but we adjusted them to perform smooth switches between *high* and *low* velocity modes, and the higher sensitivity on the center remained.

The project has also been tested in a real scenario simulating a real kitchen. In order to verify the applied *homography*, a checkerboard was computed with the aim of verifying that the *homography* was correctly computed for each region in the screen, as seen in *Figure 11*.

Figure 11: Homography testing

After that verification, the actual code was used to begin testing the other aspects of the code. Notice that the projector is plotting not only the attractors in some of the objects on the shelves, but also the user-driven red dot (see *Figure 12*).

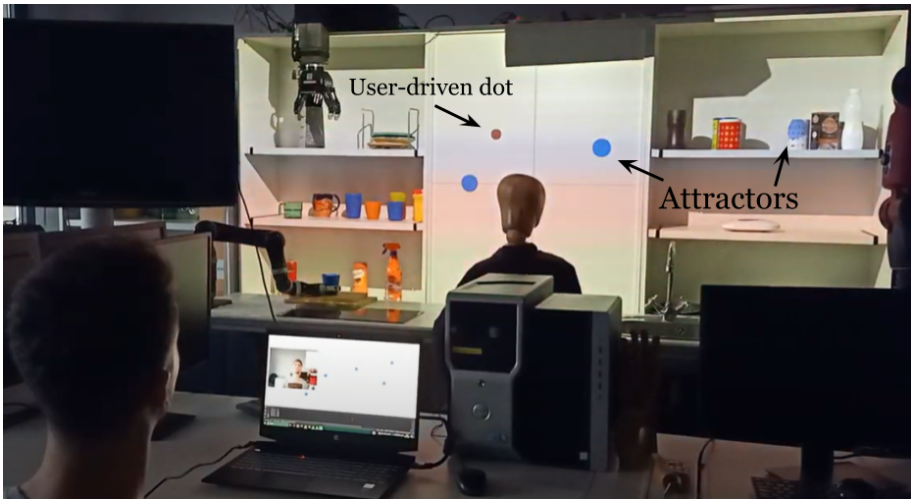



Figure 12: Testing the approaches in a real scenario

As commented before, since both approaches differed a bit, we could not get reliable conclusions on which was better. It is known that the *First Approach* is a simple 2D approximation while the *Second Approach* considers the 3D and computes some perspective transformations in order to make the estimation, being a more complete method.

Both methods have been tested computing the error between the measured angle and the labeled value from random images from the 300W_LP dataset [13]. The obtained absolute error is represented below (see *Table 2*) and the whole computation table is attached in *Annex* 5.2.

| | **YAW** | **PITCH** |
|---|---|---|
| **First Approach** | 5.91 | 5.50 |
| **Second Approach** | 5.88 | 5.35 |

Table 2: Absolute error for both approaches

The *Second Approach* could be taken as the actual head pose, but we also tried to validate it empirically using a laser pointer -which is considered to be the true positive-, pointing to the

shelve in the laboratory (see *Figure 13*).

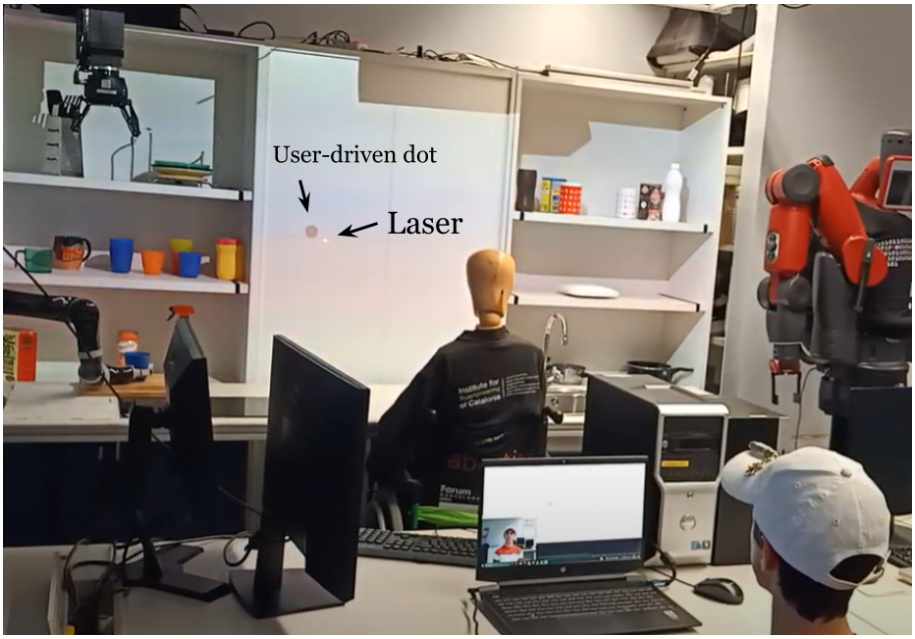



Figure 13: Using a laser pointer to test the accuracy of the 3D head pose estimation approaches

Even with the laser, we cannot determine which is the best method given the features explained in this section. The optimal solution may be a combination of both, where *First Approach* is used in the central zone and *Second Approach* at external regions.

### 3.2 Discussion

The system developed during this project will be used by some user. That means that the system must respond in a fast and precise manner to the user orders. As users of the developed system, we could verify that the time response of the system was low enough to consider it to be real time, as the red dot moved with enough velocity. In addition, the improvements made with the filters and attractors have been key to achieve a better user experience. They produce that when being static in a certain position, the dot hardly moves, and trying to select an object is a much easier task with the help of the attractors.

Although the system works well despite small displacements of the camera with respect to the user, that is, it does not affect that the camera is further or closer, more at the right or more at the left, the camera always need to be perpendicular to the user. Also, as the camera has been considered to be always perpendicular and centered with respect to the shelves, when those conditions are changed, the system does not perform well enough, as it does not take into account the relative position between user, camera and scene.

As an extension, after comparing the methods, a code was created that mixed both of them, using each in the areas that produced better results. This code, even though is useful and might be the most robust solution, needs a lot of tunning to really make a smooth transition between the approaches.

### 3.3 Future Work

In order to improve the results obtained in this project, some further work can be done:

- Calibrate the $yaw\_range$ and $pitch\_range$ to adjust it to the range of motion of the person and the particular scene.
- Calibrate the range of the attractors to adjust it to the user and scene.
- Improve the mixed approach composed by both methods to obtain a more robust control over the projected point.
- Improve the methods by taking into account the relative poses between the camera, the user and the scene.

## 4 Conclusions

The main objective of this final project was to develop a computer vision system to estimate the 3D head pose in real time, capable of assisting physically impaired individuals to select objects from kitchen shelves. To do so, the development have been done using Python.

The *First Approach* is a simple 2D approximation with low computational cost. It performs well in the central regions but underperforms with extreme values of the YAW and PITCH angles. Despite that, this method maintains a constant sensitivity across all orientations.

The *Second Approach* consists on solving the PnP problem to obtain the rotations of the head, thus it has more computational load. It performs better in the extreme values of YAW and PITCH, whereas has really high sensitivity in central regions, making it difficult to acquire a high precision to control it.

Given that the strengths and weaknesses of both methods are complementary, a combination of them might be the way to create a more robust system. The *First Approach* could be used for central regions, whereas the *Second Approach* could be used for the outer ones.

The homography to correct the projected image has been computed and applied successfully to all the images. The system time response is low enough to consider it real time, enhancing user interaction. Improvements such as filters and attractors are also key to a better user experience. Despite that, the camera must remain perpendicular and centered with respect to the scene and the user to achieve a proper functioning.

The developed system must be calibrated for each user and scene to achieve the best performance. Also, this work can be improved by combining both approaches as mentioned, and taking into account the relative pose between the user, camera and scene, to make it usable in every possible situation.

In summary, the 3D head pose estimation system developed in this project has potential for assisting in the selection of objects in the kitchen shelves through visual tracking and projection. Nonetheless, further improvements are needed to achieve a more robust system that can operate in any possible situation.

## Bibliography

[1] Yijun Zhou, James Gregson. WHENet: Real-time Fine-Grained Estimation for Wide Range Head Pose, IC Lab, Huawei Technologies Canada [Internet]; 2020. [Accessed 10-03-2024]. Available from: https://arxiv.org/pdf/2005.10353v2.pdf.

[2] Nataniel Ruiz, Eunji Chong, James M Rehg. Fine-Grained Head Pose EstimationWithout Keypoints, Georgia Institute of Technology [Internet]; 2018. [Accessed 10-03-2024]. Available from: https://arxiv.org/pdf/1710.00925v5.pdf.

[3] Anastasiya Zharovskikh. Head Pose Estimation with Computer Vision [Internet]; January 2021. [Accessed 10-03-2024]. Available from: https://indatalabs.com/blog/head-pose-estimation-with-cv.

[4] Jaykumaran R. Real-Time Head Pose Estimation FaceMesh with MediaPipe and OpenCV: A Comprehensive Guide [Internet]; September 2023. [Accessed 10-03-2024]. Available from: https://medium.com/@jaykumaran2217/real-time-head-pose-estimation-facemesh-with-mediapipe-and-opencv-a-comprehensive-guide-b6

[5] datagen. Head Pose Estimation: Use Cases, Techniques, and Datasets [Internet];. [Accessed 10-03-2024]. Available from: https://datagen.tech/guides/pose-estimation/head-pose-estimation/#.

[6] Mediapipe - Google. MediaPipe Solutions guide [Internet];. [Accessed 02-05-2024]. Available from: https://ai.google.dev/edge/mediapipe/solutions/guide.

[7] Mediapipe - Google. On-device ML for everyone [Internet];. [Accessed 02-05-2024]. Available from: https://mediapipe-studio.webapps.google.com/u/1/home.

[8] Mediapipe - Google. Face landmark detection guide [Internet];. [Accessed 02-05-2024]. Available from: https://ai.google.dev/edge/mediapipe/solutions/vision/face_landmarker.

[9] Mediapipe - Google. MediaPipe Face Mesh [Internet];. [Accessed 02-05-2024]. Available from: https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/face_mesh.md.

[10] OpenCV. Perspective-n-Point (PnP) pose computation [Internet];. [Accessed 02-05-2024]. Available from: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html.

[11] Nicolai Høirup Nielsen. Camera Calibration [Internet];. [Accessed 02-05-2024]. Available from: https://github.com/niconielsen32/CameraCalibration.

[12] Bente Konst, Jacob Nøtthellen, Stine Nalum Naess and Magnus Båth. Novel method to determine recursive filtration and noise reduction in fluoroscopic imaging – a comparison of four different vendors [Internet]; Dec 2020. [Accessed 25-05-2024]. Available from: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7856489/.

[13] Xiangyu Zhu, Zhen Lei. Face Alignment Across Large Poses: A 3D Solution [Internet];. [Accessed 02-05-2024]. Available from: http://www.cbsr.ia.ac.cn/users/xiangyuzhu/projects/3DDFA/main.htm.

# 5 Annexes

## 5.1 Mediapipe's Face Landmarker results - Landmark Indices

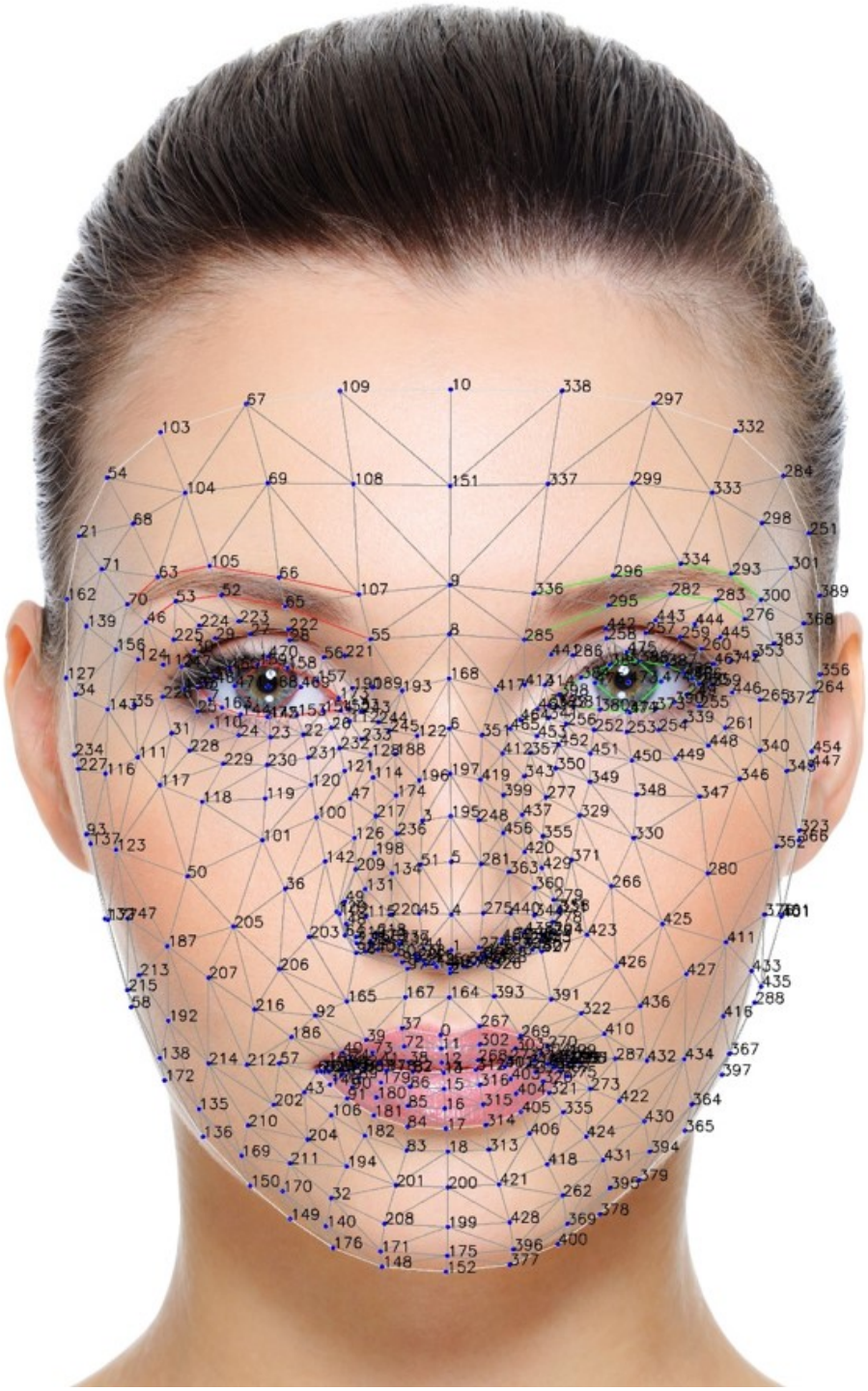


Figure 14: Landmark indices

## 5.2 Approaches error comparing with the 300W_LP dataset

| Filename | Yaw Error | Pitch Error | Abs YE | Abs PE |
|---|---|---|---|---|
| AFW_134212_1_0,mat | 9,90 | -11,30 | 9,90 | 11,3 |
| AFW_134212_2_0,mat | -4,60 | -17,50 | 4,60 | 17,5 |
| AFW_1130084326_2_0,mat | -5,40 | -4,80 | 5,40 | 4,8 |
| AFW_156474078_1_0,mat | 0,10 | 9,10 | 0,10 | 9,1 |
| AFW_1372188757_1_0,mat | -9,40 | -13,30 | 9,40 | 13,3 |
| AFW_16413031_1_0,mat | -6,60 | -9,20 | 6,60 | 9,2 |
| AFW_170817766_2_0,mat | -4,70 | 0,90 | 4,70 | 0,9 |
| AFW_18489332_1_0,mat | -0,50 | -6,30 | 0,50 | 6,3 |
| AFW_199759840_1_0,mat | 2,80 | 1,10 | 2,80 | 1,1 |
| AFW_213654866_1_0,mat | -5,10 | -9,30 | 5,10 | 9,3 |
| AFW_22699817_1_0,mat | -0,50 | -4,70 | 0,50 | 4,7 |
| AFW_2353849_1_0,mat | -12,20 | -1,20 | 12,20 | 1,2 |
| AFW_237815567_4_0,mat | 0,40 | -2,10 | 0,40 | 2,1 |
| AFW_24795717_2_0,mat | -7,60 | -5,70 | 7,60 | 5,7 |
| AFW_24795717_3_0,mat | -10,10 | -8,50 | 10,10 | 8,5 |
| AFW_255360810_1_0,mat | 8,60 | 2,90 | 8,60 | 2,9 |
| AFW_261068_1_0,mat | -7,30 | -1,70 | 7,30 | 1,7 |
| AFW_261068_2_0,mat | 2,40 | -2,70 | 2,40 | 2,7 |
| AFW_270814102_1_0,mat | 2,10 | -3,50 | 2,10 | 3,5 |
| AFW_281972218_1_0,mat | 4,00 | -14,00 | 4,00 | 14 |
| AFW_342149506_1_0,mat | -3,90 | 1,60 | 3,90 | 1,6 |
| AFW_346731834_1_0,mat | -2,50 | 4,70 | 2,50 | 4,7 |
| AFW_3989161_1_0,mat | 13,90 | -1,60 | 13,90 | 1,6 |
| AFW_45092961_2_0,mat | 8,10 | -0,80 | 8,10 | 0,8 |
| AFW_815038_1_0,mat | 6,10 | -0,60 | 6,10 | 0,6 |
| AFW_45092961_1_0,mat | -8,70 | -2,90 | 8,70 | 2,9 |
| AFW_59354095_2_0,mat | 0,40 | -9,10 | 0,40 | 9,1 |
| AFW_507521047_1_0,mat | 15,60 | -10,00 | 15,60 | 10 |
| AFW_719902933_1_0,mat | -3,60 | -3,80 | 3,60 | 3,8 |
| AFW_442651885_1_0,mat | 12,20 | 4,10 | 12,20 | 4,1 |
| AFW_5452623_1_0,mat | 4,00 | -1,40 | 4,00 | 1,4 |
| Error Average | | | **5,91** | **5,50** |

Figure 15: First approach absolute error

| Filename | Yaw Error | Pitch Error | Abs YE | Abs PE |
|---|---|---|---|---|
| AFW_134212_1_0,mat | -0,9 | -2,7 | 0,9 | 2,7 |
| AFW_134212_2_0,mat | 15,3 | -11,3 | 15,3 | 11,3 |
| AFW_1130084326_2_0,mat | 1,3 | -5,7 | 1,3 | 5,7 |
| AFW_261068_1_0,mat | 12,2 | -4 | 12,2 | 4 |
| AFW_261068_2_0,mat | -3,7 | 2,2 | 3,7 | 2,2 |
| AFW_2353849_1_0,mat | -0,2 | -1,8 | 0,2 | 1,8 |
| AFW_3989161_1_0,mat | -6,2 | -0,4 | 6,2 | 0,4 |
| AFW_16413031_1_0,mat | 5,9 | -4,2 | 5,9 | 4,2 |
| AFW_18489332_1_0,mat | 10,5 | -0,9 | 10,5 | 0,9 |
| AFW_22699817_1_0,mat | 0,4 | -6,1 | 0,4 | 6,1 |
| AFW_24795717_2_0,mat | 1,9 | -8,6 | 1,9 | 8,6 |
| AFW_24795717_3_0,mat | 4,4 | -15,6 | 4,4 | 15,6 |
| AFW_156474078_1_0,mat | -5,7 | 16,8 | 5,7 | 16,8 |
| AFW_170817766_2_0,mat | -1,5 | 1,9 | 1,5 | 1,9 |
| AFW_199759840_1_0,mat | -1,3 | -8 | 1,3 | 8 |
| AFW_213654866_1_0,mat | 3,4 | -6,3 | 3,4 | 6,3 |
| AFW_237815567_4_0,mat | -2,2 | -11 | 2,2 | 11 |
| AFW_255360810_1_0,mat | 2,9 | -4,7 | 2,9 | 4,7 |
| AFW_270814102_1_0,mat | -1,3 | -8,2 | 1,3 | 8,2 |
| AFW_281972218_1_0,mat | 50,2 | 1,2 | 50,2 | 1,2 |
| AFW_342149506_1_0,mat | 5,8 | -0,1 | 5,8 | 0,1 |
| AFW_346731834_1_0,mat | -0,9 | 5,3 | 0,9 | 5,3 |
| AFW_1372188757_1_0,mat | 17,9 | -7,7 | 17,9 | 7,7 |
| AFW_815038_1_0,mat | 2,7 | 1,2 | 2,7 | 1,2 |
| AFW_5452623_1_0,mat | 5,5 | 3,2 | 5,5 | 3,2 |
| AFW_45092961_1_0,mat | 3,8 | -11,3 | 3,8 | 11,3 |
| AFW_45092961_2_0,mat | -0,8 | -4,9 | 0,8 | 4,9 |
| AFW_59354095_2_0,mat | -0,3 | -10,7 | 0,3 | 10,7 |
| AFW_442651885_1_0,mat | -3,7 | -2,6 | 3,7 | 2,6 |
| AFW_507521047_1_0,mat | -9 | -9,6 | 9 | 9,6 |
| AFW_719902933_1_0,mat | 0,6 | -3,3 | 0,6 | 3,3 |
| Error Average | | | **5,88** | **5,85** |

Figure 16: Second approach absolute error

### 5.3 Summary of the codes presented with the report

In this Annex the functionalities of the codes attached with this report are presented:

- ***Approach1_ATTRACTOR:*** Includes Approach 1 with some attractors defined, corresponding to objects of the kitchen shelves in the lab.
- ***Approach2_ATTRACTOR:*** Includes Approach 2 with some attractors defined, corresponding to objects of the kitchen shelves in the lab.
- ***Approach_1_2_SWITCH:*** Includes Approach 1 and Approach 2, enabling the user to choose which to use at any time by pressing *'a'*.
- ***Approach_1_2_FULL:*** Includes Approach 1 and Approach 2, both executed at the same time to see the different behaviours.
- ***Approach_1_2_IMPROVED:*** Includes the Approach 1 and Approach 2, but the former is used in the central region while the latter is used in the outer region.
- ***calibration*** [11]***:*** Used to calibrate the camera and obtain their intrinsic parameters.
- ***getImages*** [11]***:*** Used before the *calibration* code to obtain images of a checkerboard pattern with the camera.
- ***mouse_coordinates:*** Used to extract the mouse cooridnates to compute the homography.