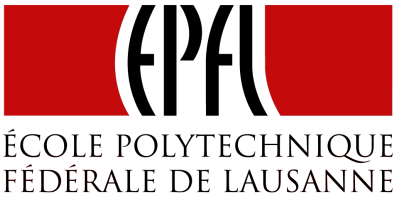
MT MA3 - Semester Project



Object localization using gaze tracking

Learning Algorithms and Systems Laboratory (LASA)

Valentin Morel

September 12, 2023

PhD Assistant:
Iason Batzianoulis

Professor:
Aude Billard

Contents

1 Introduction

Controlling a robot arm in an intuitive way is an interesting field of research. Using the gaze for this purpose could simplify the interaction human-robot. Indeed, the gaze could be used as a control input to increase the intuitiveness of the control system. It could identify the intention of the user and therefore facilitate the interaction with the robot and make it more user-friendly.

Moreover the use of the gaze as the input in a control system could be a good alternative for people who have limited functionality of the upper limb for example. It could occurred after an accident or a stroke for example. As explained in [1], the gaze is used as part of the control of an upper limb exoskeleton for rehabilitation in real-world tasks. Using a brain–computer interface (BCI) for the control of the speed of the exoskeleton arm, the gaze is tacked to identify the object of interest. This approach helped for the neurorehabilitation of the patients. It has shown a high potential if used since the earliest phase of recovery. Thanks to the gaze, the intuitiveness of the control system has been improved.

The purpose of this project was to detect the object of interest in order to send the coordinates of this object to a robot. Knowing these coordinates the robot could move to the object in order to grasp it. For this purpose, the user is wearing a headset to track the pupils to interpolate the gaze focal point of the candidate. Image processing, computer vision algorithm and machine learning regression are combined in order to achieve this task.

The first part focuses on the detection of the pupil which is critical to detect which object the user is looking at. The second part consists in mapping the tracking of the pupil with the gaze. For this non linear task, Support Vector Regression (SVR) is used. Detecting the objects and extracting their coordinates is part of the third part. In the last part, a relation between the coordinates in pixel and a reference frame is presented. All together, these different aspects achieved the goal of the project.

2 Apparatus

2.1 Headset description

The headset used for this experiment is designed by *Pupil Labs GmbH*[1]. It is composed of three cameras, as one can see on figure 2.1. As it weights only 34 grams, it is light to wear.

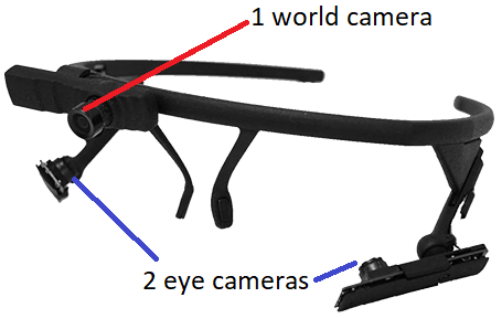


Figure 2.1 – Headset composed of three cameras used for the experiment

It is composed of one world camera with a possible resolution up to 1920x1080. For performance reasons, the resolution chosen for this camera is 1280x720 with a sampling rate of 60 Hz. This camera captures frames from the work world space.

The two eyes cameras are IR camera with IR illumination. 400x400 is the resolution chosen with a sampling rate at 60 Hz. They are used to track the pupils in order to define the position of the gaze focal point.

3 Method

All the different algorithms are performed with Python 2.7 and OpenCV 3.3.1 library. SVR is implemented using Scikit-Learn library.

3.1 Pupil tracking

In order to detect the focal point of the gaze, an accurate and precise pupil detection is essential. The developed algorithm is inspired by the one described in the following paper [2].

Three main steps are performed in order to track the pupil:

1. Approximate the pupil region using a simple feature detection in order to reduce the search space in the following steps.
2. Image processing on the delimited pupil region in order to improve the detection of the pupil.
3. Find the pupil using an ellipse fitting algorithm on the remaining processed region and then extract the center.

These three steps will be presented in the following subsections.

[1]Company website: https://pupil-labs.com/

3.1.1 Region estimation

It is assumed that the pupil region is the darkest blob surrounded by a light background. Therefore to find the region of interest (ROI) a Haar-like feature detecor is used.

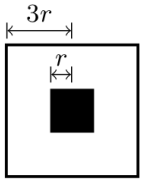
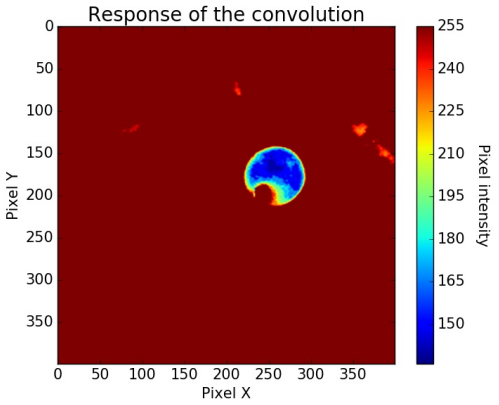


Figure 3.1 – Haar-like centre-surround feature

A Haar-like centre-surround feature is convoluted with the given frame, see figure 3.1. The one chosen is square, similar to the detector used in [2]. As it is a square Haar-like feature, it only approximates the location of the elliptical pupil, see figure 3.2b.

(a) Frame capture by the eye camera



(b) Result of the convolution

Figure 3.2 – Convolution with Haar-like feature. The figure 3.2b shows the result of the convolution. The 'blue' shape indicates the presence of the pupil and defined therefore the ROI, see figure 3.3

The highest response region is therefore define as the pupil region, as one can see on figure 3.3.

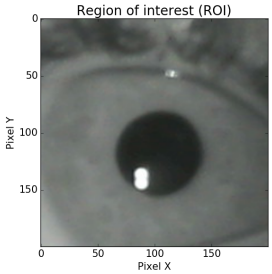
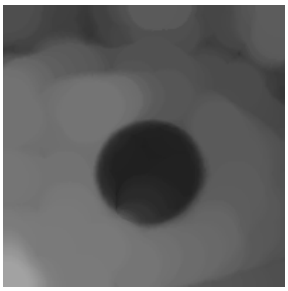


Figure 3.3 – Selected region of interest

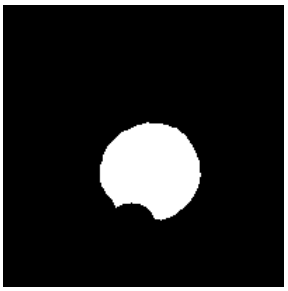
3.1.2 Image processing

After having selected the ROI of a given frame, different image processing techniques are applied on the image. First a morphological 'open' operation is performed in order to remove features like glint and eyelashes, see figure 3.4a. Assuming that the pupil is the darkest element in the image, an

intensity thresholding is applied. The value of the threshold is critical for the performance of the algorithm, see figure 3.4b. This value is typically set around 50 but depends on the luminosity of the environment. It has to be changed manually.



(a) Opening morphological operation



(b) Image thresholding

Figure 3.4 – Results of image processing techniques applied on figure 3.3

3.1.3 Ellipse fitting

The final step consists in fitting an ellipse in the thresholded image. For this purpose, the contours are extracted. It can be more than one if the value of the threshold is not perfect. To fit an ellipse, one need at least five points extracted from the contour. The circularity, see equation (1), and the area of each contour is calculated. The contour with the highest circularity and a valid area value is assumed to be the pupil. An ellipse is then fitted on this contour and the center of the ellipse is extracted and considered to be the center of the pupil, see figure 3.5.

By definition, the *circularity* is a metric that calculates how close is a shape from a perfect circle, as it is explained in [3]. It is calculated as follows:

$$\text{Circularity} = \frac{4\pi A}{P^2} \tag{1}$$

- P is the perimeter of the shape
- A is the area of the shape

The circularity of a perfect circle is 1. It is smaller for shapes that are less circular.

Figure 3.5 – Ellipse fitting and center extraction. The yellow dot indicates the center of the pupil found. The yellow ellipse indicates the shape of the pupil detected.

3.1.4 Pupil algorithm detection

The algorithm 1 presents the pseudo-code of the three different steps explained in the previous subsections. It has two inputs. The first one is a frame captured from one of the two eye cameras. This image is processed in order to detect the pupil. The second one is the chosen threshold value. This value is used to create a binary image from the processed input frame. The function returns the center of the detected pupil and the image with the drawn ellipse and center. The block diagram of the algorithm can be found in the annex, see A.1.

Algorithm 1 Pupil center detection algorithm

procedure PUPIL-CENTER($capturedFrame, threshold$)
  $rgbImage \leftarrow capturedFrame$
  $rgbImageROI \leftarrow$ convolution of Haar-like feature and $rgbImage$ ▷ Define the ROI
  $grayImage \leftarrow$ grayscale($rgbImageROI$)
  $openingImage \leftarrow$ OpenMorphological($grayImage$) ▷ Remove glint and eyelashes
  $ThresholdImage \leftarrow$ Threshold($grayImage$) with $threshold$ value
  $contours \leftarrow$ findContours($ThresholdImage$)
  for c in $contours$ **do**
    Calculate $currentcircularity$
    Calculate $currentArea$
    if $currentCircularity > oldCircularity$ **then**
      $pupilContour \leftarrow c$
      $oldCircularity \leftarrow currentCircularity$
      $Area \leftarrow currentArea$
  if length($pupilContour$) > 4 **then** ▷ 5 point at least to fit an ellipse
    if ($oldCircularity > 0.55$) And ($200 < Area < 6000$) **then**
      $MyEllipse \leftarrow$ fitEllipse($pupilContour$)
      $nbrEllipse \mathrel{+}= 1$
  if $nbrEllipse == 1$ **then**
    Extract center (cx, cy) from $MyEllipse$ ▷ Center of the pupil
    Draw pupil contour on $rgbImage$ with its center
    return $(cx, cy, rgbImage)$
  else
    return No pupil center found

3.2 Gaze tracking

In order to track the gaze of the user in the world space, a non linear regressor machine learning algorithm has to be trained. For this purpose, data are recorded as follow for each different frame:

- The center of both pupils are extracted while the user is focusing his gaze on an easy recognizable pattern, called ArUco Marker, fixed on a ruler, see figure 3.6.
- The coordinates of the ArUco Marker are saved with synchronization of the pupils detection.
- The ArUco Marker has to explore the whole space.
- The frame is discarded if at least one of the information above is missing.

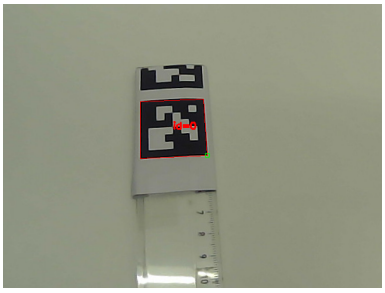


Figure 3.6 – ArUco marker used to explore the whole space on order to record data to train a regression.

The dataset used looks like the following:

ID	cx_left	cy_left	cx_right	cy_right	cx_world	cy_world
2	204	210	146	226	624	471
9	205	213	148	223	623	483
10	205	214	146	224	622	485

Table 1 – Example of data saved. All value are in pixel unit

Where:

- **cx_left** and **cy_left** are the X and Y coordinates of the left pupil center.
- **cx_right** and **cy_right** are the X and Y coordinates of the right pupil center.
- **cx_world** and **cy_world** are the X and Y coordinates of the marker, see figure 3.6

The regressor is trained twice, once for each world coordinate. Input data for both coordinate are: cx_left, cy_left, cx_right, cy_right. The target is once cx_world and once cy_world in order to determine the focal point of the gaze. The model has to be trained for each different user.

3.2.1 Support Vector Regression (SVR)

Support Vector Regression has been chosen to perform the mapping of the gaze. It is an effective tool for the nonlinear function approximation without the assumption of a prior parametric model.

As it is presented in [4], this machine learning algorithm has shown some accurate results for this task. A Gaussian RBF kernel is chosen due to its high performance in similar tasks [5]. A gridsearch is performed on the hyperparameters C and ϵ for each new dataset to train an ϵ-SVR with 5-fold cross validation.

3.3 Objects detection

This algorithm works by binarization of a grayscale image as explained below. Therefore, a high contrast is needed between the objects and the background. The setup is critical. The background of the objects has to be as white as possible. It allows a detection of dark and coloured objects with no major bright features.

The object detection algorithm developed during this project is based on image processing techniques, as explained in [6]. First the RGB captured frame is converted into a grayscale image. Then a median filter is applied in order to remove part of the noise and to smooth the image. It results in a frame with less small bright features, as one can see on figure 3.7b. Then an intensity thresholding is applied and a morphological 'open' operation is performed. The value of the threshold is a free parameter and can be tuned regarding the luminosity of the environment and the brightness of the object support.

The 2D centroid of the object shape is calculated. It is the weighted average of all the pixels constituting the shape extracted from the contour. For this purpose, one need to calculate moments, see equation (2). Image moment is a particular weighted average of image pixel intensities. It helps to find different properties of an image such as centroid, radius, area. This moment, sometimes called "raw moment", is calculated as follow:

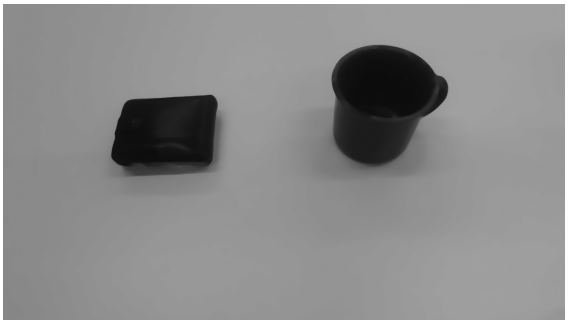
$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y) \tag{2}$$

For a grayscale image, $I(x, y)$ is the pixel intensity. The centroid is then given by this formula:

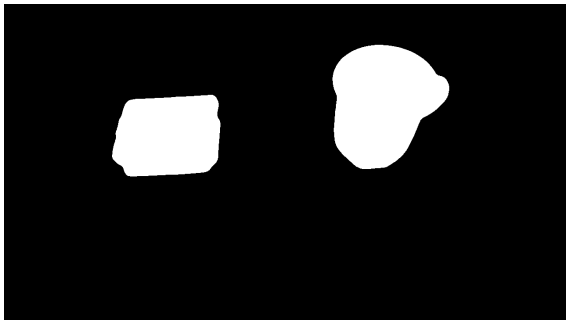
$$\begin{aligned} C_x &= \frac{M_{10}}{M_{00}} \\ C_y &= \frac{M_{01}}{M_{00}} \end{aligned} \tag{3}$$

C_x is the x coordinate and C_y is the y coordinate of the centroid. The result of all these different operations can be seen below:

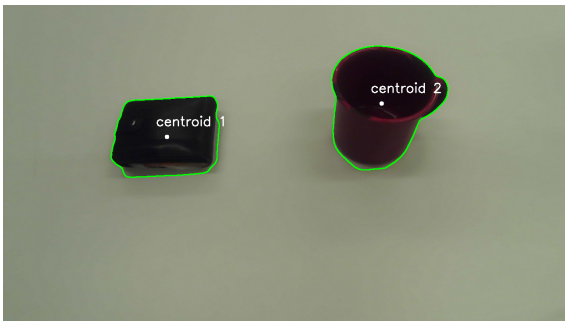
(a) Grayscale image



(b) Median blurring operation



(c) Opening morphological operation on the binarized image



(d) Contour and centroid of the objects

Figure 3.7 – Image processing techniques for object detection

3.3.1 Object detection algorithm

The pseudo-code of the object detection algorithm is presented below. It takes two parameters as inputs. The first one is a frame captured from the world camera. The second one is the chosen threshold value. This value is used to create a binary image from the processed input frame. The function returns the centroid of the detected object and the image with the drawn contours. The block diagram of the algorithm can be found in the annex, see A.2.

Algorithm 2 Object detection algorithm

```
procedure PUPIL-CENTER(myCap, threshold)
    rgbImage ← myCap
    grayImage ← grayscale(rgbImageROI)
    MedianBlurred ← medianBlur(grayImage)
    ThresholdImage ← Threshold(MedianBlurred) with threshold value
    openingImage ← OpenMorphological(ThresholdImage)
    contours ← findContours(ThresholdImage)
    for c in contours do
        Calculate Area
        if 2000 > Area > 70000 then
            Calculate c moments
            Calculate centroïd (cx, cy) based on moments
            Draw contour on rgbImage with its centroid
    return (cx, cy, rgbImage)
```

3.4 Localization of the object regarding a global frame

Extracting localization coordinates of the objects is essential for a robotic application. The transformation between the camera coordinate and the world coordinate are based on pinhole camera model, see figure 3.8. The scene view is formed by projecting 3D points into the image plane using a perspective transformation as explained in [7]. The relation is the following:

$$s\mathbf{m}' = \mathbf{A}[\mathbf{R}|\mathbf{t}]\mathbf{M}' \tag{4}$$

or

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5}$$

where:

- (X, Y, Z) are the coordinates of a 3D point in the world coordinate space
- (u, v) are the coordinates of the projection point in pixels
- $\mathbf{A}$ is a camera matrix, or a matrix of intrinsic parameters
- (cx, cy) is a principal point that is usually at the image center
- fx, fy are the focal lengths expressed in pixel units.
- s is the scalling factor
- $[\mathbf{R}|\mathbf{t}]$ is a matrix of extrinsic parameters. It describes the camera motion around a static scene.

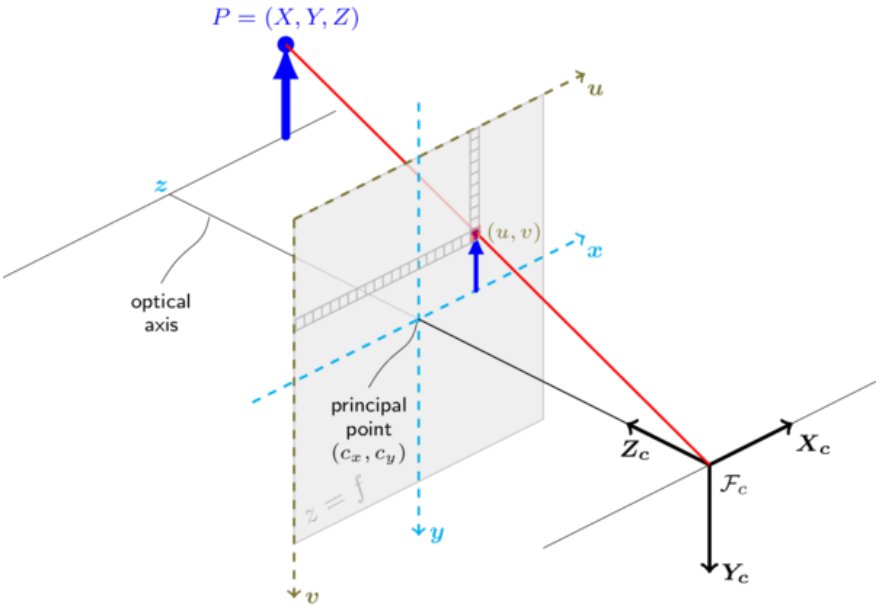


Figure 3.8 – Pinhole camera model. P is the 3D point in the world coordinate space. (u, v) are the coordinates of the projection point in pixels. Source: [7]

3.4.1 Matrix of intrinsic parameters A and distortion coefficients

The world camera used for this project includes a lens. It results a radial distortion and a small tangential distortion even after the calibration, small distortions are still present. Therefore, the distortion coefficients has to be found and the camera matrix A needs to be determined. The distortion coefficient are given by:

$$Distortion\ \ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3) \tag{6}$$

where:

- k_1, k_2, k_3 are radial distortion coefficients
- p_1 and p_2 are tangential distortion coefficients

These coefficients will be used to calculate the matrix of extrinsic parameters [R|t]

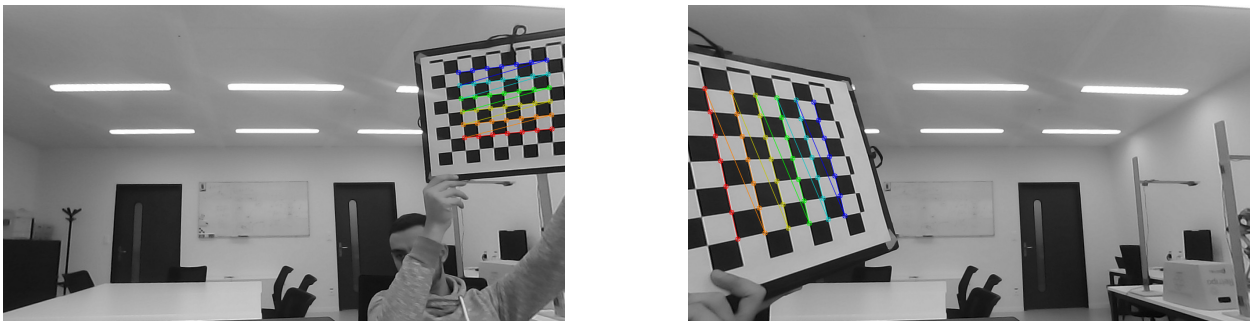
To find these coefficients and the camera matrix, a chessboard was used because it is a well defined pattern. It is easy to find square corners, as one can see on figure 3.9. The parameters of the chessboard are defined as they are known. The coordinates in the 3D real world space and coordinates in the 2D image can be extracted by an OpenCV function. Then a function estimates the intrinsic camera parameters and extrinsic parameters for each of the views. The algorithm performs the following steps:[2].

- Compute the initial intrinsic parameters.
- Estimate the initial camera pose using a function that can find the correspondences from 3D-2D points.

[2]Description of the function *calibrateCamera*: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html

- Run the global Levenberg-Marquardt optimization algorithm[3] to minimize the reprojection error. This is the total sum of squared distances between the observed feature points and the projected object points (using the current estimates for camera parameters and the poses).

To improve the results, this matrix and these coefficients have to be calculated from several different images taken from the whole work space, as explained in [8].



(a) Chessboard detection on the right *(b) Chessboard detection on the left*

Figure 3.9 – Detection of the chessboard corners from different position

3.4.2 Matrix of extrinsic parameters [R|t]

In opposition to the matrix of intrinsic parameters A, the matrix of extrinsic parameters [R|t] is different for each frame because the head of the user is not fix and therefore neither is the world camera.

For this purpose, others OpenCV functionality are used. The first one consists in detecting an ArUco Board. It acts like a single marker and provides a single pose for the camera. The advantage of using a board instead of a set of independent marker is that the relative position between the markers in the board is known a priori. Thus, the corners of all markers can be used to estimate the pose of the camera. The results of the detection can be seen below:

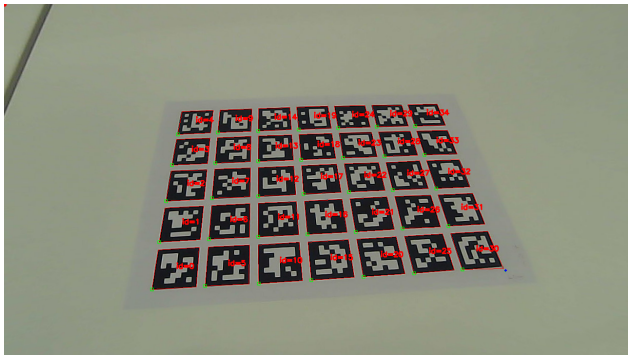


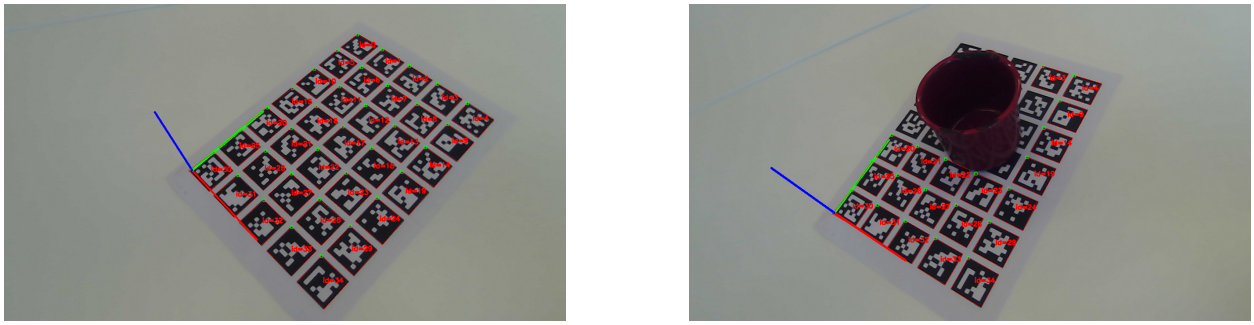
Figure 3.10 – ArUco Board detection

The second functionality of OpenCV used consists in the estimation of the board position. For this purpose, the parameters of the known board, the number of marker detected, the corners detected, the matrix of intrinsic parameters A and the distortion coefficients are given to a function. It returns the pose of a marker board composed by those markers. As the ArUco Board layout is known, it has a single world coordinate system. The returned transformation is the one that transforms points

[3]Detail of the Levenberg-Marquardt algorithm: https://en.wikipedia.org/wiki/Levenberg%E2%80%93Marquardt_algorithm

from the board coordinate system to the camera coordinate system. The result is the matrix of extrinsic parameters [R|t]. More details of the functionalities are available in [9].

Given the matrix of extrinsic parameters [R|t], the matrix of intrinsic parameters A and the distortion coefficients, a frame can be define with an other OpenCV function, as one can see on figure 3.11. It is not necessary to detect all the markers on the board but the more are detected, the more precise is the definition of the frame, see figure 3.11b.



(a) All markers detected *(b) Not all markers detected*

Figure 3.11 – ArUco Board detection with axis defining a frame. The frame is defined even if not all markers are detected, see (b). Red axis: X axis. Green axis: Y axis. Blue axis: Z axis.

3.4.3 Utilization of the defined frame

As the objects are always on the same reference plane, the Z coordinate is equal to 0. So the equation (5) can be simplified as follows:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A}[\mathbf{R}|\mathbf{t}] = \mathbf{A} \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2\mathbf{R}_3\mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{R}_1\mathbf{R}_2\mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{7}$$

where:

- $\mathbf{R}_1\mathbf{R}_2\mathbf{R}_3$ are the column of the rotation matrix $\mathbf{R}$
- $\mathbf{t}$ is the translation vector
- $\mathbf{H}$ is the homography between the reference surface and the image plane

$\mathbf{H}$ is a 3x3 matrix so it is easily invertible[4]. (u, v) are coordinates of the object detected as explained in the part 3.3. So the coordinates in the world frame can be expressed as follow:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = s\mathbf{H}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{8}$$

The coordinates are calculated in real-time for each frame captures by the world camera and send to a robot through a ROS[5] topic node.

[4] If the determinant of $\mathbf{H} = 0$, the matrix is singular and cannot be invertible
[5] Robot Operating System

3.5 Object localization using gaze tracking

The entire program combines all the subsections above in order to reach the goal of the project. In the first part, the pupils are detected and the coordinates of their centers are used to find the focal point of the gaze using the trained SVR model. In the second part, the object are detected and their centroid are calculated. If the gaze of the user focuses on one of the detected object, its middle bottom edge coordinates are converted into the world coordinates thanks to the ArUco board, see figure 3.12. These world coordinates are then communicated to the robot through a ROS topic node. The block diagram of the program can be found in annex, see A.3.

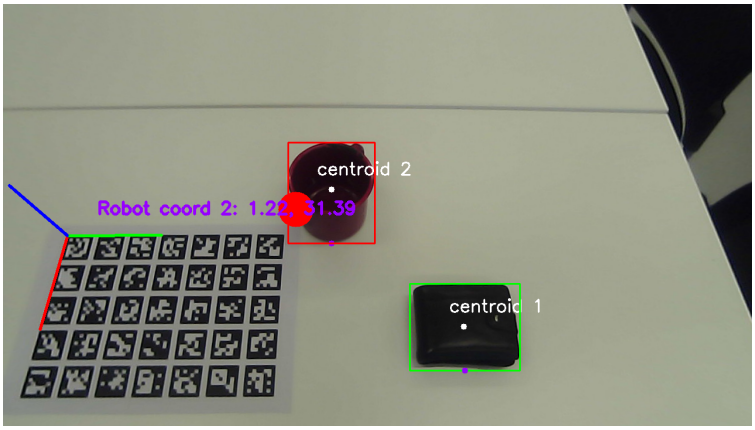
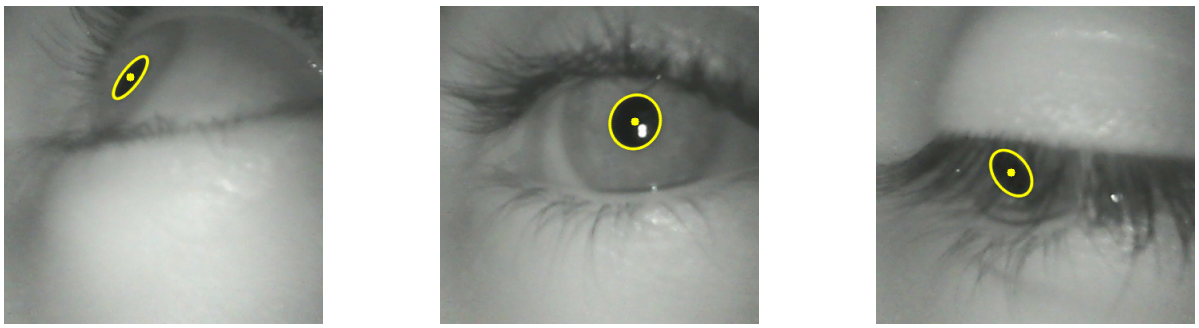


Figure 3.12 – Detection of the red cup. Robot coordinates are in cm with respect of the frame defined by the ArUco Board. Red axis: X axis. Green axis: Y axis. Blue axis: Z axis. The rectangle around the detected object change the colour to indicate which object could be grasp by the robot arm.

4 Results

4.1 Pupil tracking

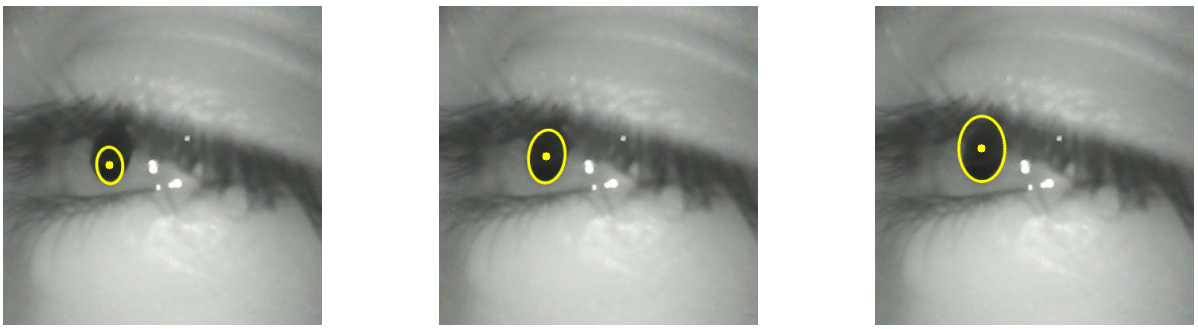
As one can see on figure 4.1, the algorithm gives good results in almost all situations. Even if the eyelashes cover most of the pupil, the extraction of the center of the pupil is accurate, see figure 4.1c.



(a) Pupil in the corner *(b) Glint in the pupil* *(c) Eyelashes on the pupil*

Figure 4.1 – Different results of the algorithm on difficult images

The principal factor that has an influence on the performance is the value of the threshold. This parameter has to be tuned regarding the luminosity of the environment. As one can see on figure 4.2, this value impacts the accuracy of the pupil detection.



(a) Threshold value: 40 *(b) Threshold value: 50* *(c) Threshold value: 60*

Figure 4.2 – Different results of the algorithm on the same image with different value of threshold. Depending on this value, the accuracy of the detection changes.

4.2 Gaze tracking

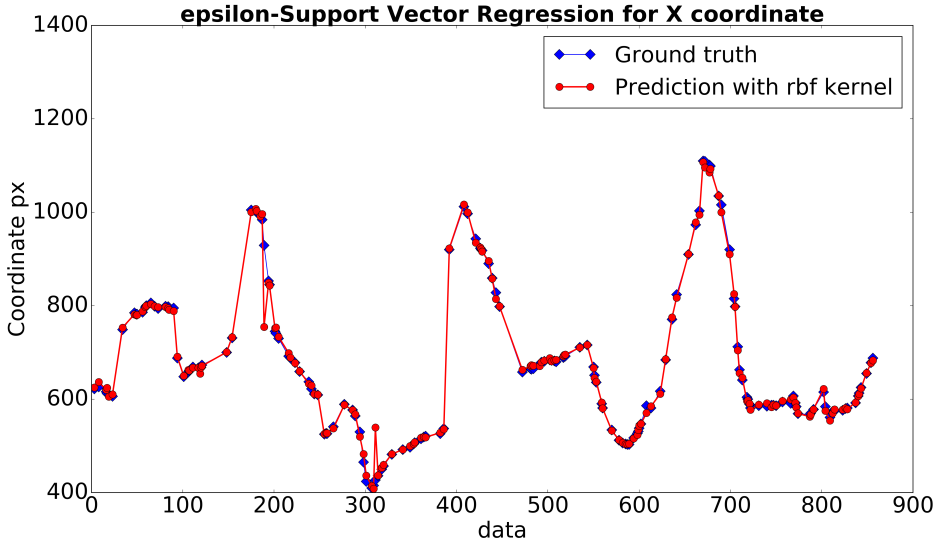
The dataset analyzed is composed of 863 different data-points. It is recorded from one unique user. 20% of the data are used for the test set and 80% for the training set. The dataset is randomly separated. A gridsearch on the train set is performed on selected hyperparameters with 5-folds cross validation in order to maximize the R^2 score metric.

4.2.1 Support Vector Regression (SVR)

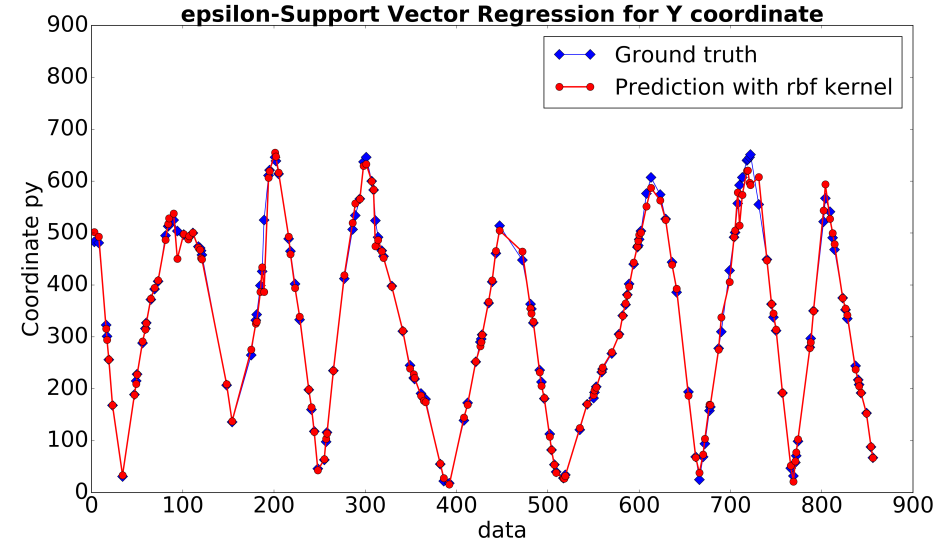
A ϵ-Support Vector Regression with Gaussian RBF kernel has been trained with the given dataset. A gridsearch on the hyper parameters C and ϵ has been done. The following results have been obtained:

Coordinate	C	ϵ	R^2 [%]	Mean absolute error [pixel]	Computation time [s]
X	1100	0.67	98.90	5.30	44
Y	1700	0.7	99.07	8.37	

Table 2 – Result for ϵ-Support Vector Regression with a gridsearch on 5 different values for each hyperparameter. The computation time was obtained with a processor Intel Core i7-7600U CPU @ 2.80GHz.



(a) X coordinate



(b) Y coordinate

Figure 4.3 – Regression prediction on the test set for both coordinates. The coordinates are in pixel unit. px: coordinate pixel in X. py: coordinate pixel in Y. Both coordinates prediction follow accurately the ground truth.

As one can see on figure 4.3, the trained models for both directions give accurate results. One can notice in table 2 that even if R^2 is higher for the Y coordinate, its mean absolute error (MAE) is bigger than for the X coordinate. However, the overall performance are accurate.

4.3 Object detection

One can observe on the figure 4.4 that a white object is not detectable on a withe background. The reason is that the algorithm works with binarization of a grayscale image. So it needs a high contrast to detect the edges of the objects. Therefore the yellow object is not perfectly detected as the contrast is not high enough.

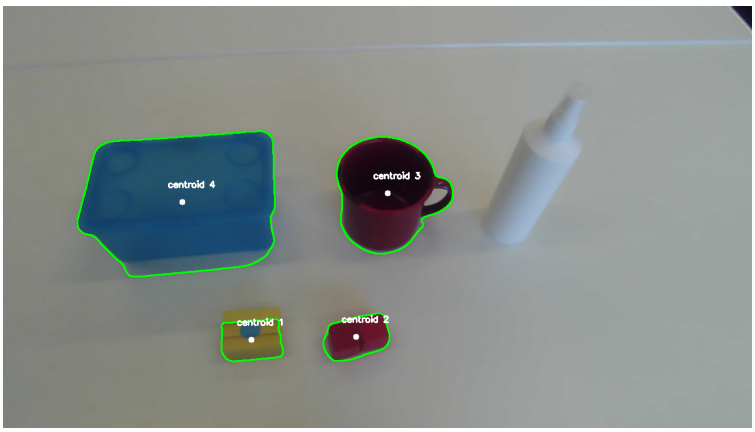


Figure 4.4 – Detection of objects with several colours on a white background

On the figure 4.5, the background is black. Therefore the white bottle is this time detected. The yellow object is better detected than on the white background. On the other hand, the red object is no more detected. It is considered as part of the background.

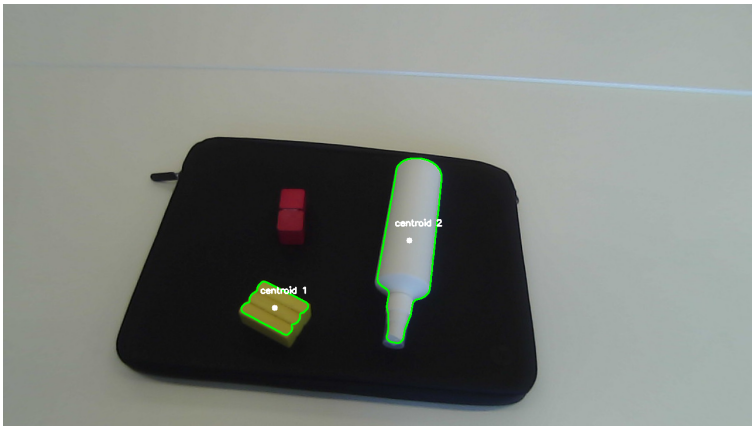


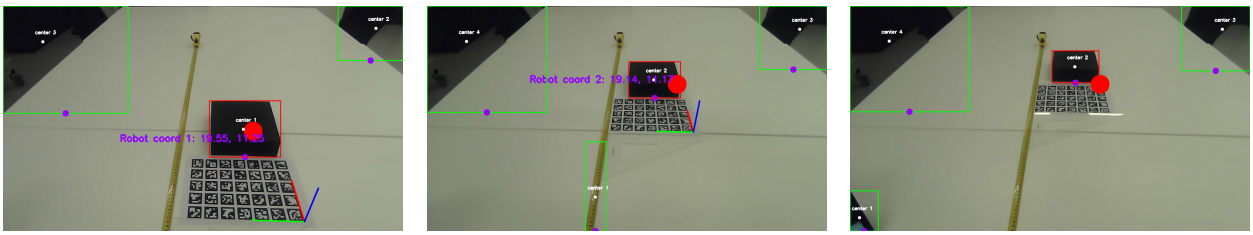
Figure 4.5 – Detection of objects with several colours on a black background

4.4 Localization of the object regarding a global frame

Several scenarios were tested in order to find the best setup to localize the objects as accurately as possible. For each scenario, the user was placed at a fix position. For each different scenario, 10 values were recorded in order to calculate the mean and the standard deviation.

4.4.1 Different positions for the ArUco Board

In this experiment, the goal was to define after which distance from the user, the utilization of the ArUco Board is no more possible to define the global frame. The object we want to detect has been placed at the same distance from the board for each different position (X: 19.65 cm | Y: 11.25 cm).



(a) Distance: 30 cm *(b) Distance: 60 cm* *(c) Distance: 70 cm*

Figure 4.6 – Different positions of the ArUco Board from the user. Red axis: X coordinate. Green axis: Y coordinate.

As one can see on figure 4.6c, if the ArUco Board is placed further than 60 cm from the user, the board is difficulty or no more detectable.

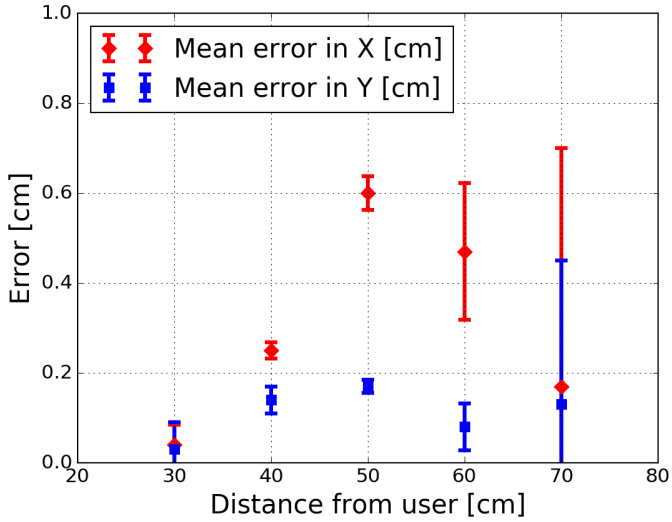


Figure 4.7 – Mean error of the object position in X and Y

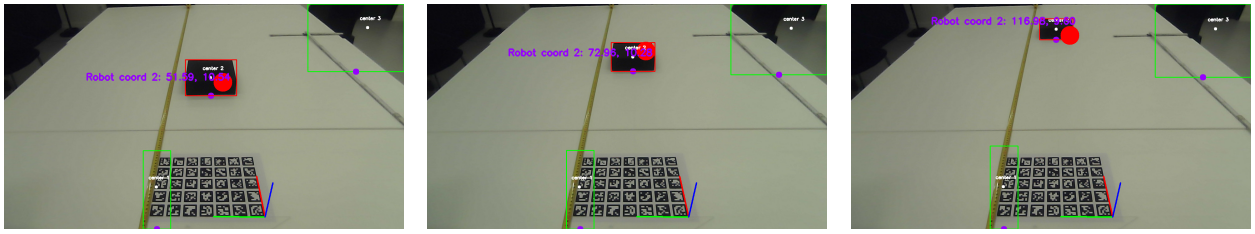
Distance of the ArUco board from the user [cm]	Accuracy in X	Accuracy in Y	Precision in X [cm^{-1}]	Precision in Y [cm^{-1}]
30	99.80%	99.73%	22.22	16.67
40	98.73%	98.76%	55.55	33.33
50	96.95%	98.49%	26.31	66.67
60	97.61%	99.29%	6.57	19.23
70	99.21%	98.84%	1.88	3.13

Table 3 – Accuracy and precision for the X and Y coordinate for different positions of the ArUco Board from the user

The figure 4.7 shows that for an object placed at a given distance from the board, the mean of the error of both coordinate is quiet stable, even if the board is placed far away the user. Furthermore the accuracy in both direction is high whatever the distance is. However, the standard deviation increases the further the board is placed from the user. Therefore the precision decreases, see table 3. For this reason, the board was placed at 30 cm from the user for the following experiments.

4.4.2 Different X position for the object

The goal in this experiment is to determine how the errors in the coordinates change regarding the position of the object for a fixed position of the board. The board was placed at 30 cm from the user.



(a) X coordinate: 50 cm *(b) X coordinate: 70 cm* *(c) X coordinate: 110 cm*

Figure 4.8 – Different positions of the object in the X direction. ArUco Board at 30 cm from the user. Red axis: X coordinate. Green axis: Y coordinate.

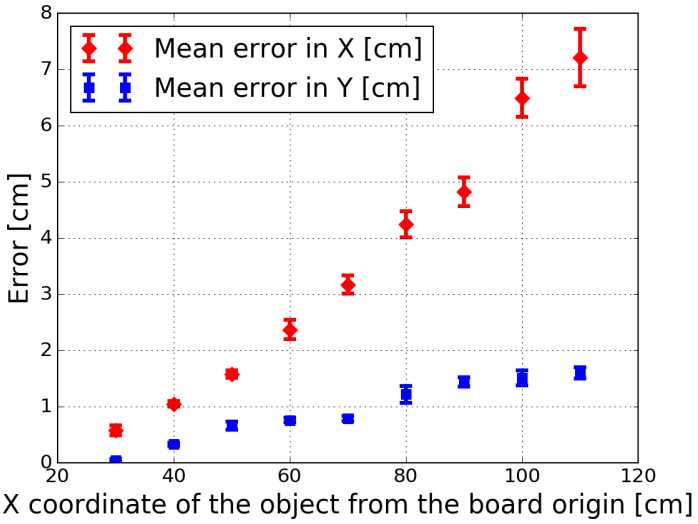


Figure 4.9 – Mean error of the object position in X and Y for different X coordinate

Different positions of the object in the X direction	Accuracy in X	Accuracy in Y	Precison in X [cm^{-1}]	Precison in Y [cm^{-1}]
30	98.07%	99.64%	11.11	33.33
40	97.38%	97.07%	22.22	34.48
50	96.84%	94.13%	14.71	13.70
60	96.05%	93.24%	5.88	25.00
70	95.47%	92.28%	6.25	20.83
80	94.70%	89.16%	4.29	6.76
90	94.64%	87.20%	3.88	11.49
100	93.51%	86.58%	2.96	7.52
110	93.45%	85.78%	1.97	10.10

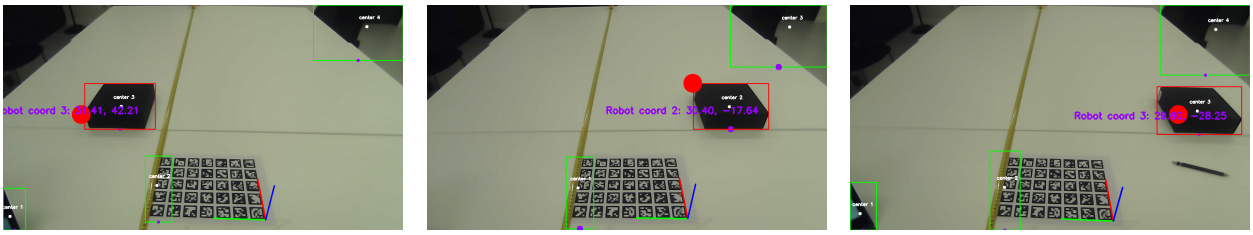
Table 4 – Accuracy and precision for the X and Y coordinate for different positions of the object in the X direction from the ArUco Board

As one can see on figure 4.9, the error in both directions increases the further the object is placed from the board. However, the mean error of the Y coordinate does not increase much and stays

below 2 cm. On the other hand, the mean error of the X coordinate tends to increase linearly and become higher than 5 cm if the object is placed further than 100 cm from the origin of the board. The standard deviation of the X coordinate also increases while it stays stable for the Y coordinate. Therefore the precision in the X direction decreases more than in the Y direction, see table 4.

4.4.3 Different Y position for the object

The object was placed at 30 cm from the origin of the board in the X direction. The position in the Y direction has been changed in order to calculate the error in the object position.



(a) X coordinate: 50 cm *(b) X coordinate: 70 cm* *(c) X coordinate: 110 cm*

Figure 4.10 – Different positions of the object in the X direction. ArUco Board at 30 cm from the user. Red axis: X coordinate. Green axis: Y coordinate.

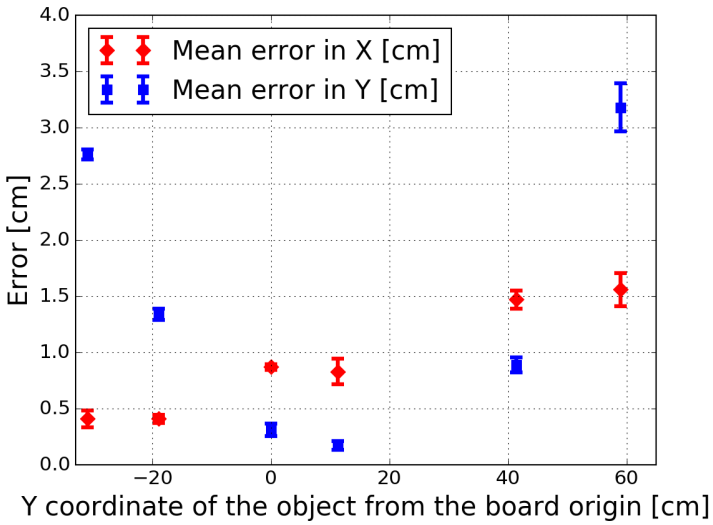


Figure 4.11 – Mean error of the object position in X and Y for different Y coordinate

Different positions of the object in the Y direction	Accuracy in X	Accuracy in Y	Precision in X [cm^{-1}]	Precision in Y [cm^{-1}]
-31	98.63%	91.10%	13.16	22.73
-19	98.63%	92.95%	27.03	20.41
0	97.10%	98.97%	41.67	17.54
11.25	97.23%	98.49%	8.70	25.64
41.4	95.10%	97.85%	12.35	14.71
59	94.80%	94.61%	6.76	4.65

Table 5 – Accuracy and precision for the X and Y coordinate for different positions of the object in the Y direction from the ArUco Board

As one can see on figure 4.11, the mean error in Y increases when the object is placed in the edges of the work-space. Therefore, the accuracy in the Y direction decreases when the object is placed in the edges, see table 5. However, the mean error for both coordinates stay below 3.5 cm.

5 Discussion

5.1 Pupil tracking

The accuracy of the pupil tracking part is high even in presence of glints and eyelashes. The chosen value of the threshold has a big influence on the performance. An automatic threshold calculation system could improve the performance of the algorithm as explained in reference [2]. As this algorithm works with a 2D ellipse model, it could be improved with a 3D model to fit the pupil in order to better detect the center of the pupil.

If the pupil of the user is in the corners of the image, the detection can be more difficult.

5.2 Gaze tracking

The use of SVR for the eye gaze mapping gave good results for experiment with a mean absolute error below 10 pixels in both coordinates.

The mapping could be improved with a blinking detection. As there is no blink detection, it can happen that some center pupil coordinates used to train the algorithms were wrong. Indeed, it is possible that even with the check on the area, the algorithm could detect a "fictive" pupil and therefore biased the dataset used to train the model. The same problem could occurred with the use of the SVR trained model. A false position of the gaze could be predicted if a pupil is detected during an eye blink. So improving this algorithm with a blink detection could enhanced the global performance of the gaze tracking.

As there is no gaze point when no pupil is detected, a low pass filter could be implemented in order to smooth the gaze detection and to reduce the noise.

5.3 Object detection

As the object detection algorithm works with image segmentation, it is sensible to the color of the objects and the color of the background. Therefore, it needs a high contrast between the object and the background. It is impossible to detect a light-colored object on a white table for example. Furthermore, the coordinates extract from the object are in 2D which could be problematic to use with a robot. For this reason, the coordinates send to the robot are the center of the bottom edge of the object. The Z coordinate is an offset which is a constant fixed for all objects.

A total different approach for this part could improve the detection. More advanced 3D models not sensible to the colors of the object should be considered to improve the global performance.

5.4 Object localization

The localization of the objects regarding the world reference frame is independent of the position of the head which is an advantage. For this purpose, the needed matrix is calculated in real time. The ArUco Board used for this purpose has the advantage that it can be placed anywhere on a flat surface. Then the coordinates calculated with this board can be easily projected on the axis defining the robot frame.

Regarding the different results obtained, the emplacement of the board should be as central as possible. Indeed, the error in the coordinates of the object increases with the distance of the object; the further away the object is, the larger the error becomes. The matrix of intrinsic parameters A calculated with the chessboard during the calibration is not totally correct. The calculation of the distortion coefficients is based on 50 images given to the algorithm. These images do not cover the whole field of view of the world camera and some regions of this field of view could have more images than other region. Therefore, the resulting coefficient could be biased. That could be an explanation of this increasing in the error. For an error below 5 cm, the origin of the board should not be placed further than 90 cm from all the objects one want to grasp with a robot arm.

6 Conclusion

The results obtained during this project have shown that the objective of simplifying the human-robot interaction using the gaze is reachable. The mapping of the gaze with SVR is performing good but needs to be trained each time the user is changed. The object detection is the part that could be improved the most in order to detect more kind of objects without regardless of their colours. The localization works well for every position of the head as long as the objects are not too far from the ArUco Board.

A Block diagrams

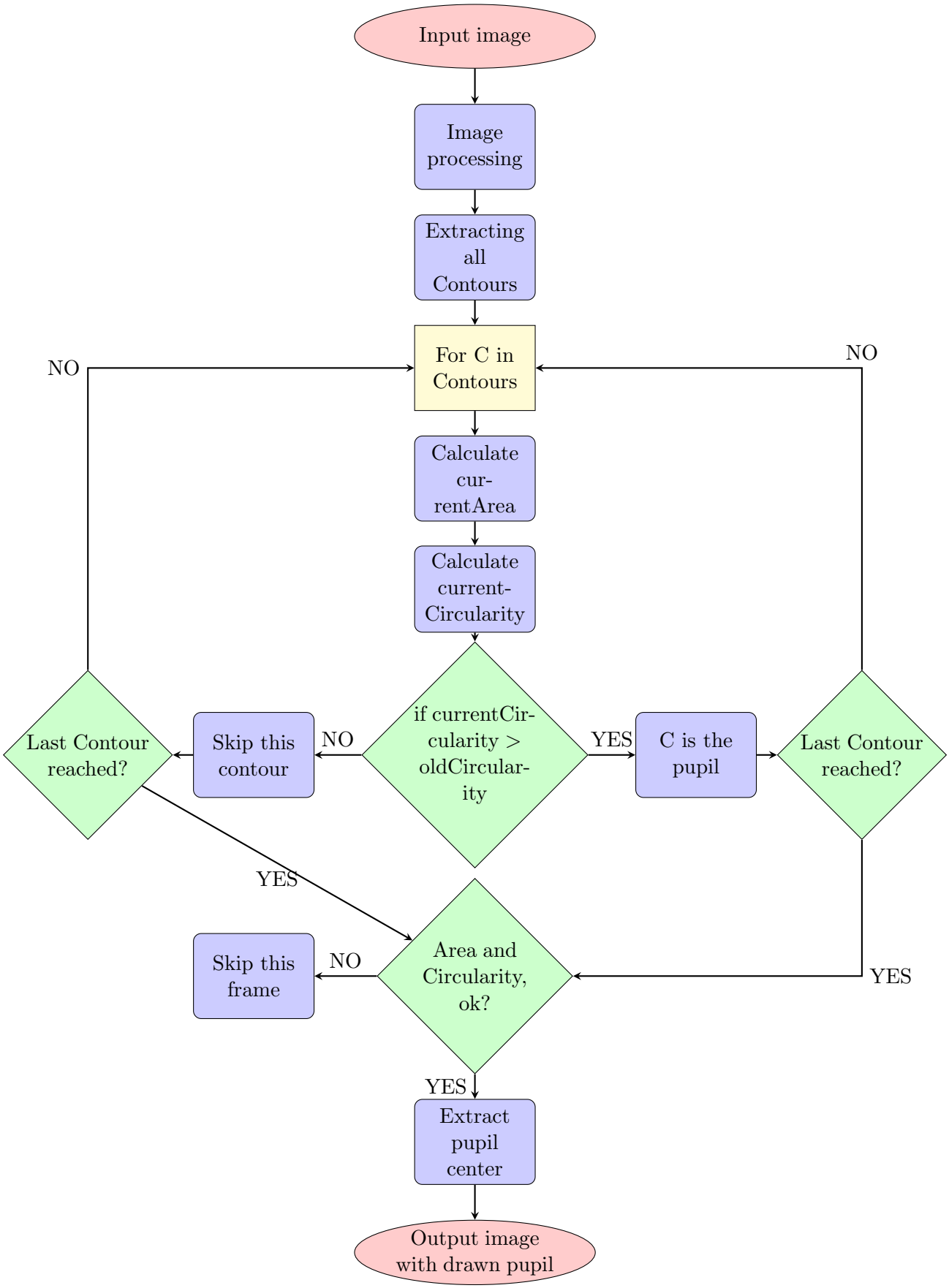


Figure A.1 – Pupil detection algorithm block diagram

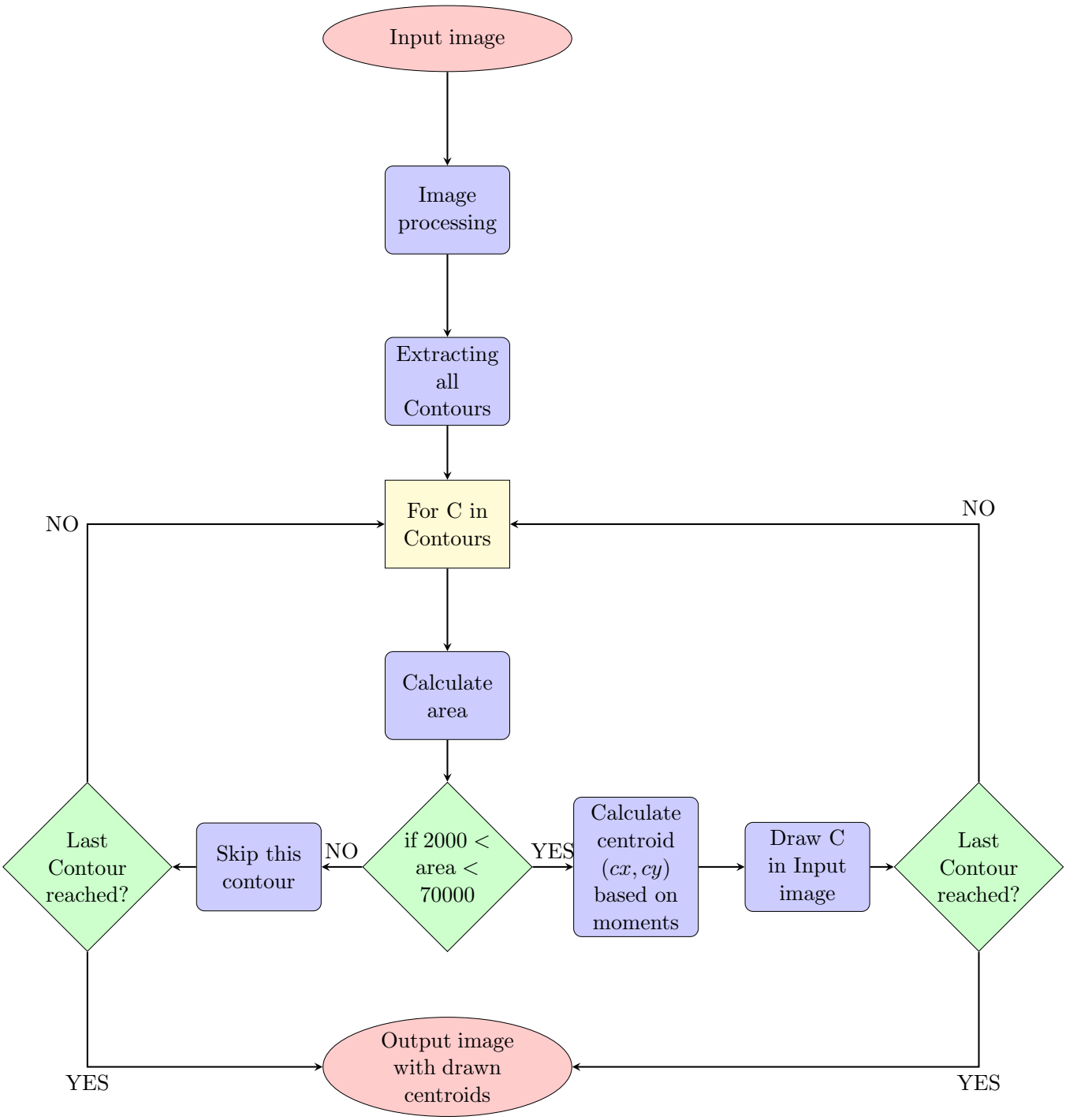


Figure A.2 – Object detection algorithm block diagram

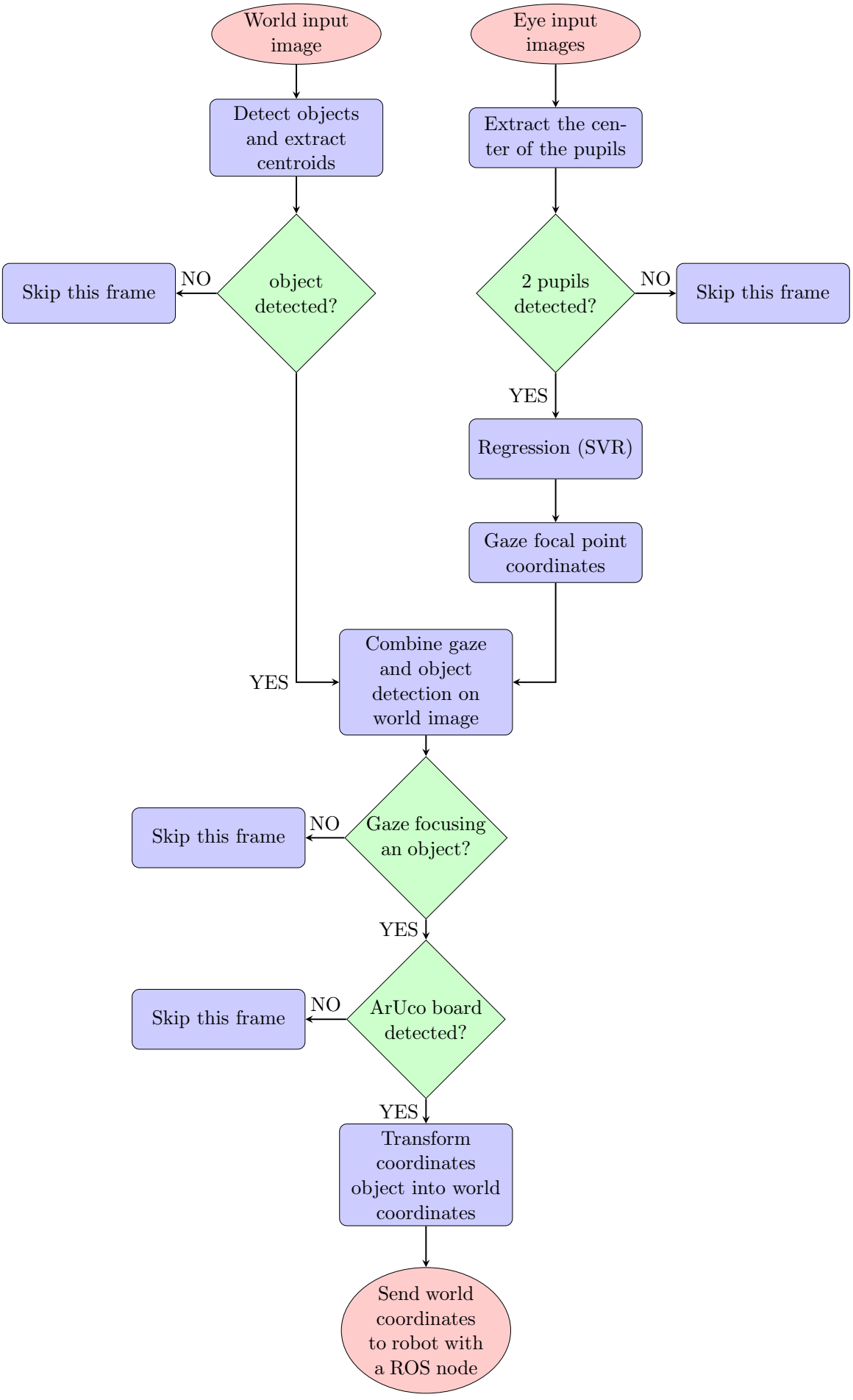


Figure A.3 – Entire program block diagram

References

[1] A. Frisoli, C. Loconsole, D. Leonardis, F. Banno, M. Barsotti, C. Chisari, and M. Bergamasco, "A New Gaze-BCI-Driven Control of an Upper Limb Exoskeleton for Rehabilitation in Real-World Tasks," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1169–1179, Nov. 2012. 3

[2] L. Świrski, A. Bulling, and N. Dodgson, "Robust real-time pupil tracking in highly off-axis images," in *Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA '12*. Santa Barbara, California: ACM Press, 2012, p. 173. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2168556.2168585 4, 5, 21

[3] "How to Calculate Circularity." [Online]. Available: https://sciencing.com/calculate-circularity-5138742.html 6

[4] B. Noris, J.-B. Keller, and A. Billard, "A wearable gaze tracking system for children in unconstrained environments," *Computer Vision and Image Understanding*, vol. 115, no. 4, pp. 476–486, Apr. 2011. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1077314210002493 8

[5] Zhiwei Zhu, Qiang Ji, and K. Bennett, "Nonlinear Eye Gaze Mapping Function Estimation via Support Vector Regression," in *18th International Conference on Pattern Recognition (ICPR'06)*. Hong Kong, China: IEEE, 2006, pp. 1132–1135. [Online]. Available: http://ieeexplore.ieee.org/document/1699089/ 8

[6] "Find Center of a Blob (Centroid) Using OpenCV (C++/Python) | Learn OpenCV." [Online]. Available: https://www.learnopencv.com/find-center-of-blob-centroid-using-opencv-cpp-python/ 8

[7] "Camera Calibration and 3d Reconstruction — OpenCV 2.4.13.7 documentation." [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html 10, 11

[8] "OpenCV: Camera Calibration." [Online]. Available: https://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html 12

[9] "OpenCV: Detection of ArUco Boards." [Online]. Available: https://docs.opencv.org/3.3.1/db/da9/tutorial_aruco_board_detection.html 13