POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Project Report

Delivery of a project in
Image Analysis and Computer Vision

Lorenzo GUERRIERI
Hugo MARINE
Lucas FOUREST

Academic Year: 2022-2023

Project context and global methodology

For this project, we have at our disposal a few custom table tennis videos, each involving a different gameplay sequence. We will select a specific one, in order to perform some Computer Vision key tasks on it (as tracking and reconstruction), knowing that we are limited to one point of view for each gameplay sequence filmed. Here we will choose to work on a video presenting a match between two players, shot from behind with a slight profile perspective, as you shown below:

Figure 1: Point of view

Initially, we will develop a method able to track both the racket in the player's hand and the ball. Then, as a logical continuation, we will try to recover the real positions of the ball, taking into account we only have one point of view, which prevents from any

triangulation possibility. One of the challenge of this part will be to find an alternative to triangulation that will allow us to overcome this limitation.
For the whole project, we will make use of *Python* and the *OpenCV* Computer Vision library mainly, as well as *NumPy*.

Contents

1 | Tracking moving objects

1.1. Racket tracking

A first obvious strategy to discriminate objects is to differentiate them by their colors. Here, the racket is apparent and its color clearly visible: this strategy seems reliable in this case.
However, it is worth noting that the BGR representation is not optimal for color-filtering operations, and that we should rather move into the HSV space, as stated by SIMSEK Gokhan in **[2]**.

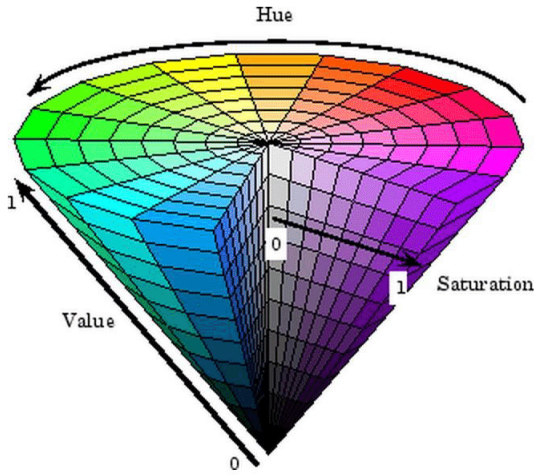


Figure 1.1: HSV space

However, it is hard to estimate which HSV representation (three dimensional vector as BGR) corresponds to a given color on first sight, thus we will rely on a purely computational method that will retrieve the HSV value of a pixel at the mouse pointer location.

This way we will be able to estimate the lower and upper HSV bounds to set as a filtering mask on the image. This color filtering mask, applied on the image, will discriminate the racket from the rest of the elements, and then a simple contour detection algorithm will be sufficient to finally compute a bounding box keeping track of the racket.

Figure 1.2: Racket tracking

1.2. Ball tracking

We must adopt a different method here. Indeed, we wish we could keep going the same way but the ball is small, moves fast and is subject to important color variations on the video due to changeing exposition and luminosity. The conditions are not satisified anymore to use such a color-based approach.
Therefore we will use here a background substraction approach to discriminate (fast) moving objects . It consists in computing the "mean frame" of the video taking into account the N last frames period, where N is a tunable parameter, as stated in the code. Then it computes a "mean pixel" over the N last frames for each coordinate, using a Gaussian Mixture Model (GMM). Then the mean frame is "removed". "Foreground" pixels (associated to moving objects) are then discriminated from "background" ones based on the stated absolute difference after substracting the mean frame, which is compared to a certain treshold T, again a tunable parameter. Here an overall view of what it does:

Figure 1.3: Foreground detection

This done, we are left with a binary (black and white) image which is still quite noisy, so we use morphological operators to perform "erosion" and "dilatation" in order to remove small irrelevant amounts pixels. More precisely, we apply a closing step (dilatation followed by erosion) followed by an opening step (erosion followed by dilatation).

Figure 1.4: After closing

Figure 1.5: After closing-opening

As explained in **[1]**, dilatation and erosion are morphological operations that both involve a structuring element of arbitrary shape B "sliding" on a binary ensemble (typically a

black and white image) X. Mathematically:

- Erosion of X by B noted $\epsilon_B(X)$ is defined as the ensemble of points $x \in X$ such that B centered in x (noted Bx) is completely included in X, meaning that every "1 pixel" in B_x meets a "1 pixel" on the image X. Formally:
$\epsilon_B(X) = X \ominus B = (x \in X | B_x \subset X)$

- Dilatation of X by B noted $\delta_B(X)$ is the opposite of erosion operation: it is defined as the ensemble of points $x \in X$ such that B_x has a non-empty intersection with X, meaning that at least "1 pixel" in B_x meets a "1 pixel" on the image X. Formally:
$\delta_B(X) = X \oplus B = (x \in X | B_x \cap X \neq \emptyset)$

Closing operation helps group areas that are close together into a single object. As such, there are some larger areas that are close to each other that should probably be joined before we do anything else. This can be a reason to perform closing first. Then an opening can be done after so that we can remove the isolated noisy areas. Usually closing structuring element size is larger as we want to make sure to get nearby pixels and the opening structuring element size is smaller so that we don't mistakenly remove any of the larger areas. Everything is visually summarized in the figure below:

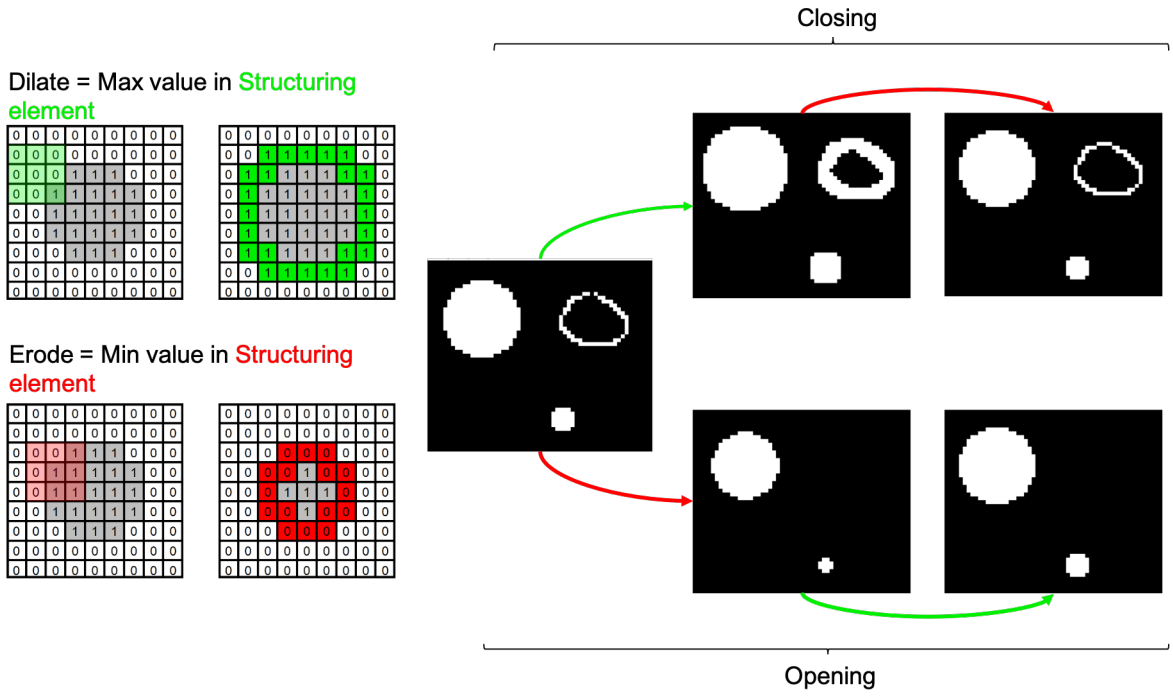


Figure 1.6: Morphological operators

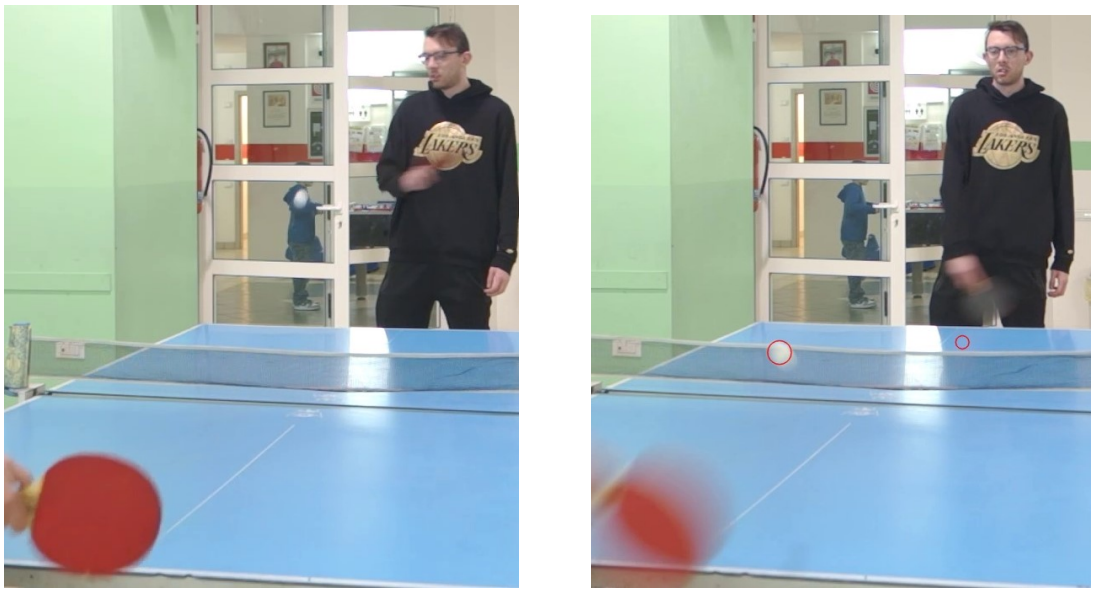
This whole closing-opening process is supposed to efficiently retain only the most relevant pixel blobs (amount of connected white pixels) and thus isolate the ball, as one of the very last blobs present on the black and white image.

Finally, we use the *openCV* blob detector with custom tunable parameters on the so-obtained image to capture (almost only) the ball pixels, and we draw the associated contour(s) on the original image.

Figure 1.7: Result

1.3. Results discussion

As stated in result videos, racket tracking here works really well, as stated in resulting videos, which is mainly due to the fact that racket as very specific color and is always clearly apparent on camera.
However, despite all the efforts put in denoising, ball tracking is still imperfect and either sometimes fails to detect the ball, either is perturbed by irrelevant detected blobs. This is the result of overall bad video conditions combined to a very small and fast moving ball, which sometime appears almost "invisible" on some frames. We should be aware of this weaknesses for the rest of the project, as the following parts rely on the quality of the input sequence retrieved by ball tracking.



(a) No detection (b) Noisy detections

Figure 1.8: Weaknesses

In addition to this sequence of $[x, y]$ points, we will also store the corresponding sequence of timestamps t, which will be needed later. From all that we would like to get back to $[X, Y, Z]$ real ball positions in time.

2 | Ball trajectory 3D reconstruction

2.1. Single point of view limitation: alternative method

Usually,to recover real positions of a moving object on a video, we need at least two different points of view of this object successive positions, and using these multiple sequence of points we can perform triangulation. Here, each video films a different gameplay sequence (we were limited to only one camera), which makes triangulation impossible.

Nevertheless, according to the literature **[4]**, it is possible to reconstruct ball shots trajectories in 3D using a single point of view and additional information about the object movement properties. Indeed, as we will describe more in detail later, each single ball shot involves generic movement equations that we can exploit to rewrite some relations and still be able to (approximatly) recover $[X, Y, Z]$ real world positions with only one perspective, only by finding a set of physic parameters (namely initial positions, speeds and accelerations) that should entirely determine the (real) trajectory.

This method just relies on the projection matrix P coefficients as well as the sequence of 2D points and their corresponding timestamps. Therefore we will need before all to find and compute P, and then we will have all we need to apply the method described above.

2.2. Cleaning input sequence

The input sequence of 2D points $[x, y]$ should, as stated before, only represent timepoints corresponding to a **single** shot (e.g the time interval between two successive surface-ball contact). Therefore we will retain only a well-chosen part of the video during which the ball is evolving in a single parabolic movement. The sequence of points returned by the tracking algorithm on this part of the video will then be our input sequence. The process can be repeated for different well chosen parts of the video, and the shot trajectory

reconstructed for each of these. But we insist on the fact that the chosen points should belong to a **single** shot sequence, therefore, given our limitation that forces us to limit the reconstruction to one single shot phase (between to bounces), we have to define and delimit a small useful part of the video to work with (which points sequence will then be automatically 'clean' as we will see now).

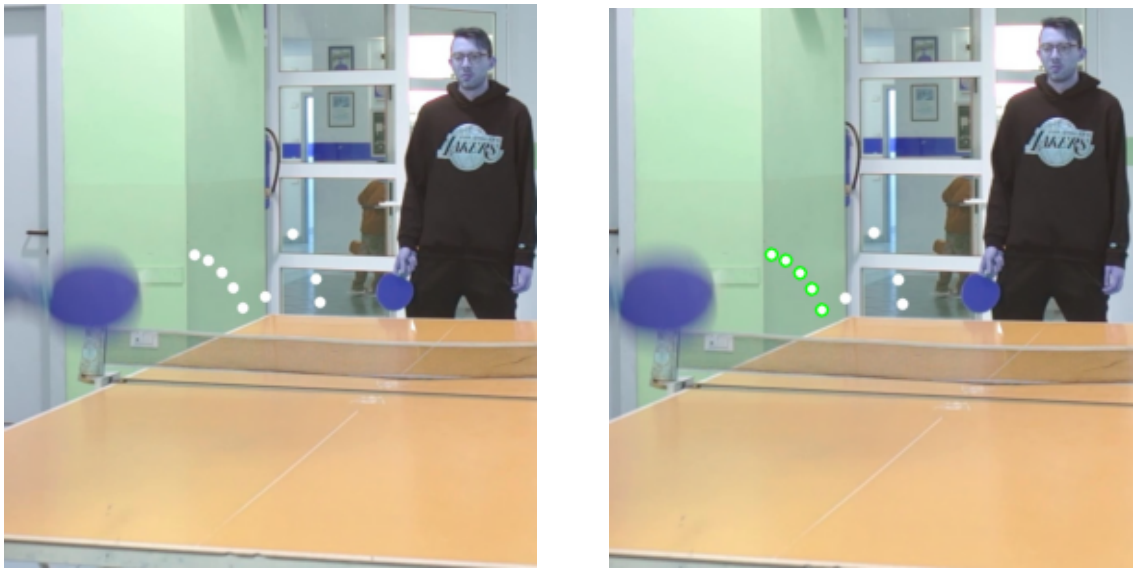
Now let us denote $S = ([x, y]_{t_1}, [x, y]_{t_2}...[x, y]_{t_n})$ our so-called input sequence of points. It might contain a few noisy points due to tracking weaknesses, as well as points that are well captured but does not belong to the considered single shot trajectory (e.g are "outside" the two bounces that delimit the shot we are trying to reconstruct). These "outlier points" might induce errors for the reconstruction process and thus should be removed from S. But we have to find a way to discriminate and detect those outliers among all the points.

We will use the following RANSAC-based methodolgy: the idea is that the "inlier" points should all more or less lie to a 2D parabolic curve as the original real movement is parabolic. Three points being the minimal set to define a parabola, we will iteratively:

- Select a subset $(p_1, p_2, p_3) \in S$,
- Fit a parabola P through this minimal set using least squares method,
- Gather all the points $p = [x, y]$ from S lying on P with an acceptable error (up to a parameter $\epsilon : \|y - P(x)\| < \epsilon$). This ensemble of points C is called the consensus,
- Keep and update of the maximal-sized consensus ensemble founded so far C_{max},

Until all the 3-combinations among S have been explored.

At the end, it returns C_{max} which should contain all the inliers (and exclude all the outliers) given the parameters we fixed (tolerance ϵ). Here is an example of our method abilities to discriminate outliers (retained points are in green):



(a) Initial (b) After RANSAC "sorting"

Figure 2.1: Useful points selection

Here for example we clearly see that the four unselected points to not belong to the shot trajectory we are trying to reconstruct because those are good points but captured before/after the bounce!

2.3. Camera calibration

As a first step, known as "camera calibration" step, we will focus on finding the projection matrix $P \in \mathbb{R}^{3\times4}$ which relate the $[X, Y, Z]$ points to their 2D mapping $[x, y]$ on the image plane by:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \lambda P \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \text{ up to any non-zero scalar factor } \lambda$$

Denoting p_{ij} the coefficients of P, the previous relation between a single pair of corresponding points gives two independent equations, that we can rewrite as follows:

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ & -y \end{pmatrix} \begin{pmatrix} p_{1,1} \\ p_{1,2} \\ p_{1,3} \\ p_{1,4} \\ p_{2,1} \\ p_{2,2} \\ p_{2,3} \\ p_{2,4} \\ p_{3,1} \\ p_{3,2} \\ p_{3,3} \\ p_{3,4} \end{pmatrix} = 0_{\mathbb{R}^2} \quad \textbf{(1)}$$

Where the goal is to find the twelve $p_{i,j}$. Therefore, if we accumulate N pairs of corresponding 2D and 3D points with $N \geq 5$, it raises at least 12 equations which makes it possible to find all the coefficients, by applying the DLT algorithm on a system of the following form:

$MP_{flatten} = 0_{\mathbb{R}^2}$, with $M \in R^{2N \times 12}$ being known.

Then, we select $N \geq 5$ points on the table image, which 3D coordinates are known in real life, given a world reference that we choose. Everything is summarized in the figure below.

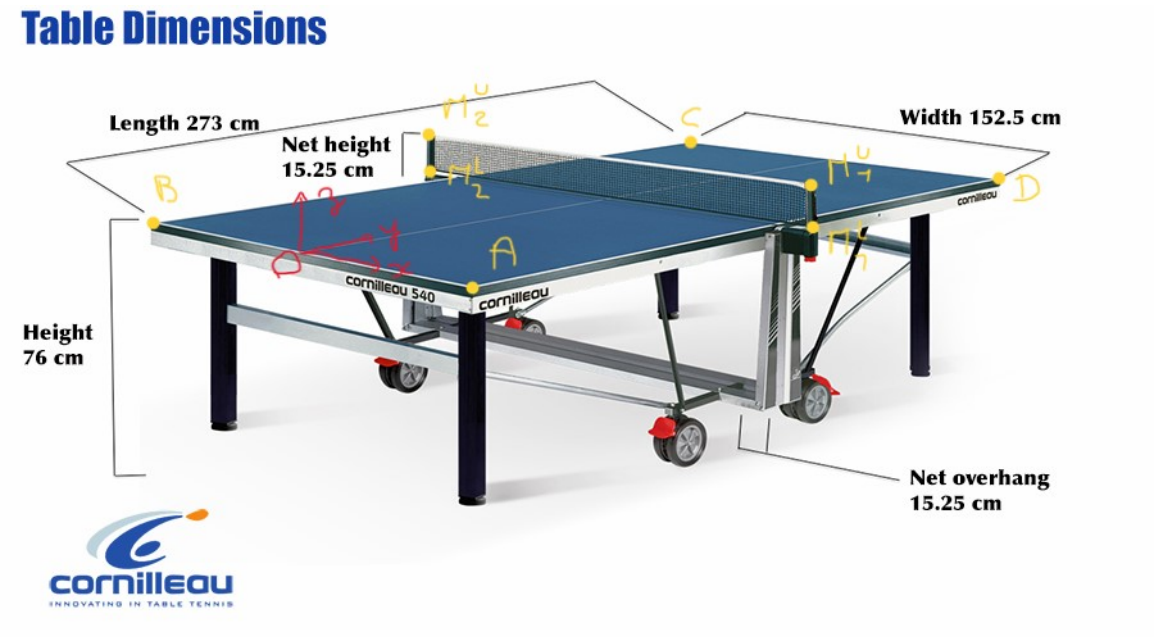


Figure 2.2: Corresponding points extraction

The finally obtained projection matrix can be verified by applying P on known real 3D points to see where they "land" on the image and compare with the location where we expected to find them. For instance, we tried to map the middle of the table, the upper middle of the net and the middle of each table quarter with our matrix P, and the obtained mappings "land on the image" as follows:

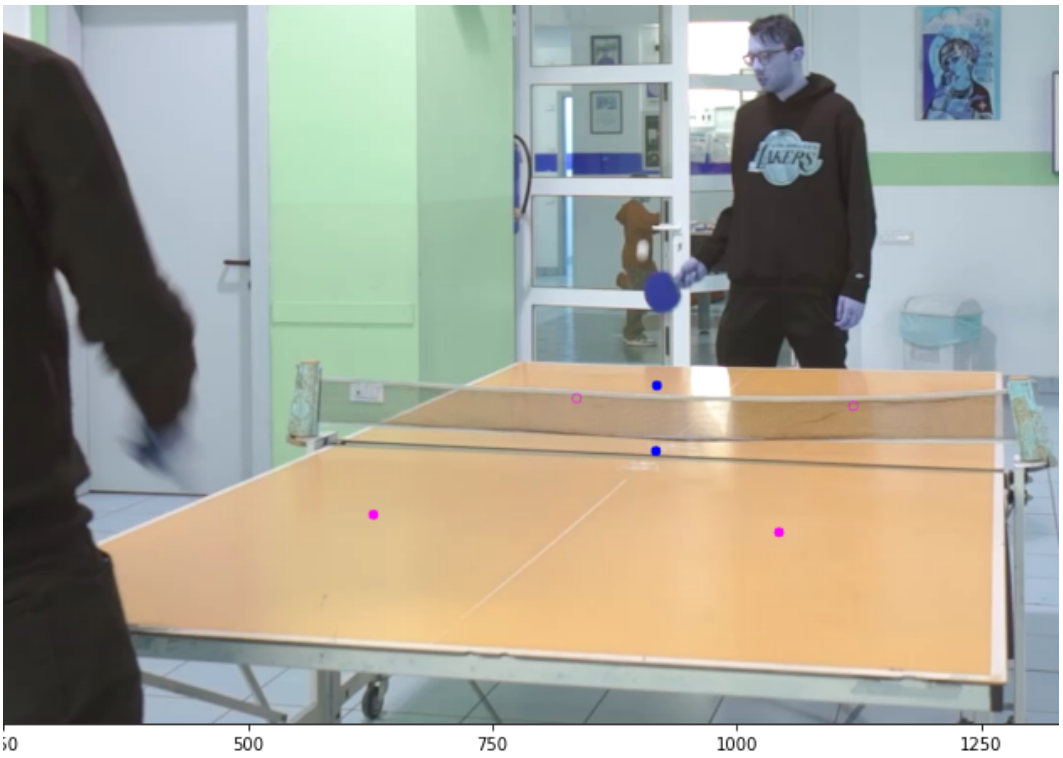


Figure 2.3: Verifying projection matrix coherence

It seems coherent and ensures more or less the quality of our result.

Having P, we can now go a little bit further and "factorize" it to get our camera intrinsics(parameters K) and extrinsics (rotation matrix$R$, camera center in world reference C), as explained in detail by NAYAR K Shree in his videos **[3]**.
Knowing that $P = [M|m] = K[R|t]$, we can exploit QR-factorization of M to find K, R and exploit the fact that m =-MC to find C.

To have an idea of the range orders (K coefficients might not be interpreted in absolute distance units), here is what we found:

$$K = \begin{pmatrix} 1514.65 & \text{-}32.97 & 991.09 \\ 0. & 1941.83 & 105.55 \\ 0. & 0. & 1 \end{pmatrix}$$

$$R = \begin{pmatrix} 0.99 & 0.17 & 0. \\ -0.03 & 0.18 & -0.98 \\ -0.17 & 0.97 & 0.18 \end{pmatrix}$$

$$C = \begin{pmatrix} 77.12 & -206.4 & 50.87 \end{pmatrix}$$

2.4. Extracting ball real movement using physics

Having P determined, the next step is to exploit this and the sequence of 2D points previously retrieved by ball tracking (from a single point of view), as well as their timestamps, to recover the real trajectory parameters of the ball. This method, described in detail by Luca PIROTTI in his project thesis **[4]**, relies on the mechanical equations of an object in free movement (with initial conditions).
Indeed, considering a single shot, the ball (real) coordinates at any time t can be described as follows:

$$\begin{cases} X_t = X_0 + v_x t \\ Y_t = Y_0 + v_y t \\ Z_t = Z_0 + v_z t - \frac{1}{2} g t^2 \end{cases} \quad where \text{ g} = 981 \text{ cm/s}^2 \text{ is the gravitationnal acceleration.}$$

These movement equations ensure that (during a single shot) the coordinates variables $[X_t, Y_t, Z_t]$ are decoupled and evolve independently of each other, which brings consequent additional information. Replacing $[X_t, Y_t, Z_t]$ by their time depending equations in formula **(1)** which relate them to their corresponding 2D points$[x_t, y_t]$, we can rewrite the latter as:

$$\begin{pmatrix} p_{11} - x_t p_{31} & p_{11}t - x_t p_{31} t & p_{12} - x_t p_{32} & p_{12}t - x_t p_{32} t & p_{13} - x_t p_{33} & p_{13}t - x_t p_{33} t \\ p_{21} - y_t p_{31} & p_{21}t - y_t p_{31} t & p_{22} - y_t p_{32} & p_{22}t - y_t p_{32} t & p_{23} - y_t p_{33} & p_{23}t - y_t p_{33} t \end{pmatrix} \begin{pmatrix} X_0 \\ v_x \\ Y_0 \\ v_y \\ Z_0 \\ v_z \end{pmatrix}$$

$$= \begin{pmatrix} x_t(\frac{p_{33} g t^2}{2} + p_{34}) - (\frac{p_{13} g t^2}{2} + p_{14}) \\ y_t(\frac{p_{33} g t^2}{2} + p_{34}) - (\frac{p_{23} g t^2}{2} + p_{24}) \end{pmatrix}$$

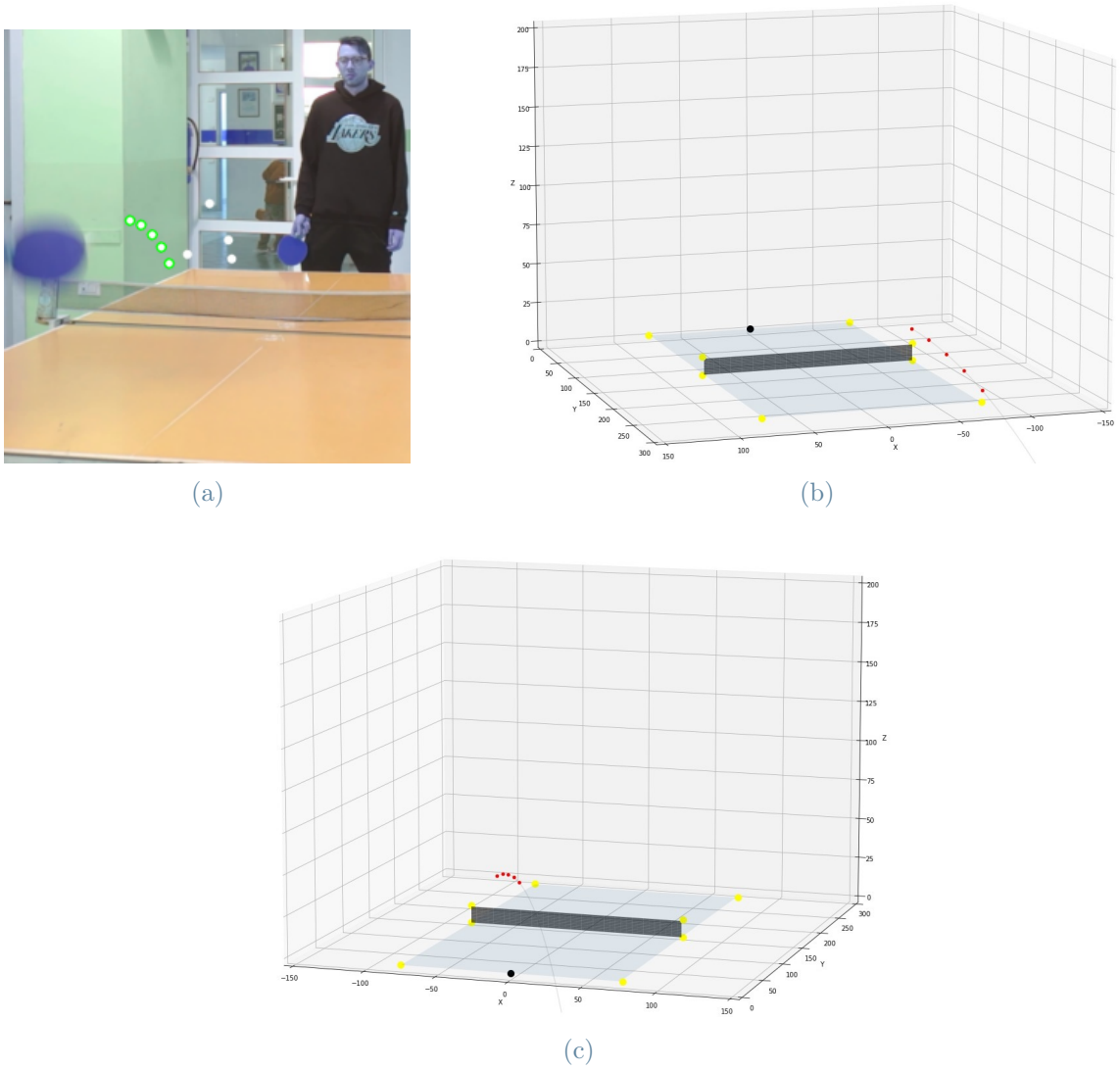
Which leads us to a matricial system of form $AX = B$, X being the unknown vector to found and A and B known elements (as we now have full knowledge of P).

Using the least squares method of the *NumPy* linear algebra library, we can find a solution vector $X = \begin{pmatrix} X_0 & v_x & Y_0 & v_y & Z_0 & v_z \end{pmatrix}^T$ that minimizes $\|AX\text{-}B\|$, and therefore gives an optimized approximation of the real trajectory parameters.

Having this parameters, we can now compute the real trajectory (and thus the real coordinates) of the ball at any time, not to forget that it works only for a single shot (e.g the interval of time between two consecutive surface-ball contacts). To retrieve the real ball positions at any other time of the game, we would need another sequence of well chosen timestamps, delimiting a **single** free movement phase!

2.5. Results discussion

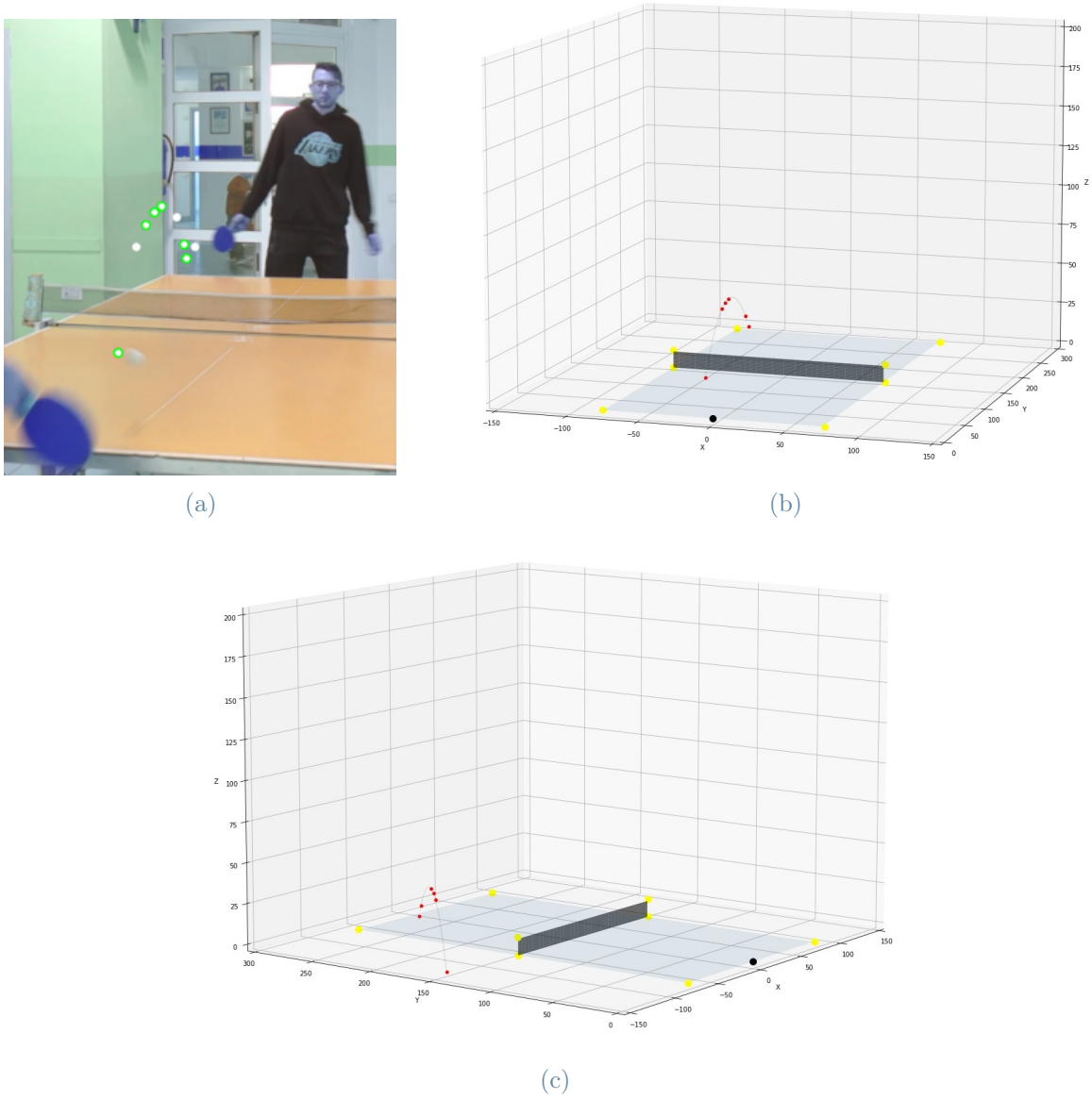
Here are some results of the previously described experiences, plotted in 3D.



(a) (b)

(c)

Figure 2.4: Reconstruction example 1



(a) (b)

(c)

Figure 2.5: Reconstruction example 2

As we can state, sometimes it performs more or less well (see example 1), but sometimes the result obtained is notably far from the groundtruth we expected to find back. the model, in addition to being limited to the reconstruction of a single shot on a short sequence self-defined, shows rather poor performance sometimes. This slightly disappointing final result is not surprising given all the accumulated/multiplied precision losses along the overall pipeline.

3 | Conclusion: limitations and possible improvements

Through this project we found a way to perform 3D reconstruction from a single point of view, using a specific situational method based on movement Physics. However, as stated before, the model is not very robust and the experimental results are sometimes disappointing. It can be explained by many factors:

- Poor ball tracking due to bad video conditions, hence we could hardly reach great performances and were very limited by that
- Manual points selection for correspondences might lead to inaccuracies when computing the coefficients of P
- RANSAC algorithm to clean the input sequence is not very robust: we can only extract the "right" sequence of points on already short input sequences, and it often involve some parameter (tolerance) tuning to make a "satisfying" selection, hence it lowers the degree of automatization in the pipeline
- The assumption that the ping pong ball is submitted to such simple movements law is maybe too optimistic: it is a very specific kind of ball (light and empty) which movement equations should probably include much more complex elements in physic equations as air resistance, friction forces...

Some possible experiences and improvements are left open for a next project:

- For another trial, we can begin by filming each gameplay sequence with different points of view, using multiple smartphones placed variously around the table. Hence we would not be very limited since the beginning and we could apply more robust method as calibrated or uncalibrated triangulation.
- We could have try to track also the black face of the first player's racket to keep tracking of the two faces and then of the racket at any time and in any case.

- In the case where we would run into such another project in the future but with a multiple points of view advantage, we can also think about recovering the players movements (e.g forehands and backhands) in a 3D environment.

4 | References

[1] U-Bourgogne University, "Analyse d'images: Opérateurs morphologiques et classification" course, https://ufrsciencestech.u-bourgogne.fr/master1/TraitementImages/partie_MIGNIOT/cmTI.pdf

[2] SIMSEK Gökhan, Article "Object detection from image and video using HSV color space", https://blog.paralect.com/post/object-detection-image-and-video

[3]NAYAR K Shree, Columbia University, "Camera Calibration" Videos, https://www.youtube.com/watch?v=S-UHiFsn-GI&list=PL2zRqk16wsdoCCLpou-dGo7QQNks1Ppzo

[4] PIROTTA Luca, Thesis "Single view ball tracking and 3D trajectory reconstruction in basketball videos", https://www.politesi.polimi.it/handle/10589/142844