Object detection for the metaverse

Mattia Franzin 239930, Riccardo Parola 239928

1 Introduction

The idea behind this project is to map real objects into a virtual scene, in order to create a virtual reality environment, where the player can perceive an object through their headset and grab it with the feeling of touching it physically. This can have multiple uses in various fields ranging from medical use to didactic. We choose a simple object to track, and developed different approaches for tracking, with a specific focus on delay and accuracy since these two aspects are fundamental for immersion in the virtual environment and for the interactions. In order to have a simple object of a given size, we opted to 3D print a cube, which allowed us to precisely set our dimensions. The task was divided into different approaches: first with a Motion Capture system by Optitrack and then, using a camera or a stereo camera to try to get the same information. All the code and the environment for the development are on GitHub.

2 Different approaches

We wanted to try and experiment with different methods, we had access to a very heterogeneous group of tools for this task provided by the university. We focused on four different approaches as they were the most used in literature for object tracking:

- Motion capture;
- Color;
- Stereo camera;
- ArUco;

We compared the solutions and draw conclusions on the best method to use in our use case.

2.1 Motion capture approach

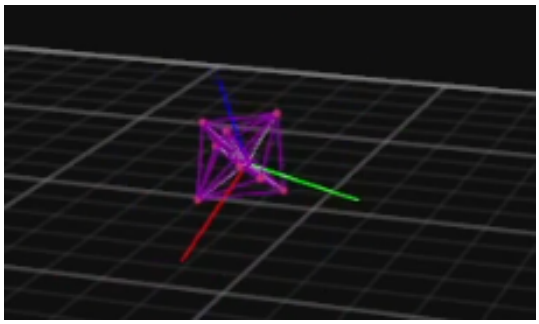

Figure 1: Cube Rigidbody in Motive.

Motion capture is a system that uses a large number of infrared cameras to capture the position with very high accuracy and very low latency of markers. The system that we used was composed of eight high-speed cameras that have an infrared light built-in. The markers that we used are small spherical, and reflective, which allows the cameras to triangulate correctly each single marker from all

directions even if occluded for 5 of the cameras. This is an optimal situation for this type of project since precision and latency are crucial for the immersion of virtual reality.

Figure 2: Cube with attached asymmetric markers.

First of all, we had to prepare the object that we choose in order for the system to track it. We attached the Motion Capture markers to our cube in an asymmetrical pattern as in fig. 2: this allows us to understand which side of the object are we looking at, otherwise, if we placed the markers evenly on each side of the cube, we would not be able to understand its rotation. We used the software called Motive which allows us to use the camera system to track the objects creating a rigid body and assigning that marker position to our cube and giving it a unique id. Once the object is registered the software uses the distance between the markers to identify uniquely the object. Thanks to a Unity add-on than we where able to stream the data captured by the system to Unity which allowed us map the points to a 3d asset. With this we were able to precisely track the movement of the object but we encountered some problems: we noticed that the mapping between the real object and the model was not correct, it was in fact 90 degrees rotated. We tried to manually handle it, but we noticed that our cup was no longer moving or rotating, although the code was correct. To fix this, we changed the rotation in which the cube was registered to match the cup rotation, and after doing it, the cup was shown correctly. We think this problem was related to different frames of reference in the 2 software: in Motive, the Y axis is the vertical one, while in Unity it is the Z one. In unity we than created a scene, recreating a soccer field where a user could rise the cup.

2.2 Color approach

Using a single camera is very challenging to track precisely an object's motion especially if accuracy is key for immersion. We explored different possible starting from using colors to identify and track the position of the object. The first idea that came to our mind was to use a Rubik's Cube instead of a plain one so that we can extract the color information to identify it and calculate the rotation of the cube. We created a mask for each color to subtract to determine the position and to recognize the cube's face we were looking at as in fig. 4. We used the HSL color space which theoretically performs better than the others allowing us to don't take into account the light changes in the scene. After some initial problems with the color threshold we were able to detect and identify each color of each face, even though the environment light can alter the perception (a yellow light will make the white face yellowish), but, after relaxing our color ranges and some fine tuning with thresholding and blurring we were able to perceive the colors correctly as shown in fig. 3.

The tracking was more challenging because the idea was to get the visible colors and then, using their areas, calculate the percentage of each singular color with respect to the totality: so for a cube rotated 45° along the z-axis, ideally, we should see 2 colors with 50% each, and so on. It turns out that mapping that color percentage information to an actual rotation was quite difficult, and probably more axis rotation may lead to the expected rotation, and trying to estimate it starts to become difficult because it works only for some faces. This solution is very environment and camera dependent, we noticed that for each different camera that we used, we needed to change the thresholds for the color masking making it not stable.

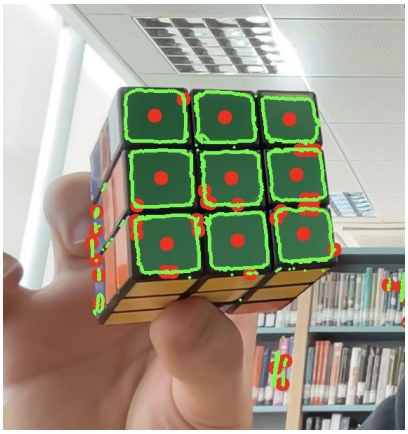

Figure 3: The color approach on a rubrics cube.

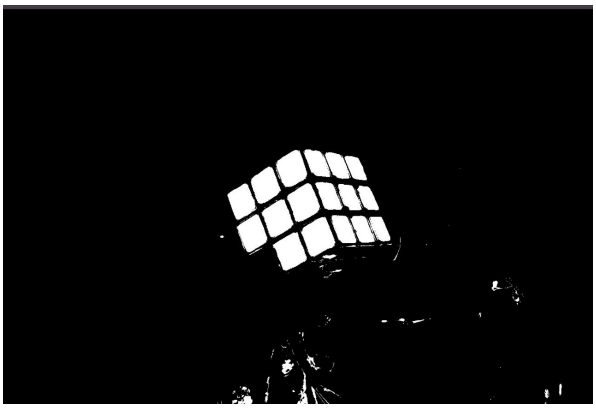

Figure 4: Color mask on rubrics cube.

2.3 Stereo camera approach

The colored approach had a second flaw, it is very challenging to estimate the distance of an object precisely with only one camera so we opted to a stereo camera. A stereo camera like the ZED 2 that we used, is a type of camera that uses two lenses to capture images and video in 3D. The two lenses are spaced apart to mimic the distance between our eyes, which allows the camera to capture two slightly different views of the same scene. This creates a point cloud, which can be used for a variety of applications. The combination of the color detection for the rotation and the distance estimation from the stereo camera theoretically should have good results. We realized soon that this solution was not feasible since the distance and rotation estimation were not very accurate even with a lot of fine-tuning.

Figure 5: Stereo camera visualization.

2.4 ArUco approach

ArUco is a great technology that allows through stickers with a black and white pattern to track precisely the distance and rotation. This method is based on different computer vision algorithms like corner detection and edge detection to identify the maker and its id associated. There are different methods that given the dimension of the ArUco can estimate the distance and its rotation very precisely.

2.4.1 Calibration

Most cameras introduce significant distortion to images especially cameras like zed who has a wide-angle lens. Two major kinds of distortion are:

- Radial distortion

  - causes straight lines to appear curved, it becomes larger the farther points are from the center of the image

- Tangential distortion
  - occurs because the image-taking lens is not aligned perfectly parallel to the imaging plane this cause for some areas in the image to look nearer than expected.

For this approach to work is fundamental to compensate for these defects of the camera, and this is done using a black-and-white pattern which is simple to analyze with corner and edge detection algorithms and a script that extracts a calibration matrix and distortion parameters along the camera matrix that contains physical information such as focal length and optical centers (is the part of the image in witch there are no distortions)

$$Camera\_matrix = \begin{bmatrix} f_x & 0 & c_x \\ a & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

We used the grid in fig. 6 and the OpenCV library to register the calibration. This metric allows for compensating the specific lens distortion to get a precise camera model that will be used along with the ArUco markers to estimate their positions in space.

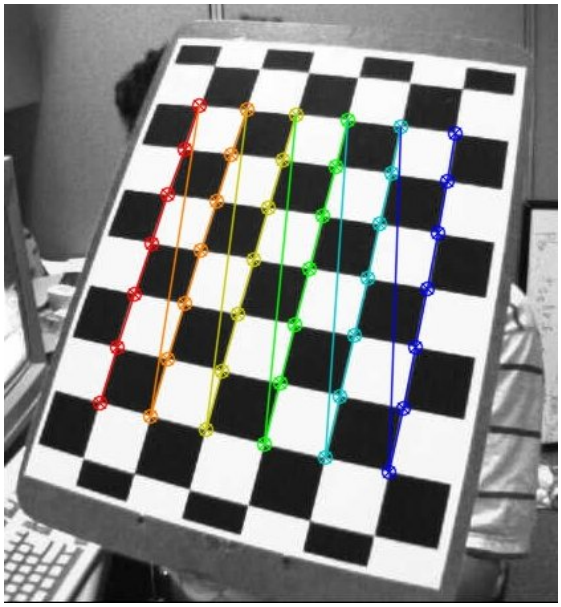

Figure 6: Calibration grid from OpenCv library.

2.4.2 Rotation

We applied a different marker on each side of the cube with a white border: this is required since this method works best when there is great contrast around the marker. Since this method allows us to easily calculate the distance of the marker, a simple camera is enough. When more than one marker is perceived, only the one with the biggest area is used to calculate the rotation of the cube, since the error probability is lower; then, a relative rotation is assigned to each face of the cube, based on how the marker was placed. When a marker is detected, a list of translation and rotation vectors are returned and, as shown on fig. 7, they can be used to project any point in the correct perspective, allowing us to 3D reconstruct the cube starting from just a face of it. Also reference frames can be projected on the markers, and knowing the relative rotations between different faces of the cube, a global and coherent reference frame can be constructed to represent the actual rotation of the cube. But that's not all, first because we have to convert this rotation vector into a rotation matrix or a quaternion in order to use it, but we also have to calculate the centroid of the cube: to do that we know that it lies

behind each face, with a distance equal to the length of a marker (marker is already half-size a cube edge), so we just need to project that particular point using our rotation and translation vectors. The result is very precise as seen in fig. 8, even if this method does not work with objects that are too far, since the markers would not be perceived. Using centroid and rotation information, an object of any shape can be inserted into the Unity scene.

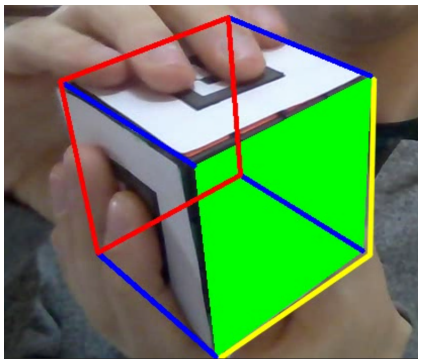

Figure 7: 3D cube reconstruction.

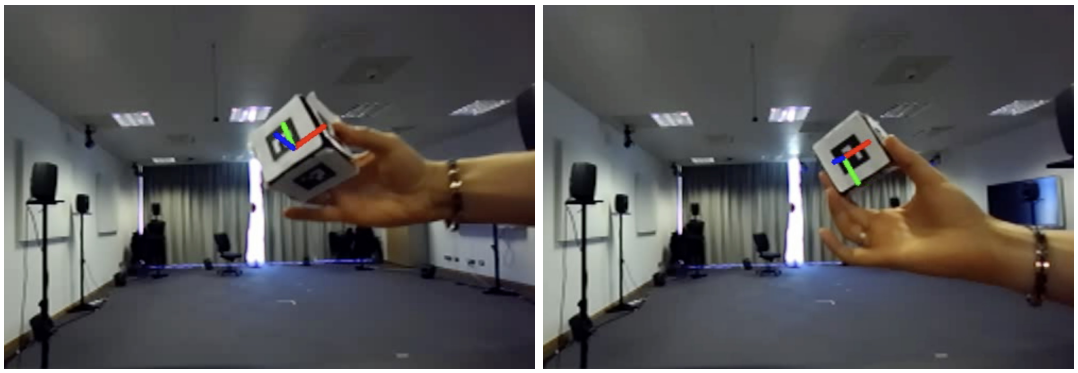

Figure 8: ArUco centroid estimation.

3 Conclusions

Different solutions were tested but these are very specific and system dependent which means that no solution is the best.

- Motion capture
  - requires a complex system that is very expensive and difficult to operate
  - very precise and responsive with very low latency and is not affected by occlusions
  - object needs setup with markers

  The most reliable and precise solution is the motion capture system but at the same time the most expensive and complicated to set up. Motion capture has many advantages in this task which range from very low latency, high precision, and a very high capturing rate to the complete absence of occlusions.

- Color combined with stereo camera
  - requires a fairly simple setup easy to operate and setup
  - not very precise and responsive affected much by different lighting conditions and occlusions

The color solution on the other hand is a compromise on the hardware needing only the stereo camera for the depth estimation but the performances are very poor. This solution is very system and environment dependent with many cases in which needs to be re-tuned accordingly for each situation.

- ArUco:
  - setup extremely simple with a simple camera
  - object needs setup with ArUco markers

  The most simple setup is the ArUco which is the second by performance, giving a reliable solution that only needs a simple camera. The only setup that needs to be done is the camera calibration which is an automated calculus by simply putting the chessboard in front of the camera the grid, as well as placing the markers in the correct order and orientation.

In this project, we explored different solutions for different setups, comparing the results. Each of the approaches was interesting to work with, while exploring different solutions for the same problem. We compared the solutions we found outlining the differences with upsides and downsides. Overall we suggest the ArUco and the Motion Capture approaches, that have worked the best for us: they can be easily adapted to your needs without having to waste too much time, letting you focus just on your own tasks.