Mines Paristech

Engineering course 2nd year

Part-time semester at Sorbonne university - Institute of Robotics and Intelligent Systems

Report written by

David POULAIN

Machine learning for Augmented Reality with robotic video projection

September 2018 - January 2019

Supervision

Mr	Cédric HONNET	Research Engineer at the ISIR lab (HCI group)
Dr	Yvonne JANSEN	CNRS Researcher at the ISIR lab (HCI group)
Dr	Sinan HALIYO	Associate Professor

Abstract

To answer the limits of projection for augmented reality, this thesis presents a robotic projection-based system sending information from the ceiling, the Ceilibot. The concept is to motorize a video projector associated with a depth sensor and camera to make them rotate and translate on rails fixed on a ceiling in order to control the projection zone.

This report is about the beginning of the coding : the hardware is already built. It will focus on tracking and following users, and how to find flat surfaces suited for projection. After a first part that describes the state of the art around projection-based augmented reality, the second part presents the purposes of the Ceilibot. The third part is about the work I did here, at ISIR, during my internship and the final part is meant for the students that will follow me, to give them ideas of improvements and details about how my code works.

This is one of the first steps in the development of the Ceilibot focusing on the basic principles for tracking, following, finding surfaces. It is the first step of the software development.

Contents

Introduction 1

1 State of the art 3
- 1.1 PenLight . . . 3
- 1.2 LightSpace . . . 4
- 1.3 OmniTouch . . . 4

2 A global project 7
- 2.1 Description of the entire project . . . 7
- 2.2 The specific project I will be working on . . . 8
  - 2.2.1 Description of the project . . . 8
  - 2.2.2 State of progress . . . 9
  - 2.2.3 Personal Objectives . . . 9
  - 2.2.4 Hardware specifications . . . 9
    - 2.2.4.1 The NVidia Jetson TK1 embedded dev kit . . . 9
    - 2.2.4.2 The Asus Xtion Pro . . . 10
    - 2.2.4.3 The Intel RealSense R200 . . . 10
    - 2.2.4.4 The Optoma GT1080e . . . 11

3 Work achieved 13
- 3.1 Objectives . . . 13
- 3.2 Hardware and software used . . . 13
- 3.3 Detection and tracking . . . 13
  - 3.3.1 Different approaches and defects . . . 13
  - 3.3.2 Limitations . . . 14
- 3.4 Motor command . . . 14
  - 3.4.1 The principle . . . 14
    - 3.4.1.1 Case with only one user . . . 14
    - 3.4.1.2 Case with two or more users . . . 15
  - 3.4.2 Issues and solutions . . . 15
  - 3.4.3 Limitations . . . 16
- 3.5 Finding flat surfaces . . . 16
  - 3.5.1 The Principle . . . 16
  - 3.5.2 Detouring . . . 16
  - 3.5.3 Find the biggest surface . . . 17
    - 3.5.3.1 The idea of the cross . . . 17
    - 3.5.3.2 Grow into the biggest rectangle . . . 18
  - 3.5.4 Limitations . . . 19

4 Continuation 21
- 4.1 About the code . . . 21
  - 4.1.1 Ordering the classes . . . 21

4.1.2 Dealing with a choice to do between equal surfaces 21
4.2 Ideas for improvement 22
4.2.1 Project on a surface the user is looking at 22
4.2.2 Loss of tracker accuracy while moving 22
4.2.3 Take perspective into account for surface finding 22
4.2.4 Project a non distorted image 22

5 Conclusion **23**

List of Figures

1.1 The PenLight system . . . 3
1.2 The Lightspace system . . . 4
1.3 The OmniTouch system . . . 5

2.1 The WheelyBot . . . 7
2.2 The CeiliBot . . . 8
2.3 CeiliBot concept . . . 8
2.4 The CeiliBot . . . 9
2.5 The NVidia Jetson TK1 . . . 10
2.6 The Asus Xtion Pro . . . 10
2.7 The Intel R200 . . . 11
2.8 The Optoma GT1080e . . . 11

3.1 Margins in camera view . . . 14
3.2 Motor moves if the user is off the margins . . . 15
3.3 Motor moves if one user is off margins, and we find the centroid of all users 15
3.4 Centroid (red cross) is in the center, and users are inside margins . . . 15
3.5 Frame from camera stream . . . 17
3.6 Result after Canny() function . . . 17
3.7 Biggest cross found in red . . . 18
3.8 Biggest square centered on cross's center . . . 19
3.9 Biggest 2 side extension from previous square . . . 19
3.10 Biggest 1 side extension into a rectangle . . . 19

Introduction

During the fourth year of my Master's degree in engineering at Mines Paristech, I got the opportunity to intern in the ISIR Laboratory (Institute of Intelligent Systems and Robotics).

In the last years, we have observed a rapid emergence of systems and technologies based on virtual and augmented reality (AR). If virtual reality is maybe more hyped than AR, the latter is becoming more and more popular thanks to its potential to improve the real world with elements of the virtual world. We can define AR as an enhanced version of reality where live views of physical environments are augmented with superimposed computer-generated images over a user's view of the real-world, thus enhancing one's current perception of reality.

The fields using AR are fairly diverse already: medicine, history, advertisement, entertainment, industry, etc. and it is becoming a part of our lives as many smartphone applications are using it too. Projection-based AR uses a projector to project media on real surfaces, sometimes allowing interaction with the user.

While there are many advantages about projection-based AR as a direct immersion, such as multi-users utilization or avoiding to carry a head-mounted device (HMD), there are also some limits. The main flaw of projection-based AR is the problem of mobility and the limited possibilities of the projection area because the system must be fixed somewhere. To overcome these problems, the solution chosen was to use a robot. Implementing a projection-based system on a moving robot open many opportunities. The robot I was given is called the Ceilibot : it is a robot attached via a rail to the ceiling. It can move all along the rail that goes all around a room.

The goal of this internship is to build the first software pieces to enable the Ceilibot to find users, track them, move and follow them, and find adequate surfaces for projection. For this, it uses a Kinect-like camera, and two motors (one for moving along the rail, the other to control the projection's tilt). It will be used for human-computer interactions in AR applications.

After introducing the subject of projection-based AR and presenting the related work, this report describes the process followed to build the premises of the robot's software. The first step was to understand and make every part work fine on its own, then to find how the different parts would interact, finally to think of smart ways to process all the data in a reasonable time.

1 State of the art

Many existing systems use projection-based AR. Inquire about them will allow us to discover the different techniques and the limits of each solution to create our own system.

1.1 PenLight

PenLight is a digital pen that increases the visual feedback of classic digital pens. To provide richer visual feedback, the idea is to have a digital pen embedded with a spatially-aware miniature projector including a 3D optical tracking technology to capture the 3D location of the pen. Its main goal is to visually augment physical paper to enable virtual functionality with paper and multi-layer interactions.

The strong point of PenLight is the control of multi-scale widgets and the navigation in virtual layers of information and multivalent documents. The interaction design space can be divided into several input layers and display layers.

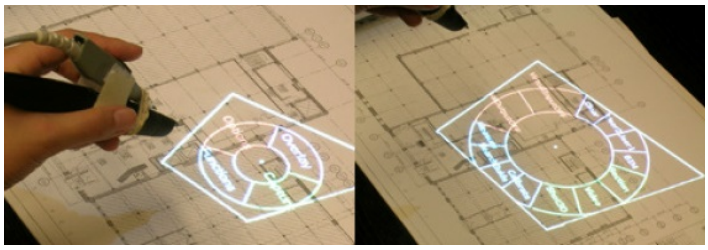

Figure 1.1: The PenLight system.

For the PenLight prototype, the three components were separated but they were implemented to simulate as if they were mounted on the pen. The components used are a Destiny IO2 Bluetooth digital pen, a Polhemus FastTrak 3D magnetic tracker and a Mitsubishi XL4U video projector.

The 2D tracking is accomplished with a camera inside the pen that recognizes its location on the page and the page number. A pressure-sensitive tip switch on the pen senses when the pen is in contact with the paper and a wireless Bluetooth connection links the pen with the CPU, so that the pen strokes can be stored virtually, and if desired, displayed, in real time.

Because the real project will integrate the projector and the 3D tracking onto the digital pen unlike the prototype, an issue is about improving the optical pattern tracking as high accuracy 3D tracking is not existing yet, and the image stability due to the high frequency of pen movement. And the location of the projector must be also thoughtful since it determines the size of the projected image and the pen's center of mass.

1.2 LightSpace

LightSpace is a project leaded by Microsoft Research. It is a smart room that allows the user to interact in the room thanks to projectors and depth cameras. It can transform any flat surface in the room into an interactive display where users can use hand gestures and touch to manipulate projected content.

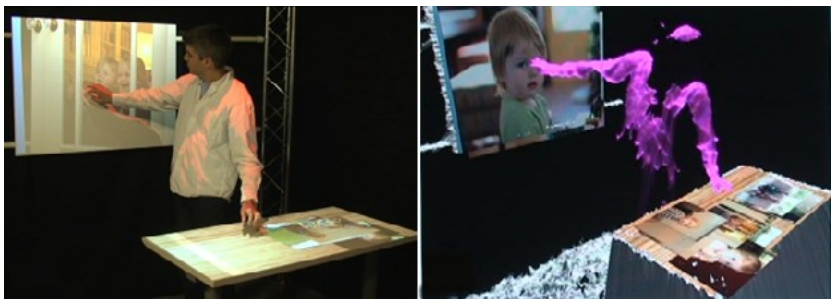

Figure 1.2: The Lightspace system.

LightSpace is a 10ft x 8ft x 9ft interactive space with 3 projectors and 3 depth cameras suspended on the ceiling. The user can move objects between interactive surfaces through-body by simply touching the object and then touching the desired location. The user can literally drag an object off an interactive surface and pick it up with their bare hand. And detecting the user position allows various spatial interfaces as moving the hand up and down to open up a menu directly on the hand. The interactive functionality relies on the ability of the cameras to calculate depth of the objects in the scene.

The camera body houses an IR camera, a RGB camera and an IR light source. The light source projects a pattern on the environment. The IR camera captures this pattern and computes depth from the distortion of the pattern in the image. The resulting depth image contains a depth estimate for each pixel in the image. After the calibration of the cameras, the projectors and the simulated interactive surfaces, the cameras can capture in real time a 3D mesh model of the entire sensed portion of the space. Virtual objects may be placed on top of the mesh in the same scene.

To detect the user and his movements, LightSpace uses a technique less complex than hand or skeletal tracking. The system computes the projection of the 3D data to create a new image that can be thought of as having been generated by a virtual camera. This data is rendered to a single virtual camera view and it is analyzed using simple 2D image processing techniques.

1.3 OmniTouch

Another project leaded by searchers from Microsoft Research is OmniTouch **?**. It is a wearable depth-sensing and projection system that enables interactive multitouch applications on many surfaces of the user environment including his own body as his hands, arms or legs. The system must be worn on the shoulder.

It is composed of a custom short-range PrimeSense depth camera, a Microvision ShowWX+ laser pico-projector and a desktop computer for prototyping purposes.

Omnitouch can detect multitouch input on surfaces by tracking the user fingers (Figure

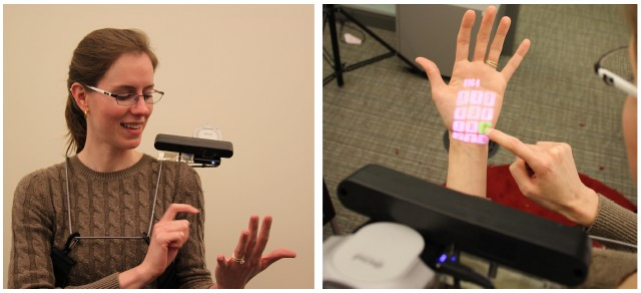

Figure 1.3: The OmniTouch system.

??). A depth map of the scene is taken and the depth derivative is computed using a sliding 5x5 pixel window. Then, vertical slices of cylinder-like objects are researched on the derivative image for path finding and tip estimation. This technique is easier to process because it treats a conventional 2D image.

To detect if a finger is in contact with a surface (finger click), OmniTouch uses a secondary process. It computes the the midpoint of the finger path. Then, it floods filling towards the fingertip. When the finger is hovering above a surface or in free space, the flood fill expands to encompass the entire finger. When the finger contacts a surface, the fill operation floods out into the connecting object.

Many applications were developed on OmniTouch as a keyboard application, a menu navigation, a post-it application that let the user writes quick notes on his palm, a map panning or a painting application where the left arm is the color palette and the right hand the brush.

2 A global project

2.1 Description of the entire project

The project I have been working on is part of a much bigger project : its aim is very general and is not restricted to only one support. The goal of this project is simple : build a device that is :

- Autonomous
- Able to track and follow users
- Able to understand what they are doing
- Able to use Augmented Reality to display useful information related to the recognized activity

In order for such a project to be achieved, a few main axes of research can be found :

- Hardware : what will be the support of the device ?
- People tracking and activity recognition : what camera and software to use ?
- Move and follow users : how to have the device follow users ?
- Augmented Reality : How to know where to project, what to project ?

When I got my internship at the ISIR, I was given the choice to continue the work done on one of those two already existing supports : one named the WheelyBot and the other named the CeiliBot.

The WheelyBot is a four-wheeled robot, of about 10cm long. It is supposed to be able to follow a user by driving after him, quite like a dog with an augmented reality gadget on its back.

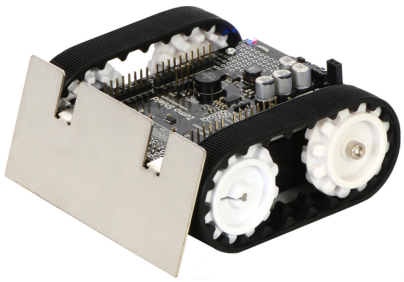

Figure 2.1: The WheelyBot

The CeiliBot is basically a moving platform attached to the ceiling by a rail that goes all around the room. On the platform is a kinect-like camera and a video projector. I chose the second.

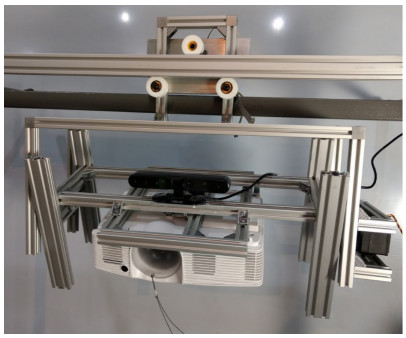

Figure 2.2: The CeiliBot

2.2 The specific project I will be working on

As previously said, I chose to continue the work done on the CeiliBot

2.2.1 Description of the project

The name "CeiliBot" comes from "Ceiling" and "Robot" : it is basically an autonomous robot attached to the ceiling via a rail. This rail goes all around a room, about one meter away from the walls, thus enabling the robot to move all around the room.

The robot itself is composed of a computer, a camera (RGB -for Red Green Blue - and depth), a projector and two motors. The computer orchestrates all of those parts. The camera is meant for user tracking, detection of surfaces for projection and activity recognition. The projector is meant to display data through AR (Augmented Reality) and the motors are meant to move the device (one allows to move along the rail, the other allows it to tilt downward or upward to follow a user or to find a surface fitted for projection.

The image below is meant to fix ideas : it is the concept of the robot. You can see the rail to which the robot is attached (imagine the motor here to make it move), the structure, the pivoting angle and at the end, the camera (Kinect-like) and the projector.

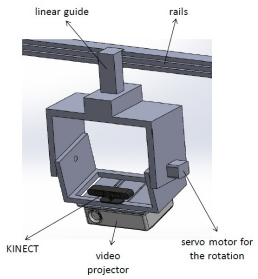

Figure 2.3: CeiliBot concept

2.2.2 State of progress

When the project was handed to me, only the structure had been done. Here is a picture of what it looks like :

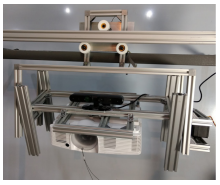

Figure 2.4: The CeiliBot

I had all the hardware required : the structure was built, the motors were in place, the camera, computer and projector were bought. What was left to do was all the coding work : acquiring signals from the cameras, analyzing images to find users and flat surfaces, running the motor to follow the users, recognizing human activity, projecting on the desired area... You will find just below the specifications of all the hardware that was left to me.

2.2.3 Personal Objectives

After talking with my internship supervisor, I cleared those objectives among all that needed to be done :

- Detect and track people
- Follow users
- Find flat enough surfaces for projection

2.2.4 Hardware specifications

Those are the specifications of all the hardware that was handed to me and that I used during my internship.

2.2.4.1 The NVidia Jetson TK1 embedded dev kit

This is a well-known kit for machine learning programmers. It is basically a card of about 10cm x 10cm that runs Linux and all we need to make our robot work.
https://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

Its specifications are :

- NVIDIA Kepler GPU with 192 CUDA Cores
- NVIDIA 4-Plus-1^{TM} Quad-Core ARM® Cortex^{TM}-A15 CPU

Figure 2.5: The NVidia Jetson TK1

2.2.4.2 The Asus Xtion Pro

This is a camera (depth and RGB) manufactured by RealSense, the same company that makes the famous Kinect.
https://www.asus.com/fr/3D-Sensor/Xtion_PRO/specifications/
Its specifications are :

- VGA : 30 fps, 480 x 360
- View angles (degres) : Horizontal 58, Vertical 45, Diagonal 70
- Depth view : between 80cm and 3,5m

Figure 2.6: The Asus Xtion Pro

2.2.4.3 The Intel RealSense R200

This is a camera (depth and RGB) manufactured by Intel.
https://ark.intel.com/fr/products/92256/Intel-RealSense-Camera-R200
Its specifications are :

- VGA : 30 fps, 480 x 360
- View angles (degres) : Horizontal 59, Vertical 46, Diagonal 70
- Depth view : between 50cm and 3,5m

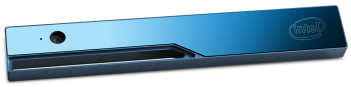

Figure 2.7: The Intel R200

2.2.4.4 The Optoma GT1080e

This is a video projector manufactured by Optoma.
https://www.optoma.fr/projectorproduct/gt1080e
Its specifications are :

- 1080p, short focal, 3000 lumens
- consumption : 10W

Figure 2.8: The Optoma GT1080e

3 Work achieved

3.1 Objectives

As said before, the objectives are to :

- Detect and track people
- Follow users
- Find flat enough surfaces for projection

3.2 Hardware and software used

For this work I am using a Jetson Tegra K1 board. The camera is an Asus Xtion Pro Live. Later, due to some troubles, I will use the Intel RS200. All the specifications can be found right above.

I coded in C++ because it is widely used for this purpose and quite easy to understand and use.

3.3 Detection and tracking

To begin with, let's make the difference between the two : detection is performed on a single image and aims to find a specific thing in the picture (a human being in our case). Tracking involves multiple images (frames) and aims to follow the moves of one particular object that was seen on the first frame. Therefore, if we want to know where some users are located, we first have to detect them, and then have to track them.

3.3.1 Different approaches and defects

The first idea to track a user in an environment could be to look for what is moving : if the environment is inert, then the only moving thing is our user. The major issue here is that we want to be able to locate the user(s) even if their activity makes them move very little, so we need some more complex detection program.

To find a human in a still picture, machine learning is certainly the best tool. The idea is simple : you show a lot (thousands) of images to your computer and on each of them show it where are the humans. Afterward, you just ask your computer to look at your own image and it shall be able to find humans in it. The method I use is called "HAAR cascade". My tracker uses a trained file found online and it works well. If you are not happy with it, you can train it yourself, as many tutorials can be found on internet.

I could not find much options on detection, but many trackers exist. One of the most efficient I could find is called MOSSE. Trackers differentiate by the way they analyze frames to follow an object. The most common defects are slowness (some are no better than 1 fps), loss of track if the subject is too fast and sensitivity to light illumination (in front of a window making a lot of light, it won't be able to follow).

3.3.2 Limitations

I am satisfied with my HAAR trained MOSSE tracker but I believe that the tracking part will be included in the research of the intern following me that will be about activity recognition, or at least it will be easy to replace it. Therefore I made a very easy tracker class in my code that should be easy to overwrite that will allow the motor control process to work just fine even though the tracker works differently.

3.4 Motor command

The objective here is to follow one user if it is going to get out of the camera's view. If two or more users are in view, the idea is to maximize the number of users we can keep inside the camera's view. In this portion I will only follow the users horizontally (along the rail) but the exact same principle can be applied for vertical movements. But I guess that from the ceiling, at the right angle, the camera does not have to tilt if the user goes backward or forward.

3.4.1 The principle

First of all, I defined margins (for example 10 percents of the camera's output width, in red in picture) on the left and right of the camera view because the tracker (or camera rather) loses quality on image borders, and we need some time to react if a user suddenly decides to get out of the camera's view.

Figure 3.1: Margins in camera view

Then, I decided to use only relative instructions to get rid of increments and so on : if the user goes to the right (black cross in next picture), I just ask the robot to go to the right (red arrow, moves the camera view) until the user is in the center of the view. This principle allows me not to process anything in order to get a goal position, but I have to make a good stopping condition : when the motor moves, the tracker gets less reliable, therefore we have to be careful because the tracker may lost track of the user, and never stop if the condition is to get the user to the center of the view.

3.4.1.1 Case with only one user

The idea is simple and allows the code to remain so. If the user goes too much on the right (too much being out of the margins), the robot goes to the right at a said speed,

and stops when the user has reached the center of the camera view.

Figure 3.2: Motor moves if the user is off the margins

3.4.1.2 Case with two or more users

The idea is to go back to the previous case : find the centroid of all the users we want to keep in view, and apply the same idea as before but with this center, as one single user.

In practice, after the tracker has found all of the users, we want to know who are the 2 most spaced users that can fit within the camera view (minus the margins). If one of them is off-margins, we ask the robot to move, until the center of those 2 persons reaches the center of the camera view.

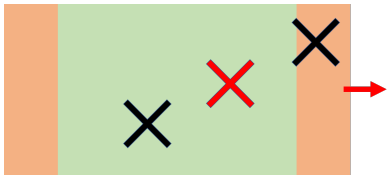

Figure 3.3: Motor moves if one user is off margins, and we find the centroid of all users

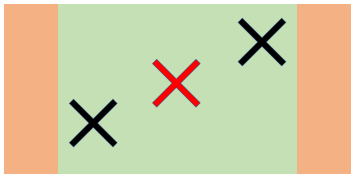

Figure 3.4: Centroid (red cross) is in the center, and users are inside margins

3.4.2 Issues and solutions

As mentionned above, the tracker loses reliability when the robot (the camera) is moving. Therefore, the condition to stop, when the user reaches the center of the view, may not be reached if the tracker loses track of the user.

The first thing I did implement to address this issue was to take information from the tracker when the motor is moving only if it has found the exact same number of persons than before moving. It is a condition to ensure that the tracker is quite right, but it still has some limitation.

For example, if someone was detected to the extreme left so that it could not fit in the margins, the motor would move without wanting to keep this person in view. The

motor will begin to move and the tracker will lose track of this one, but it used to be one more counted user. During its move, the tracker will find 1 person less and the condition described before will prevent the tracker from being listened to.

The last thing I did, to prevent any unforeseen error, was to implement a timer, to limit the time during which the motor could run without stopping. It is just an arbitrary time (about 3 to 5 seconds). If the tracker does not find all users it detected before moving, not a single time, during this amount of time, we stop the motor.

3.4.3 Limitations

This code only follows the user(s) along one single axis (the rail). It would be very easy to duplicate those ideas to enable the following of users vertically, using the servo motor.

Here, the motor always stays on one side of the room, it will always be able to find, track and follow users while remaining on only one side of the room. It is not really important for now but when we will be looking for surfaces to project on, it might be of interest to project on the wall the user is looking at (not in his back), and therefore to go to the opposite side of the room, without losing track of the user.

3.5 Finding flat surfaces

To be able to locate the biggest - almost - flat surface in view is vital if we want to project readable data for the user(s).

3.5.1 The Principle

Here I have two entries : one depth image and one RGB image and the aim is to find the biggest and most suitable surface available for projection.

Certainly, the best result would be obtained by combining the data from the two images. But here comes a bit of trouble : with my new camera, the Intel RS200, the depth camera is of pretty poor quality. It is supposed to have an operating range between 0.5m and 3.5m but, for mine at least, the range is between 0.5m and 1m, making it not really fit to use in a room to find surfaces... Anyway, I will begin by working on my RGB data, and I believe it is better like this, as we will see.

To begin with, since image treatment can be quite complex and take a lot of time to the computer, I will give a particular attention to keep my code very simple in order to minimize the time spent on finding a surface, time during which we could lose track of a user for example. The idea is to look for an adequate surface every given amount of time (5 seconds for example) and then to go back to tracking and moving.

3.5.2 Detouring

OpenCV has a very handy function in image treatment named Canny(). It allows to get a black image with, in white, the contours of objects in the image. So, every given amount of time, I first use the Canny() operator with the values I determined before in

order to get the contours. The threshold value allows to set from how much of a contrast two colours should be told to belong to different objects. It is pretty accurate and draws contours around the surfaces that are fitted to be projected on, without any contour inside.

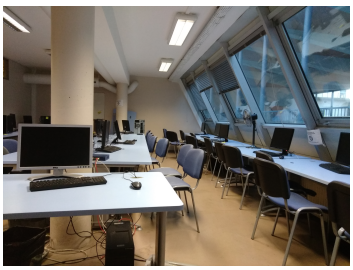

Figure 3.5: Frame from camera stream

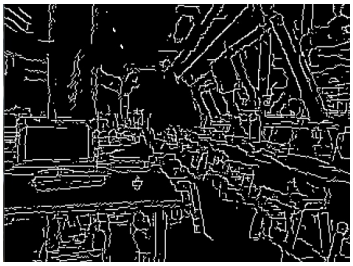

Figure 3.6: Result after Canny() function

Therefore, I have a black image with white contours and not a single white dot inside of large surfaces that could be good to project on. I now only have to find the biggest surface. In the end, the Canny() function deals with the part of choosing to what extent a surface is considered flat : is a pencil on a table a problem ? Is a computer on a table a problem ?

3.5.3 Find the biggest surface

Now that the image is well detoured, I have to look for the biggest surface to project on. But what does the biggest surface mean ? It definitely is a matter of biggest area, but also a matter of compactness : we wish not to project on some very linear shape, the closer we are to a square, the better. Finally, one last issue stands in the way : if the surface is not perpendicular to the projector, the camera won't see a rectangle but a trapezoid, and therefore finding the biggest surface includes looking for the biggest trapezoid too.

3.5.3.1 The idea of the cross

First, since most of what the camera sees is in front of it is not tilted (due to its position on the rail), I decided to begin by only looking for squares (and not trapezoids), that are not inclined (to begin with something simplified and see how it goes).
I have a black image with white contours and I want to find the biggest square that can fit. Instead of trying to fit a square around every black dot and keep the biggest, what

would be very long in terms of operations, I decided to look for the biggest cross I could make on every black dot. To do that, I begin from a dot, find how many consecutive black dots I can find up, then down, then to the left and then to the right. The biggest cross is the one whose sum of those consecutive dots is highest.

Figure 3.7: Biggest cross found in red

Because the image I work on has a lot more black dots than white ones (after detouring), I decided to begin by working on those white dots, in order to minimize the number of operations. The idea is still the same : finding the biggest cross, but I try to find it after having found a white dot (which corresponds to an edge). Once I have a white dot, I look to the right, for how many consecutive black dots there are. Then, for every one of those consecutive black dots, I look vertically for black dots, thus making my cross again. I keep for this cross the maximum number of vertical consecutive black dots.

There is a little tricky part : I always begin by looking to the right of a contour, because it would be redundant to go to the left since the previous contour, to the left of this one, will end up looking at the same pixels. But if no contours are detected to the left, then the left pixels won't be examined. Therefore, we have to add a condition to also begin the examination to the extreme left column of the image.
Because with such a method I could end up finding a very linear form as my "biggest area", I try to make a square of given size around a point that has just be found as the center of the new biggest cross. If it fits, I keep it, if not, I keep the previous point. After some tests, it seems to be very reliable as a mean to find the areas of interest for projection, even when I tilt the camera in a completely random position. I am therefore very happy with this solution as it is very simple, quick and effective, and is very reliable.

3.5.3.2 Grow into the biggest rectangle

After some experiments, I found out that this technique was very good at finding the areas that look like the best place to project on, and very fast for the computer. I now have one point (the center of the cross), that I know is inside the area on which I wish to project a display as big as possible. I now want to find the biggest rectangle I can fit in (for simplification purpose : I should look for trapezoids, but we will see that non tilted rectangles work just fine). My first idea was to begin from this point, to begin by growing equally on each side into a square until I reach a white dot, then to try to grow into a square but only 2 sides at a time, and finally to grow one side at a time into the biggest rectangle possible.

The problem could be that with my previous cross, I am not sure that my point belongs

Figure 3.8: Biggest square centered on cross's center

Figure 3.9: Biggest 2 side extension from previous square

Figure 3.10: Biggest 1 side extension into a rectangle

to the biggest square I could draw, but after all the experiments I made in rooms, it looks like the particular shapes that could lead to such mistakes are not found commonly in rooms (and kept away by the detouring function), therefore there is no need to go further since the method I am using works very well, and pretty quickly.
I am now able to find the biggest rectangle on the biggest area of interest for projection.

3.5.4 Limitations

The problem I get most of the time is the following : if, while looking at the RGB camera output, two surfaces look almost the same size, where a human would just make a choice, the code may alternate between one and the other. To avoid this, I made some statistics : I make an average of where the area of interest is the most and then project on it. Therefore it won't alternate if at few times it finds it elsewhere, or only at the beginning, when, after some loops, I reset the average.

Another big issue is perspective ; I only look for the biggest surface as the camera sees it, but if a surface is farther than another, it can appear smaller to the camera. Therefore, we have to use both depth and RGB stream in order to determine which surface is truly the biggest.

The final step before being able to project a screen on the found surface is to be able

to project nicely on a surface that may be tilted, because a tilted surface ends up with a distorted display.

The first approach could be to get its tilt with the depth camera : determine the angle, modify the image and project in order to have a non distorted result.

A second approach is to first project without any correction, observe how distorted it is and correct it, in a closed loop. It could be a better way since this is the principle used in most video projectors that auto adjust.

4 Continuation

This chapter is meant to help students that will follow my research to better understand my code and how / why it evolved, and to give them ideas for improvement.

4.1 About the code

4.1.1 Ordering the classes

I first wanted to divide my code in a few classes : one for the camera, one for the tracker, one for elaborating the motor command, one for the motor and one for the display.

I encountered a few issues though. I could not manage to make an independent class for the camera. Even though I don't claim to be very experienced in coding, I don't think that it could be done : the error I got was quite rare and after my research, it seems like it misses some function in the official library from Intel, that won't make it work through the headers system in C++. In other words, it forced me to keep everything related to making the camera work in the main.cpp file, what did confuse my code a bit. My successor may be able to solve it, or switching for another camera will probably fix it.

I had a quite similar issue for the last tracker I used : I couldn't put it as I wanted in its class. This is why in the main.cpp file are some things quite weird. For example, the vector containing the coordinates of the tracked users should stay in the Tracker class, but I could not manage to make it work, I had some errors no one knew about. I therefore declared it in main.cpp as a workaround. As I guess someone will be replacing the current tracker, he should try to get rid of these small tricks and make a working Tracker class.

4.1.2 Dealing with a choice to do between equal surfaces

It may happen that in the camera view are two surfaces that could both be as good for projection. In such a situation, a human would just make a choice and keep it, but the computer here could switch for one another each time it looks again for a surface. To improve it, one may want to make some statistics to keep as a choice where the algorithm chooses to focus most of the time. But what I guess is better is to store all the surfaces we found as good for projection, because when trying to help a user in its task, we may wish to project in the direction he is looking at, and therefore switching to another surface, that we would already know.

4.2 Ideas for improvement

4.2.1 Project on a surface the user is looking at

For now, my code is able to find a surface to project on, but it might be in the back of the user, therefore we could want to go to the opposite side of the room to be able to project where the user is looking at. What is difficult in the process of going all around the room is that we are very likely to lose track of the user. We could maybe elaborate the command before moving, let go the robot and hope that the user does not move too much before the time that it reaches its destination.

4.2.2 Loss of tracker accuracy while moving

One thing we observe while testing the robot is that most of the time, it stops moving not because its target has reached the right location but because of the timer, that is here to stop the motor if the tracker loses track of the user.

To improve this, the first thing is to increase the motor's speed step by step, rather than sending a crenel. It will reduce the vibrations of the robot and therefore improve the tracker's accuracy.

4.2.3 Take perspective into account for surface finding

The issue discussed here is that a big surface that could be fitted for projection, if far away from the camera, is seen a lot smaller than closer ones can be. Therefore, the ideal would be to have depth data over the surfaces we are looking at and make the comparison of sizes while taking the distance into account, thus comparing true length rather than seen lengths.

4.2.4 Project a non distorted image

When it comes to the part of projection, what we have is only a rectangle. It is very likely that the surface it corresponds to is tilted, resulting in a distorted image if we try to project as it is. The easiest way is probably to project, analyse through the camera the shape it has, and adjust, or to directly use depth data to determine the surface's tilt.

5 Conclusion

This internship allowed me to take part in a very interesting and challenging project : building a completely autonomous robot able to follow users, understand their doings and help them in their tasks. It was really nice to work at the Institute of Intelligent Robotic Systems. It has given me many knowledge on coding and project management.

I did my best to begin the software editing of the ceilibot. I am quite proud of some results I had. I leave some ideas of improvement in this document for my successors, and I hope this project will continue to go forward and one day be completed.