Hendrikse Jérémy

Session report 18

For this report I won't do it chronologically because it would involve way to much steps and would be way to complicated to describe because I will cover approximately 40 hours of work.

During the last hours I've been working on the stereo camera, this objective with this is two simultaneously detect objects with the camera and have they're position relative to the robot, the camera will also enable us read QR codes and associate them to the object they belong.

Acquiring camera stream

The first step was to acquire an image stream from the camera using python. I found code on the internet to receive images. We first define a camera object and then we can read an image from the camera when we need it, and we are able to process the information in the image.

Because this was a stereo camera it is connected by a single usb and so we receive the frames from both cameras simultaneously in one single frame that we need to crop to separate both frames and obtain left and right frame.

Also, during this step, I had to find out the best imageSize I could get, and it turned out to be 1920*1080. I also had to become a little bit familiar with the OpenCV libraire I was using.

Calibrating intrinsic and extrinsic camera parameters

The next step after receiving the Stereo camera information was to calibrate it.

First we need to calibrate the cameras independently to obtain the distortion coefficients and camera matrix. To do so we use the OpenCV library and chessboard pictures that we took with the camera. At least 50 pictures for each camera are required, the chessboard must be completely visible in each picture and at different position and orientation in each of them. The libraire is then going to try finding the corners for each frame and draw them.

At this step the calibrateCamera() method from OpenCV returns the camera intrinsic parameters: distortion coefficient, matrix, RMSE value… . The first two will be used to further calibrate the cameras together. The RMSE (root mean squared errors) must be under 1px for a good calibration but up to 3 can be tolerable. (I obtained ~2.2 for each camera which can be the source of my biggest problem bap disparity map, more on that later).

The next step is to find the extrinsic parameters (parameters connecting both cameras). This step is pretty similar to the first one, except it now takes at least 50 pictures of each cameras taken simultaneously and returns a lot of information's but the most important are: RMSE value, rotation matrix and translation vector. After this step I've got a decent and appropriate translation vector and rotation matrix but a RMSE value of 42px. (to solve this problem, I will try to exclude the pictures that have the highest RMSE value or try a calibration using circle grid patterns).

At the end of this step, I wrote a code to show the differences between left and right frames received from the camera and the rectified version. Normally when close to straight lines, before we see distortion (lines are curved) and after there is no more distortion. Also, the rotation and translation matrix should make the two cameras have the same plane, this point is REALLY important to have a decent disparity map. (this was one of my biggest problem and I think it is pretty much good).

Getting disparity map

The last step is to get the disparity map showing the differences in depth this part is the most difficult one. To do so the OpenCV libraire uses a block matching algorithm (BM for short) this algorithms has to be tuned, trying to find the optimal parameters (there are 11 and most have values going from 0 to 100). This is the part where I still have trouble.

Problems encountered.

I've had a lot, really a lot of problems. First understanding the OpenCV libraire (really big librarie with condensed documentation).

Understanding what camera calibration, disparity maps etc were.

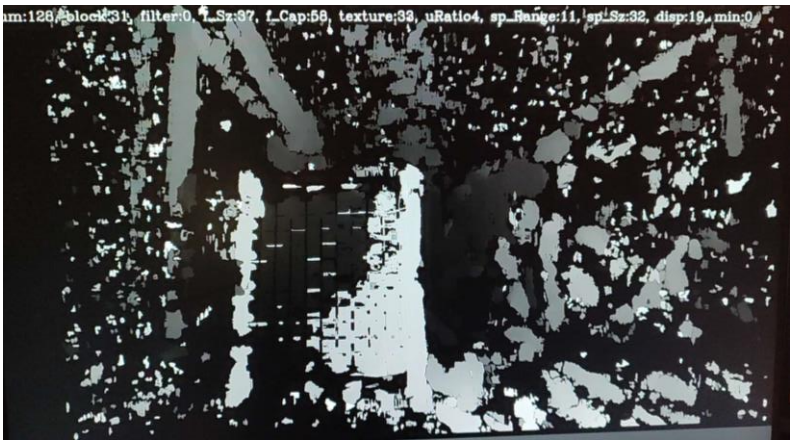
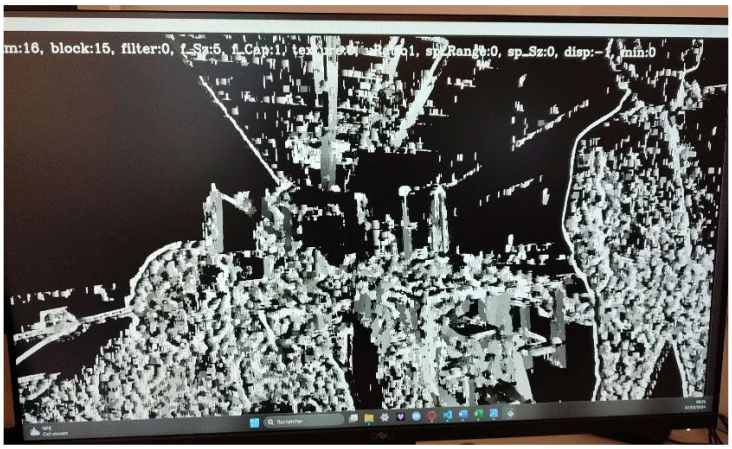
Storing numpy arrays and disparity maps (5 dimensions images).

Finding out from where the problems came because I didn't understand quite well what everything did.

BM parameters different form frame to frame and live.

And a lot of random problems.

Pictures

Disparity map	
	

Rectified pictures (top original, bottom rectified)