TER Report - Interior 3D detection in a cobotics context

BIZARD Astor
·
Supervised by : Olivier Aycard · Jerôme Maisonnasse

June 2017

Acknowledgement This report is the result of a TER research internship at Université Grenoble Alpes, Grenoble, France.

Abstract We present an approach to compute 3D data in a real-time security context to evaluate distance between humans and robots. The implementation is oriented to be compatible with the ROS platform, and overcome drawbacks of low-quality depth cameras, such as important noise. It is meant to function without the robot broadcasting its position, therefore only by perception.

Keywords 3D Depth Sensor · RGB-D Perception · Collision Avoidance · ROS · Physical Human-Robot Interaction

1 Introduction

The problem of collaboration between humans and industrial robots - also known as cobotics - induces security issues : how to ensure the security of the human, without limiting the possibilities of the collaboration ? We thus need to find a way to do this without preventing the humans to approach the robot. The work must therefore be done on limiting the robot moves to a secure zone, calculated knowing the position of the persons. This limit must be computed in reasonable time to ensure security.

2 A static vision approach

We need a way to perceive the 3D environment in which the robot moves. Using 3D depth sensors seems a pretty good way to get an estimation of a part of this environment.

2.1 Where to place the sensors

2.1.1 On the robot itself

In many situations where robots need to perceive their environment (using a perception-decision-action scheme), the sensors are intuitively placed on the robot, as a human got his eyes and ears on himself to perceive its own environment. But there are some robots that cannot handle sensors on their bodies : for example industrial arms that move in every direction cannot give a reliable perception (cf Fig. 1). Additionally, we want our architecture to work with the robot not broadcasting its position or movements, so we wouldn't be able to correct sensor data with its movements.

Fig. 1: A typical application of cobotics with an industrial robotic arm.

2.1.2 Static sensors

A different and more reliable way to get informations is to place static sensors. The advantage of this method is that we can use several sensors to cover a wider range in the room, and that without obstructing the robot's or human's movement. It however requires the presence of several sensors, as objects or persons could possibly hide the robot and compromise the availability of reliable information. In our case, this it what we will be using.

2.2 The ROS API

One of the most used API in the world of robotics is the ROS platform. It works as a set of executables, named nodes, which communicate with messages published on topics. The advantage of this platform is that it provides a standardized API for communicating with autonomous robots. In our case, we present a solution with two nodes, added to the sensor node : the first one will be charged to analyze the data provided by the sensor, identifying moving objects and persons ; the second one will check - from the information given by the first node - that the human security is not compromised. Person detection and tracking is a common problem, however the only public ROS node which provided person tracking for the model of camera we are using is not maintained anymore for the newest versions of the platform.

2.3 Sensor data

We are using low-cost RGB-D cameras, with the OpenNI standard - we are working with the ASUS XTion PRO LIVE model. We are using the openni_camera node of ROS [1] to get data as a ROS message. We will be using only depth data for the moment.

2.3.1 Depth space

A depth camera provides data in an implicit depth space (cf Fig. 2) : a distance is given for each pixel of the image. Posing d as the distance at the pixel (i,j), we get the point (i,j,d) in the depth space with the camera as the origin.

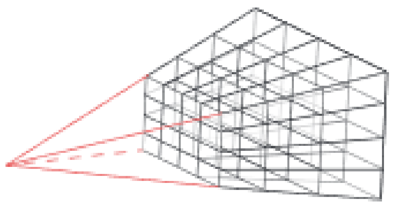


Fig. 2: Representation of the Depth space [2].

3 Data processing

3.1 Data processing pipeline

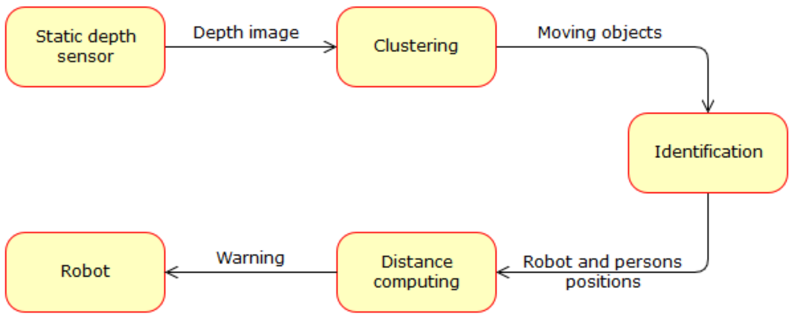


Fig. 3: The data processing pipeline.

3.2 Detection of moving objects and persons

To analyze the data provided by the sensor, we want first want to cut the image in two parts : the static objects (such as walls) and the dynamic, moving ones (cf Fig. 4a, 4b, 4c). Existing studies already present person detection with RGB-D sensors, but with some limits, as sometimes persons won't be detected and on the other side static objects such as furniture will be identified as persons. This kind of problem could come from the fact that the camera could be moving. In our case, we prefer static cameras along with high reliability detection, as we don't want to compromise security, as well as we don't want to stop the robot when it is too close to the table it is placed on. The static nature of the cameras allows to first store a reference background of the field of view when we start ; it requires the room to be empty of persons or robots. We then compute on each frame the differences between the frame and the background to get the dynamic objects.

3.3 Clustering

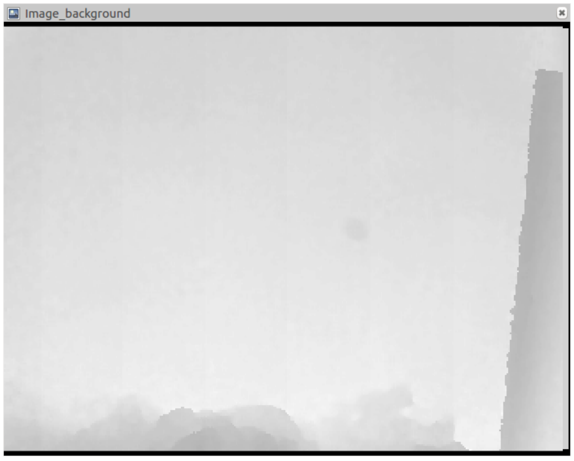
Afterwards, we want to separate the several objects we identified : it is known as clustering - each object being a cluster (cf Algo. 1). The main challenge of this algorithm is to provide real-time reliable data ; the longest part is the $FuseTopAndLeftClusters()$ function, as the two clusters can be pretty big : a naive approach of re-computing the cluster for the whole image is not possible to stay above 1 frame per second.

Algorithm 1 3D Clustering : Compute the list of the clusters in the depth image

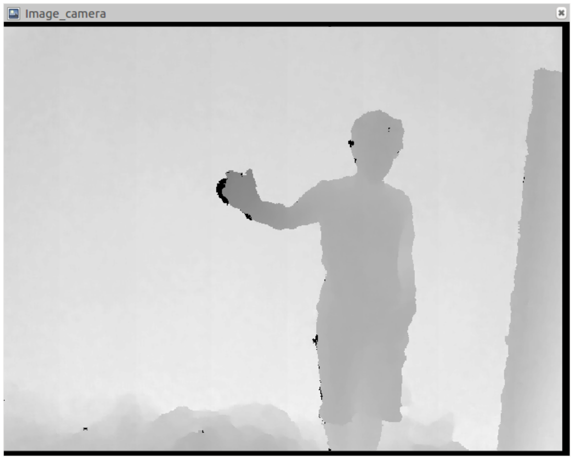
```
Require: $img[h][w]$ // the depth image
  $cluster[h][w]$ // a tab to store the cluster of each pixel

  // Start with the top-left corner
  if $isDynamic(img[0][0])$ then
    $cluster[0][0] \leftarrow NewCluster$
  else
    // This pixel belongs to the background
    $cluster[0][0] \leftarrow Background$
  end if

  // Compute the left column
  for $i : 1 \rightarrow h - 1$ do
    if $isDynamic(img[i][0])$ then
      if $diff(img[i-1][0], img[i][0]) < Threshold_{Depth}$ then
        // This pixel and the previous have similar depths : they are close and belong to the same object
        $cluster[i][0] \leftarrow cluster[i-1][0]$
      else
        // This pixel and the previous one have different depths : they belong to different objects
        $cluster[i][0] \leftarrow NewCluster$
      end if
    else
      $cluster[i][0] \leftarrow Background$
    end if
  end for

  // Do the same for the upper line ($j : 1 \rightarrow w - 1$)
  // ...

  // Compute the rest of the image
  for $i : 1 \rightarrow h - 1$ do
    for $j : 1 \rightarrow w - 1$ do
      if $isDynamic(img[i][j])$ then
        $isNextToTopPixel \leftarrow diff(img[i-1][j], img[i][j]) < Threshold_{Depth}$
        $isNextToLeftPixel \leftarrow diff(img[i][j-1], img[i][j]) < Threshold_{Depth}$
        if $isNextToTopPixel$ and $isNextToLeftPixel$ and Top and Left clusters are not already the same then
          // This pixel connects two objects
          $FuseTopAndLeftClusters()$
        else if $isNextToTopPixel$ then
          $cluster[i][0] \leftarrow cluster[i-1][0]$
        else if $isNextToLeftPixel$ then
          $cluster[0][j] \leftarrow cluster[0][j-1]$
        else
          $cluster[i][0] \leftarrow NewCluster$
        end if
      else
        $cluster[i][j] \leftarrow Background$
      end if
    end for
  end for
```

3.3.1 Noise minimization

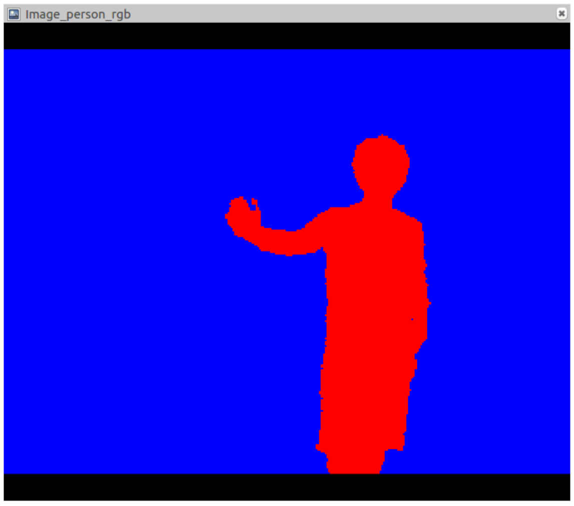
The cameras we are using are cheap, but it indeed has drawbacks, such as important noise (cf Fig. 5a) : these cameras use infra-red projection to get the depth data - cheaper than laser but less effective. We get important noise on these images (the black dots on Fig. 5a), often on smooth or glossy surfaces. We need to get rid of those black dots, as they flicker on each frame, thus leading to identifying them as moving objects (cf Fig. 6). One known method of noise removal is to maintain a level on each pixel, increasing it when it is dynamic and decreasing it when static, to prevent single-frame noise. Our problem is that the noise we get flickers but stays at the same position, making this method irrelevant. We therefore chose to simply remove the black dots - which give a depth value of 0 -, (as it represents around 95% of the noise) from the background reference (by simply replacing them by the nearest non-zero value). We then consider all the zero depth values as static objects (cf Fig. 5b). However, there is still noise giving non-zero depth values, especially on far away surfaces. We chose to limit the zone of interest of the view to within 4 or 5 meters of the camera to avoid these problems.



(a) The background stored at the beginning.

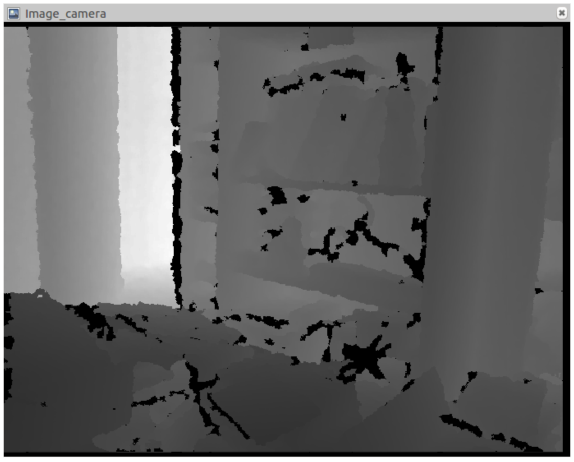


(b) An image recorded while a person was in the field of view.

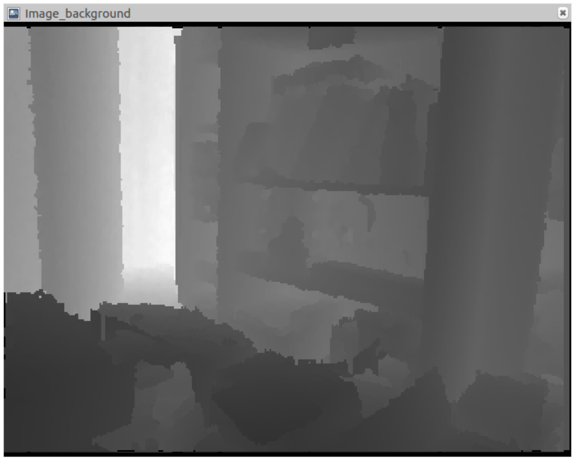


(c) The separation between static objects (in blue) and moving ones (in red).

Fig. 4: A simple clustering example.



(a) The depth image, as received by the camera : dark points are near the camera, bright ones are far.



(b) The same depth image after noise treatment.

Fig. 5: Noise suppression.

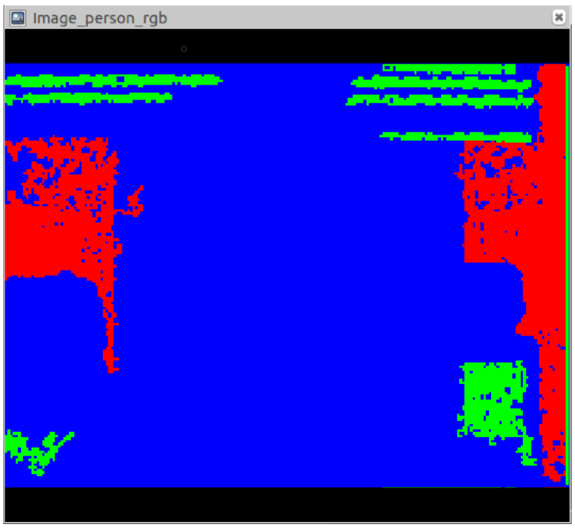


Fig. 6: Important noise interpreted as moving objects (in red and green).

3.4 Identification

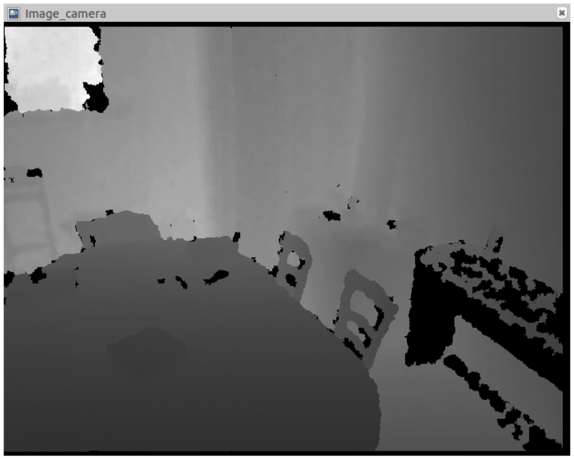
Once we separated the several objects in the image, we need to identify persons and robots. In our case persons are identified only with their size - if it is a moving object, around 1m75 tall, and around 50cm wide, it must be a person. This is to be adapted with the situation : if a robot is 1m75 tall, or manipulates objects of this size, we need another method of identification (cf Section 4). The identification of the robot induce the same kind of problems and has not been resolved during the internship.

3.5 Distance computing

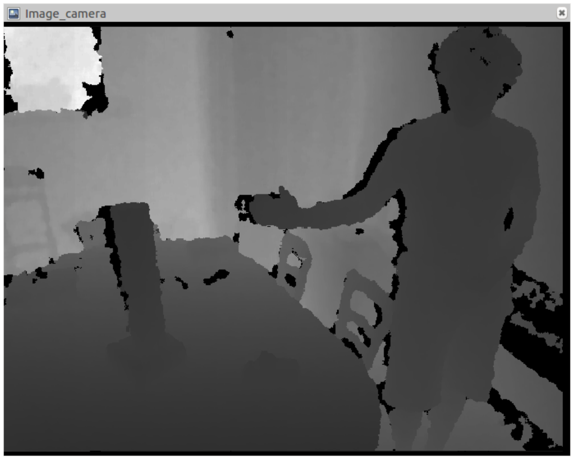
Afterwards, we need to compute the distance between each person and the robot. We are receiving cluster data in the depth space, however distance computing is far easier in a cartesian space (cf Fig. 8), thus we transpose coordinates in this space. The transposition of a point P_d in the depth space to a point P_c in the cartesian space is given by

$$P_{cx} = \frac{(P_{dx} - c_x)P_{dz}}{fs_x}, P_{cy} = \frac{(P_{dy} - c_y)P_{dz}}{fs_y}, P_{cz} = P_{dz} \text{ [2]}$$

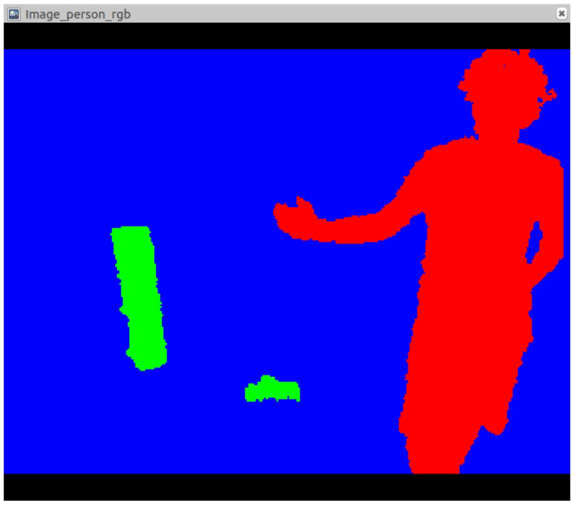
where f, s_x, s_y, c_x, c_y are calibration data from the depth sensor (focal length, size of a pixel and coordinates of the focal axis in the image plane) [2]. All it is left to do is to compute the distance between each point of the person and each point of the robot. However, this method leads to very important treatment time (30 seconds per frame when one person is in the field of view). We thus need to reduce the number of points to compute. One way is too first reduce the size of the image, by sampling it - but it could have consequences on precision. An other and more widely used way is to reduce the number of points of the robot to only a sample of points of interest - for example one point at each articulation of the robot. As we know the shape of the robot, we can choose these points in order to ensure the same security. If a person is too close from one of these points, a message is sent to the robot to stop.



(a) The background stored at the beginning.



(b) An image recorded while a person, the robot and an object were in the field of view.



(c) The separation between static objects (in blue), moving ones (in green), and persons (in red).

Fig. 7: A simple identification example.

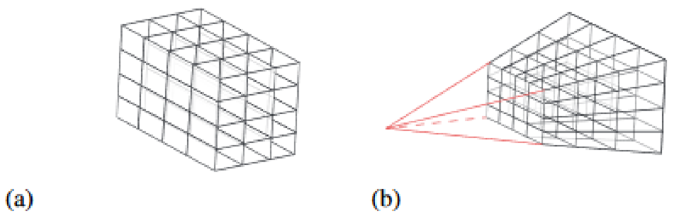


Fig. 8: Representation of Cartesian space (a) and Depth space (b) [2].

4 Conclusion and future work perspectives

To conclude, we have a full prototype of data-processing, working in small interior pieces - the quality of the sensors restricting its size to around 5x5 meters. Model-based identification should be the main improvement to bring to the work done in the internship. It can be done with only depth data (by expliciting the depth grid) or along with the RGB image - however, correspondence between RGB and depth data is reliable only in a range between 1.5 and 4 meters from the camera. This kind of identification will lead to easier sampling of the points of the robot - and eventually the persons, as we may know the global shape and the articulations of a human. We could also think about fusing sensor data (cf Article [2]), though it might not be necessary in our context, as independent sensors should be enough to detect if a person is too close.

References

1. ROS OpenNi node for the Kinect : http://wiki.ros.org/openni_camera
2. F. Flacco and A. De Luca, Real-time computation of distance to dynamic obstacles with multiple depth sensors, IEEE Robotics and automation Letters (2016). http://comanoid.cnrs.fr/publications_files/2017/IEEE_Robotics_and_Automation_Letters/RAL2016_MultiDepth_preprint.pdf
3. F. Flacco, T. Kröger, A. De Luca and O. Khatib, A depth space approach for evaluating distance to objects, J. of Intelligent and Robotic Systems, vol. 80 (suppl. 1), pp 7-22 (2015). http://www.dis.uniroma1.it/~labrob/pub/papers/JINT14_DepthSpace_final.pdf
4. F. Flacco, T. Kröger, A. De Luca and O. Khatib, A depth space approach to human-robot collision avoidance, Proc. IEEE Int. Conf. on Robotics and Automation, pp. 2353-2359 (2013). https://cs.stanford.edu/group/manips/publications/pdfs/Flacco_2012_ICRA.pdf
5. P. Jia, S. Ioan, C. Sachin and M. Dinesh, Real-time collision detection and distance computation on point cloud sensor data, Proc. IEEE Int. Conf. on Robotics and Automation, pp. 3593–3599 (2013). http://ioan.sucan.ro/files/pubs/realtime_cd_sensor_data_icra_2013.pdf
6. C. Morato, K.N. Kaipa, B. Zhao and S.K. Gupta, Toward Safe Human Robot Collaboration by using Multiple Kinects based Real-time Human Tracking, Journal of Computing and Information Science in Engineering (2014). http://www.enme.umd.edu/~skgupta/Publication/JCISE2013_Morato_draft.pdf
7. N. Najmaei and M. Kermani, Prediction-based reactive control strategy for human-robot interactions, IEEE Int. Conf. on Robotics and Automation (ICRA), pp. 3434–3439 (2010). https://pdfs.semanticscholar.org/19b6/b377548396e47c90d5c5ea6630a265016a22.pdf
8. Z. Zhang, Microsoft Kinect sensor and its effect, IEEE MultiMedia, vol. 19, no. 2, pp. 4–10 (2012). http://ieeexplore.ieee.org/document/6190806
9. T. Gecks and D. Henrich, Human-robot cooperation : Safe pick-and-place operations, Proc. IEEE Int. Works on Robot and Human Interactive Communication, pp 549-554 (2005). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.856&rep=rep1&type=pdf