COBEM2015

23rd ABCM International Congress of Mechanical Engineering
December 6-11, 2015, Rio de Janeiro, RJ, Brazil

3D edge detection aimed to the automatic programming of welding robots

Miguel Angel Villanueva Portela
Universidad Nacional de Colombia, Departamento de Ingeniería Mecánica y Mecatrónica – Ciudad Universitaria, Cra 30 #45-03, Edificio 407 oficina 201. Bogotá,11001000, Colombia
mavillanuevap@unal.edu.co

Ricardo Emiro Ramirez
Universidad Nacional de Colombia, Departamento de Ingeniería Mecánica y Mecatrónica – Ciudad Universitaria, Cra 30 #45-03, Edificio 407 oficina 201. Bogotá,11001000, Colombia
reramirezh@unal.edu.co

Abstract. *Programming a welding robot is a challenging and time-consuming task, different kinds of tools have been developed to facilitate this process. One of these new methods is called intuitive robotic welding programming, in this method the operator hand-guides the robot to the desired welding positions. Other methods use virtual user interfaces to do the same, teach the desired welding positions. This research aims to facilitate the programming process related to welding robots. In this stage, it is carried out through the automated record of position and orientation of straight edges belonging to an assembly that are going to be welded. This process is developed around the processing of a 3D point cloud. Based on the Random Sample Consensus (RANSAC) algorithm, statistical outliers removal and pass through filters and the function planeWithPlaneIntersection of the point cloud library (PCL). An algorithm to process a 3D point cloud was development. This processing algorithm takes as an input the 3D point cloud captured from the working assembly to identify and record the straight edges that will be welded. The target edges reside on T joins with square groove and/or in inside corner join with no preparation. The information from the straight edges is going to be implemented in the next stage of this research as an input of the path and trajectory planner. The proposed method in this work is able to detect straight edges of a working assembles which contain T joins with square groove and/or in inside corner join with no preparation that will be welded. The proposed algorithm was tested on metallic and non-metallic mock-ups. The process relies in artificial vision 3D sensor, in this case a Microsoft Kinect was implemented. The decision of employing mock-ups rather than real working assembles was made due to the low resolution the Kinect sensor has. The benefits of the proposed system span from the easy of use to the quick recognition of the geometric features required in the path and trajectory planning process related to welding robots.*

Keywords: *3D Vision, Welding Robot, Point Cloud Processing, Geometric Features identification*

1. INTRODUCTION

Since welding is one of the most fundamental manufacturing processes, is natural that companies and researchers develop new methods and tools to improve its productivity. One way to reach this goal is reducing the time spent in the programing of a welding robot. To do this it is necessary to develop a system that can generate the programing code required by a welding robot to carry out some specific tasks. Therefore the automatic edge detection is essential as it acquire the position and orientation of the edges that are going to be welded, and thus this information will be passed down to the path and trajectory planner.

The idea of edge detection has been worked by some researchers before, but their work was oriented in the use of 2D images, Chen *et al.* (2005) implemented sub-pixel edge detection and stereovision technology as a matching method based in the epipolar line to get the edge three-dimensional information. Dinham and Fang (2013) followed a similar methodology as Chen *et al.* (2005), but they achieved better results because their method is more effective when the images have low contrast between the edges and the weldment, and also improved the matching and triangulation methods. Pachidis and Lygouras (2007) put into practice a pseudo stereovision system (PSVS). They used a correspondence algorithm to rearrange pixels to calculate the final path points. Chongjian *et al.* (2007) paid attention to provide welding direction and determine the gab width between the work pieces and obtained the image by means of a CCD sensing system and to get rid of the unused arc light they develop a special filter technology that combines the following methods: Laplace enhance, edge detector and straight line fitting based on Hough transform. With this methodology they could found the two edges of the gap.

The previously mentioned works mainly used butt weld joins, nevertheless there were no method for fillet weld joins, which are more common. Dinham and Fang (2014) worked on this issue, proposing a method that implemented an adaptive growing line algorithm based on gray-scale pixel intensities for robust identification of weld joins, and also they used

a stereovision system to capture a pair of stereo image of the weld join.

This research focuses on the processing of tree-dimensional point clouds to automatically record the position and orientation of straight edges belonging to an assembly that are going to be welded. In the following sections the proposed algorithm is explained, straightaway the results that were found are discussed, and at the end of the paper are the conclusion and the corresponding references.

2. 3D EDGE DETECTION

The idea of edge detection have been treated mainly on 2D image, some research are given by Dinham and Fang (2013), Dinham and Fang (2014), Pachidis and Lygouras (2007). Nevertheless Bähnisch *et al.* (2009) developed and algorithm to detect 3D edges for surface reconstruction based in the Canny edge detection algorithm. This research uses the RANCAC paradigm which were introduced by Fischler and Bolles (1981), which is capable to fitting a model from a group of data with a significant percentage of gross error. The RANSAC algorithm is used to detect planes from point clouds that belong to a working mock-up, then the 3D edges are calculated with the intersections of these planes.

3. PROPOSED METHOD

The proposed method is around the processing of a 3D point cloud. Based on the Random Sample Consensus (RANSAC) algorithm, statistical outliers removal, pass through filters and the function *planeWithPlaneIntersection* of the point cloud library (PCL), thus an algorithm to process a 3D point cloud was developed. The designed algorithm is depicted in Fig. 1.

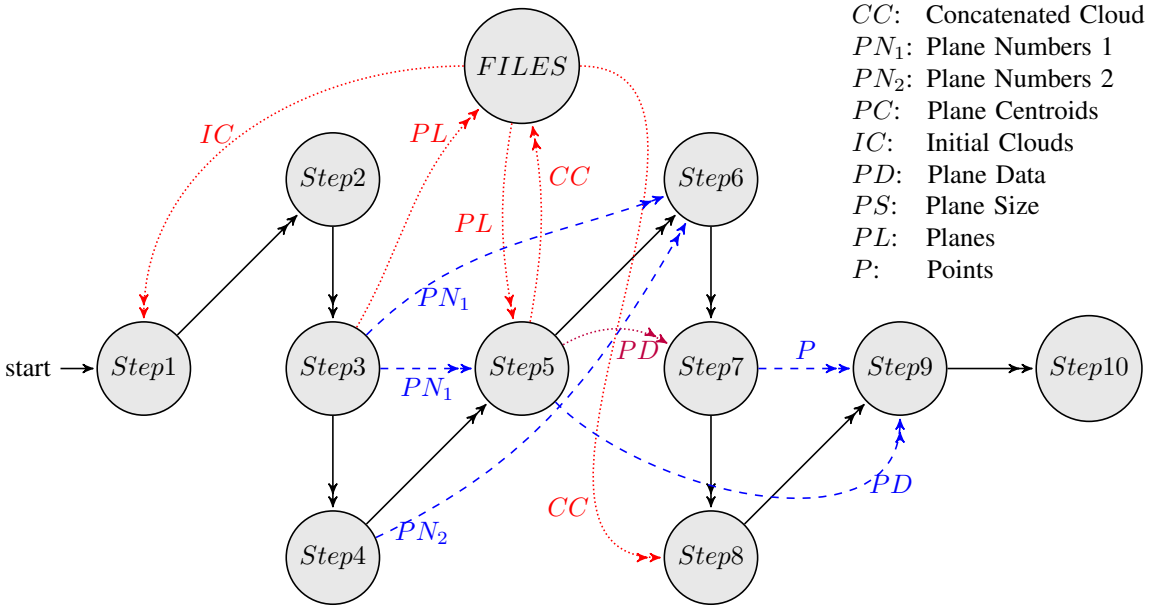


Figure 1. 3D edge detector algorithm diagram

3.1 Algorithm

Step1: In this first step the n point clouds that were taken previously are loaded, and all are concatenate into one single cloud CC_n.

$$CC_n = cloud_{(0)} + cloud_{(1)} + ... + cloud_{(n)} \tag{1}$$

Step2: A series of Pass Through filters *pcl::PassThrough<pcl::PointXYZ>* in X, Y, Z are applied to the concatenated cloud CC_n to eliminate points that do not belong to the work piece.

Step3: The RANSAC object *pcl::SACSegmentation<pcl::PointXYZ>* is created and applied to CC_n to estimate the plane parameters, thus obtain the mathematical model that is the plane equation Eq. (2) of all the i planes found, each time a plane in found is numbered and subjected to a statistical filter *pcl::StatisticalOutlierRemoval <pcl::PointXYZ>* to eliminate outliers, then a *PCD* file is saved with all the points that form this plane, also the plane data like size *PS*, number *PN* and its coefficients *PC* are saved to latter use in **Step4**, **Step5** and **Step6**. The previous procedure is a segmentation process and is carry out until the size of CC_n reach a minimum value.

$$A_i X + B_i Y + C_i Z + D_i = 0 \tag{2}$$

Step4: All parallel and concurrent planes must be identified to select among them the biggest one. Parallel planes are classified by Eq. (3), and the concurrence is defined by Eq. (4), where $P = \{p_1 + ... + p_j\}$ is the set of all parallel and concurrent planes, $\{k : 0 \leq k \leq j\}$ and (x_j, y_j, z_j) are coordinates of a point in the j-th plane. At the end the number that represent each one of the selected planes are stored to be used in **Step4** and **Step6**.

$$\frac{A_j X}{A_k X} \approx \frac{B_j Y}{B_k Y} \approx \frac{C_j Z}{C_k Z}, \qquad \text{where} \qquad k \neq j \tag{3}$$

$$D = \frac{A_k x_j + B_k y_j + C_k z_j + D_k}{\sqrt{A_k^2 + B_k^2 + C_k^2}}, \qquad \text{where} \qquad k \neq j \tag{4}$$

Step5: Some of the *PCD* files saved in the **Step3** are loaded, the selection criteria are the plane numbers stored in the last step. When each plane is loaded its geometric limits and centroid related to the Kinect sensor are determined, this information is stored for later use. A new point cloud CC is created with the loaded clouds by means of concatenation. Then, this new cloud CC is stored for latter use in the **Step8**.

Step6: The angle between the non-parallel planes are calculated by Eq. (5) to be used as a grouping criteria and the numbers of each couple of planes are stored for use in the **Step7**.

$$\cos\varphi = \frac{A_j A_k + B_j B_k + C_j C_k}{\sqrt{\left(A_j^2 + B_j^2 + C_j^2\right)\left(A_k^2 + B_k^2 + C_k^2\right)}}, \qquad \text{where} \qquad k \neq j \tag{5}$$

Step7: Taking the plane equation of the pair of planes stored in the last step, the line that is generated by their intersection is calculated, but this operation is limited just for adjacent planes, in order to distinguish two contiguous planes, the condition Eq. (6) must be *true*, where $P_c = \{p_{c1} + ... + p_{cw}\}$ is the set of all the pair of planes, one duple is defined as $\{p_r, p_{r+1}\}$, $\{r : 0 \leq r \leq w\}$ and $\bigvee$ is the or symbol.

$$DX_c > K_f\left(DX_r + DX_{r+1}\right) \bigvee DY_c > K_f\left(DY_r + DY_{r+1}\right) \bigvee DZ_c > K_f\left(DZ_r + DZ_{r+1}\right) \tag{6}$$

where,

$$\begin{aligned} DX_r &= \left\| Plane_{(r)_{MaxLimitX}} - Plane_{(r)_{MinLimitX}} \right\|, \\ DY_r &= \left\| Plane_{(r)_{MaxLimitY}} - Plane_{(r)_{MinLimitY}} \right\|, \text{The same way for } \ Plane_{(r+1)} \\ DZ_r &= \left\| Plane_{(r)_{MaxLimitZ}} - Plane_{(r)_{MinLimitZ}} \right\| \end{aligned} \tag{7}$$

and,

$$\begin{aligned} DX_c &= \left\| Plane_{(r)_{CentroidDistanceX}} - Plane_{(r+1)_{CentroidDistanceX}} \right\|, \\ DY_c &= \left\| Plane_{(r)_{CentroidDistanceY}} - Plane_{(r+1)_{CentroidDistanceY}} \right\|, \\ DZ_c &= \left\| Plane_{(r)_{CentroidDistanceZ}} - Plane_{(r+1)_{CentroidDistanceZ}} \right\| \end{aligned} \tag{8}$$

The K_f parameter is a constant, where $\{K_f : 0.55 \leq K_f \leq 0.7\}$, these values have been experimentally determined. All the lines calculated are numbered and stored.

Step8: The point cloud CC created in the **Step5** is loaded, and is taken as a base to build a *Kdtree* structure *pcl::KdTreeFLANN<pcl::PointXYZ>*.

Step9: The length of all the lines calculated earlier are fitted using Eq. (9) and the plane limits determined in **Step5**, the new lines are stored.

$$\frac{x - x_0}{x_0 - x_1} = \frac{y - y_0}{y_0 - y_1} = \frac{z - z_0}{z_0 - z_1} \tag{9}$$

Step10: Until this moment the lines are relative to the Kinect sensor coordinate system, in order to make them relative to the robot coordinates system a transformation is done by Eq. (10), where s is a scale factor, R is the rotation matrix, $\mathbf{T_o}$ is the translation vector, $\mathbf{C_k}$ is de coordinates vector relative to the Kinect sensor and $\mathbf{C_r}$ is the vector of transformed coordinates.

$$\mathbf{C_t} = sR\mathbf{C_k} + \mathbf{T_o} \tag{10}$$

3.2 The graphical user interface developed

One graphical user interface *GUI* was developed to accomplish two objectives, the first one was aimed to make a fast and easy way to test the 3D edge detector algorithm during its development, and the second one, as a tool to implement the proposed algorithm and provide the way to give the last line configuration. The second objective is the more important because not all the edges detected are weld joins or has to be completely welded, these criteria depend on the work piece design, and for the moment is under the designer control. The GUI writes a *txt* file, which's contain all the lines coordinate, the purpose of this file is to pass this information to the path and trajectory planer. In the Fig. 2 the GUI is presented and in Fig. 3 the tools are shown and their description is as follow: Fig. 3a the Pass Through filters tool, Fig. 3b Translation tool, Fig. 3c Rotation tool, Fig. 3d Tool to change the line length or to eliminate it, Fig. 3e Tool to trim lines.

The tools involved in the GUI development were **QT Creator 3.3.0** based on **QT 5.4.0 (GCC4.6.1, 64 bit)**, **C++** was the programming language implemented, the **openNI** framework were employed to provide the interface with the **Kinect sensor** and the **Point Cloud Library PCL 1.7** were used to capture and process the point clouds. The development was done under the operative system **Ubuntu 14.04.1 LTS trusty** on a 64 bits system.

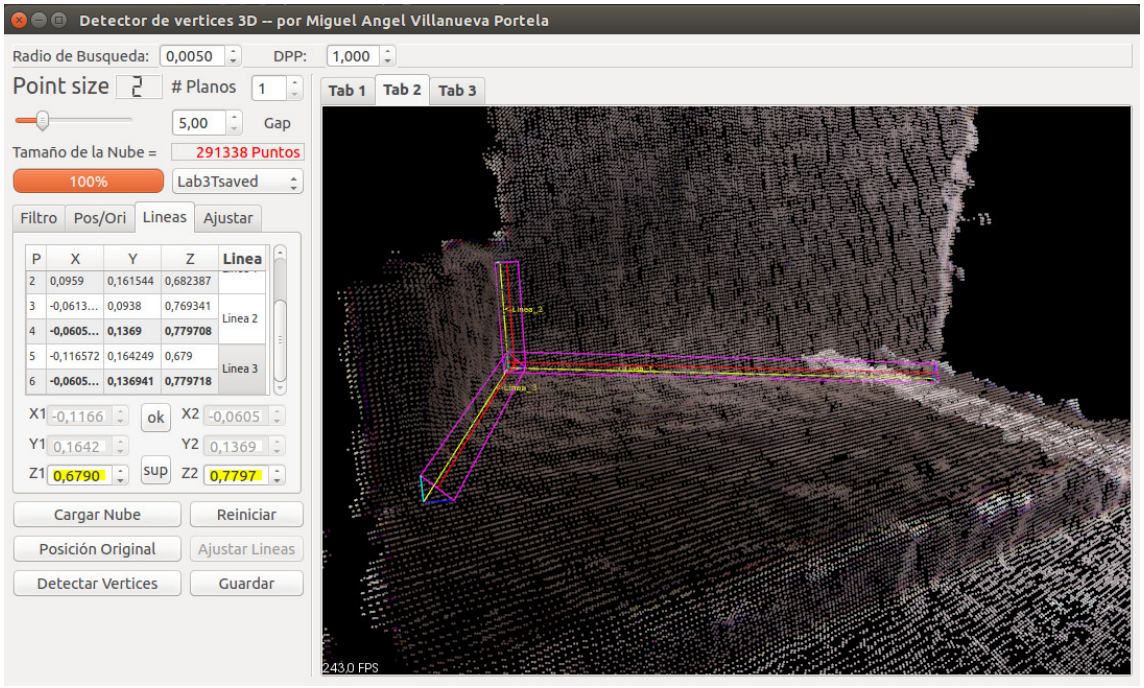
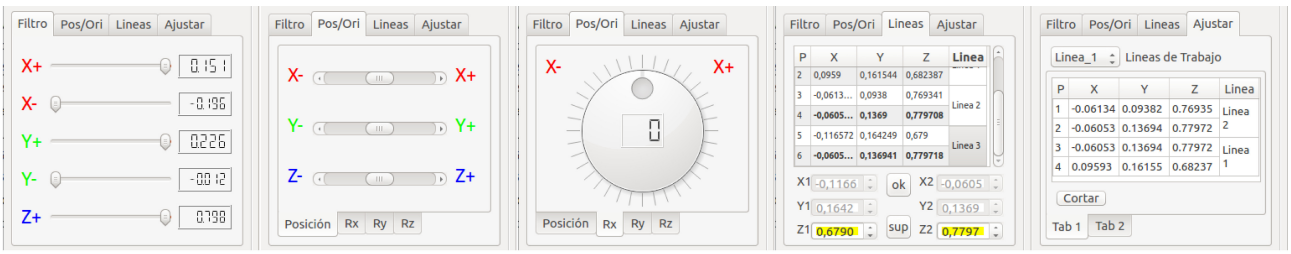


Figure 2. 3D edge detector GUI



a) b) c) d) e)

Figure 3. Set of implemented tools.

4. RESULTS

The system implemented to test the proposed algorithm can be observed in Fig. 4, it consists of one robot ABB-IRB-140 and one Microsoft Kinect sensor, plus some metallic and non-metallic mock-ups.

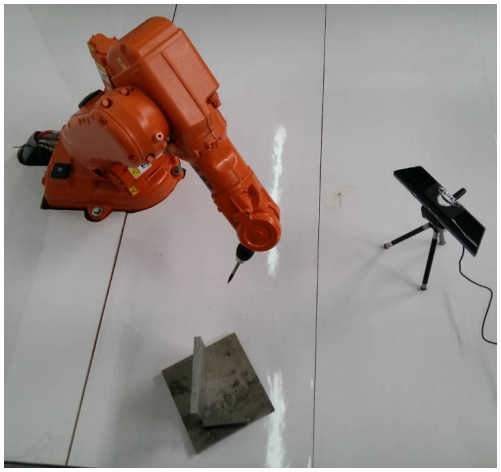


Figure 4. Robot ABB-IRB-140 and Kinect sensor

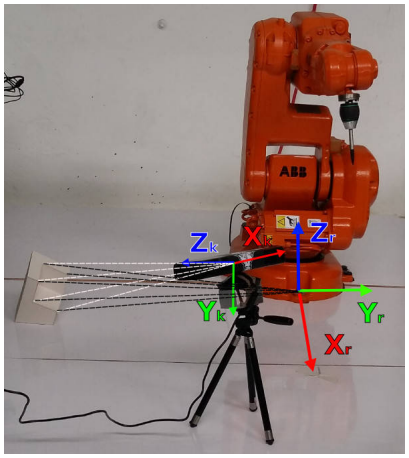


Figure 5. Coordinates systems

4.1 From Kinect coordinate system to robot coordinate system

As is observed in Fig. 2 the GUI provides the coordinates of the two points that define each line, these coordinates are relative to the robot coordinate system by means of Eq. (10), these systems are illustrated in Fig. 5 . The selected method to find the transformation was elaborated by Horn (1987), which uses unit quaternions. In this case, the Kinect sensor is always in the same position, thus the transformation just has to be calculated one time, unless the robot or the Kinect coordinate system changes. For this purpose a non metallic mock-up was made, it is shown in the Fig. 6, and seven control points were chosen on the mock-up, which were measured with one ABB robot IRB-140 and also with the proposed algorithm Fig. 7. The transformation is calculated in **GNU Octave 3.8.1**, the data are summarized in Tab. 1 and plotted in Fig. 8, the calculated quaternion $\mathbf{q_r}$ and his representation as a rotation matrix R are in Eq. (11) as well as the translation vector $\mathbf{T_o}$ and the rotation angle θ.

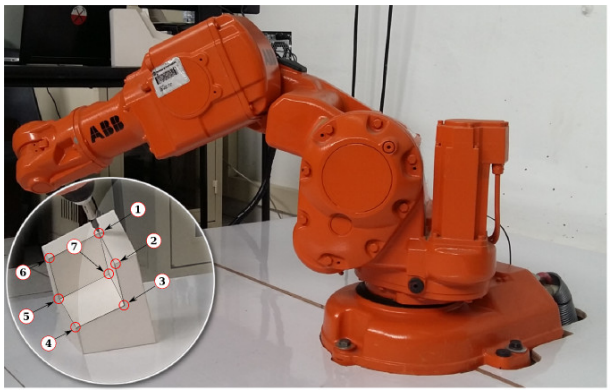


Figure 6. Control mock-up and robot ABB IRB-140

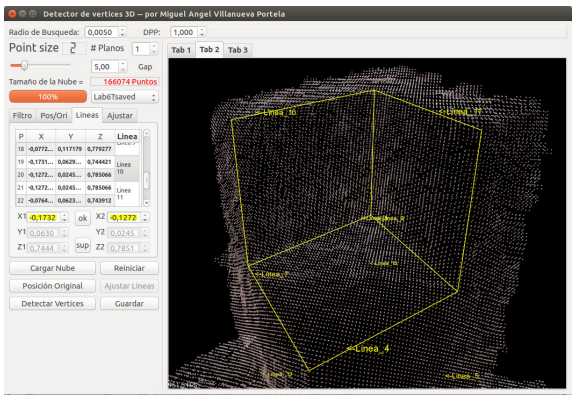


Figure 7. Control points measured with the GUI

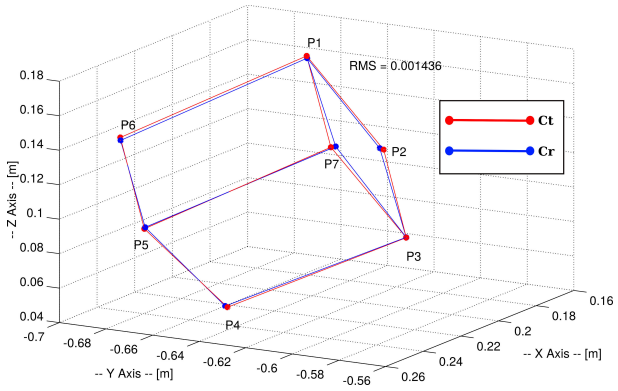


Figure 8. Plot of the $\mathbf{C_r}$ and $\mathbf{C_t}$ points

$$\mathbf{q_r} = \begin{bmatrix} q_0 \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} -0.37637 \\ 0.48334 \\ 0.61721 \\ -0.49374 \end{bmatrix}$$

$$R = \begin{bmatrix} -0.249454 & 0.224982 & -0.941889 \\ 0.968303 & 0.045211 & -0.245651 \\ -0.012683 & -0.973313 & -0.229129 \end{bmatrix} \quad (11)$$

$$\mathbf{T_o^T} = \begin{bmatrix} 0.88936 & -0.33779 & 0.37330 \end{bmatrix}$$

$$\theta = 224.22°$$

Table 1. Measurement of the control points

Points measure by the robot - $\mathbf{C_r}$				**Points measure by the Kinect - $\mathbf{C_k}$**			**Transformed points $\mathbf{C_t}$**				**Error**$^{(1)}$ [%]
Point 1	X_r =	0.187300	m	X_k =	-0.127217	m	Point 1	X_t =	0.187170	m	0.069187
	Y_r =	-0.652400	m	Y_k =	0.024521	m		Y_t =	-0.652717	m	0.048649
	Z_r =	0.170000	m	Z_k =	0.785066	m		Z_t =	0.171168	m	0.686993
Point 2	X_r =	0.221500	m	X_k =	-0.076405	m	Point 2	X_t =	0.221766	m	0.120004
	Y_r =	-0.593600	m	Y_k =	0.062338	m		Y_t =	-0.591697	m	0.320637
	Z_r =	0.143500	m	Z_k =	0.743912	m		Z_t =	0.143145	m	0.247242
Point 3	X_r =	0.200900	m	X_k =	-0.077282	m	Point 3	X_t =	0.201013	m	0.056208
	Y_r =	-0.598900	m	Y_k =	0.117179	m		Y_t =	-0.598754	m	0.024389
	Z_r =	0.081800	m	Z_k =	0.779277	m		Z_t =	0.081676	m	0.151960
Point 4	X_r =	0.253100	m	X_k =	-0.123554	m	Point 4	X_t =	0.253029	m	0.027879
	Y_r =	-0.634400	m	Y_k =	0.150394	m		Y_t =	-0.633451	m	0.149634
	Z_r =	0.058500	m	Z_k =	0.744240	m		Z_t =	0.057962	m	0.919683
Point 5	X_r =	0.225200	m	X_k =	-0.173419	m	Point 5	X_t =	0.224925	m	0.122100
	Y_r =	-0.691300	m	Y_k =	0.119557	m		Y_t =	-0.691894	m	0.085914
	Z_r =	0.081500	m	Z_k =	0.779919	m		Z_t =	0.080433	m	1.308700
Point 6	X_r =	0.246800	m	X_k =	-0.173188	m	Point 6	X_t =	0.245573	m	0.497309
	Y_r =	-0.684400	m	Y_k =	0.062975	m		Y_t =	-0.685508	m	0.161930
	Z_r =	0.142700	m	Z_k =	0.744421	m		Z_t =	0.143636	m	0.656010
Point 7	X_r =	0.165100	m	X_k =	-0.127399	m	Point 7	X_t =	0.166424	m	0.801784
	Y_r =	-0.658100	m	Y_k =	0.081165	m		Y_t =	-0.659079	m	0.148775
	Z_r =	0.107900	m	Z_k =	0.820671	m		Z_t =	0.107880	m	0.018826

$^{(1)}$ Percentage Error $\left\| \frac{C_t - C_r}{C_r} \right\| \times 100$

The root-mean-square (RMS) deviation for C_r and C_t is : RMS deviation $= 0.0014360$

From the Tab. 1 is possible to see that the biggest and smallest percentage error value are 1.308700% and 0.018826%, this is equivalent to a difference of 1.046 mm and 0.00002 mm respectively, this is an acceptable result considering the measuring error introduced by the human factor at the moment of positioning the end effector during the measure process with the robot ABB IRB-140, because this robot has a repeatability of 0.03 mm.

4.2 Test on metallic mock-up

The following figures Fig. 10, Fig. 12 and Fig. 14 display the result of some tests carried out on metallic mock-up, in Fig. 10 is possible to see how the segmentation part works in the algorithm, each detected plane is colored with a different color, in Fig. 12 and Fig. 14 more than the necessary lines appear this is the raw result, these lines can be suppressed using the GUI. In the future is expected to add another step to the algorithm, where the parallel lines can be reduced. The Fig. 15 shows a problem that sometimes appears when a metallic mock-up has good finished on the surface, sometimes this problem can be avoided by changing the work piece orientation relative to the sensor and modifying the light conditions.

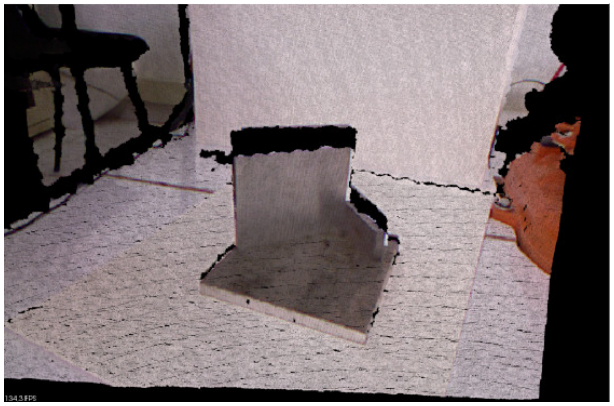


Figure 9. Control mock-up and robot ABB IRB-140

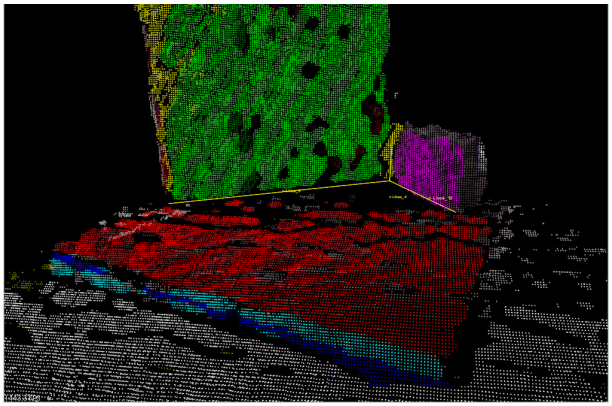


Figure 10. Control points measured with the GUI

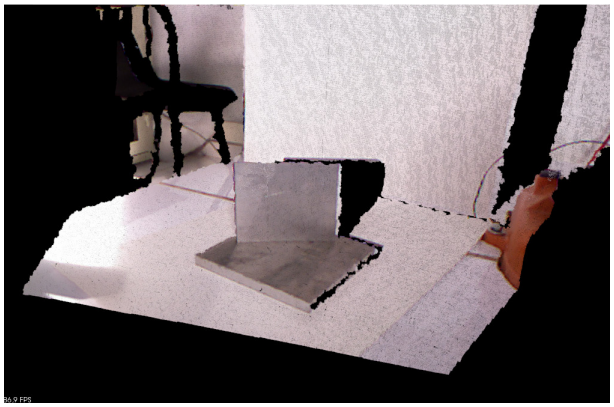


Figure 11. Control mock-up and robot ABB IRB-140

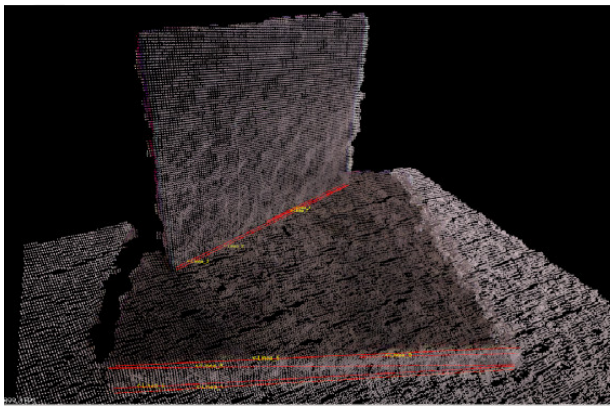


Figure 12. Control points measured with the GUI

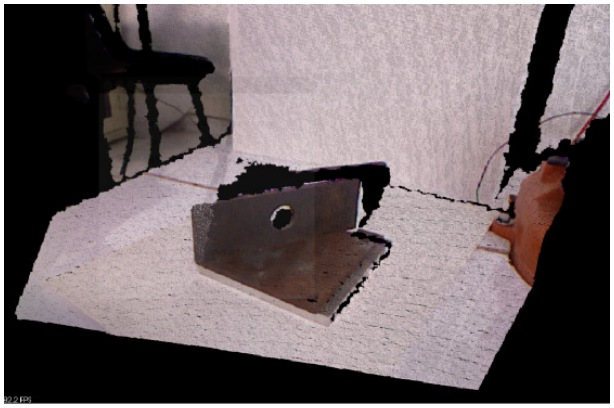


Figure 13. Control mock-up and robot ABB IRB-140

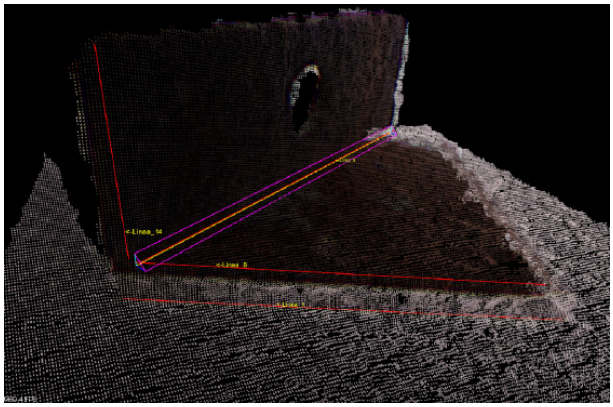


Figure 14. Control points measured with the GUI

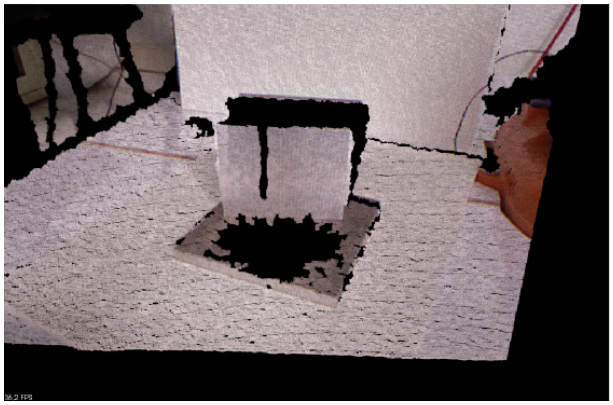


Figure 15. Laser Problem on a metallic mock-up

5. CONCLUSIONS AND FUTURE WORK

The goal of identify and record the straight edges of T joins with square groove and/or in inside corner join with no preparation that will be welded, in order to facilitate the programming process was fulfill. This means that the proposed methodology of work with 3D rather with 2D images is functional and able to be implemented in some applications were the 2D image processing is dominant. Such result imply that a new stage of processing algorithms is emerging.

Is relevant to stand out that there were no need to perform a sensor calibrations as it was required when the stereovision technology is employed, this is possible since the proposed algorithm works directly with three-dimensional points captured by the IR deep sensor instead of the two-dimensional pixels. Nevertheless the Kinect comes pre-calibrated from the factory, and its default parameters was enough. However, to do a calibration on the Kinect sensor to improve its precision is also possible, as is explained in Karan (2013) and in Gui *et al.* (2014). In the future the Kinect calibration will be performed to verify how much the accuracy is improved for the present application.

It is necessary to tune up some details in the last steps of the algorithm to reduce the human intervention even more. As the capacity of classify and filter parallel lines, too.

The GUI developed demonstrate to be easy to use, it has good performance during the algorithm execution, and it's a valuable tool during the design stage. The benefits of the proposed system span from the easy of use to the quick recognition of the geometric features required in the path and trajectory planning process related to welding robots. For the future is expected to improve the algorithm code to enhance this performance, and also add optimization algorithms to get better results in the last steps. Another improvement will be adding the capacity to detect cylindrical surfaces.

Another point to take in count is to use a better mock-up in the process of find the transformation between the sensor and robot coordinates systems to reduce the error of measure, implement a 3D printed muck-up as an option in the future.

The coming work has the challenge to adapt the 3D edge detector algorithm in such way that it can work with registered point cloud.

6. ACKNOWLEDGEMENTS

The first author thanks to Colciencias for the Ph.D. course scholarship and both authors thanks the Departamento en Ingeniería Mecánica y Mecatrónica DIMM for the support in the present research.

7. REFERENCES

Bähnisch, C., Stelldinger, P. and Köthe, U., 2009. "Fast and Accurate 3D Edge Detection for Surface Reconstruction". *Pattern Recognition*, pp. 111–120. doi:10.1007/978-3-642-03798-6_12.

Chen, S.B., Chen, X.Z., Qiu, T. and Li, J.Q., 2005. "Acquisition of Weld Seam Dimensional Position Information for Arc Welding Robot Based on Vision Computing". *Journal of Intelligent & Robotic Systems*, Vol. 43, No. 1, pp. 77–97. doi:10.1007/s10846-005-2966-6. URL http://dx.doi.org/10.1007/s10846-005-2966-6 http://link.springer.com/article/10.1007/s10846-005-2966-6.

Chongjian, F., Chen, S.B. and Lin, T., 2007. "Visual Sensing and Image Processing in Aluminum Alloy Welding". In T.J. Tarn, S.B. Chen and C. Zhou, eds., *Robotic Welding, Intelligence and Automation*, Springer Berlin Heidelberg, 1954, chapter 32, pp. 275–280. ISBN 978-3-540-73374-4. doi:10.1007/978-3-540-73374-4_32. URL http://dx.doi.org/10.1007/978-3-540-73374-4_32 http://link.springer.com/chapter/10.1007/978-3-540-73374-4_32.

Dinham, M. and Fang, G., 2013. "Autonomous weld seam identification and localisation using eye-in-hand stereo vision for robotic arc welding". *Robotics and Computer-Integrated Manufacturing*, Vol. 29, No. 5, pp. 288–301. ISSN 07365845. doi:10.1016/j.rcim.2013.01.004. URL http://linkinghub.elsevier.com/retrieve/pii/S0736584513000057.

Dinham, M. and Fang, G., 2014. "Detection of fillet weld joints using an adaptive line growing algorithm for robotic arc welding". *Robotics and Computer-Integrated Manufacturing*, Vol. 30, No. 3, pp. 229–243. ISSN 07365845. doi:10.1016/j.rcim.2013.10.008. URL http://linkinghub.elsevier.com/retrieve/pii/S0736584513000896.

Fischler, M.a. and Bolles, R.C., 1981. "Random Sample Consensus: A Paradigm for Model Fitting with". *Communications of the ACM*, Vol. 24, pp. 381–395. ISSN 00010782. doi:10.1145/358669.358692.

Gui, P., Ye, Q., Chen, H., Zhang, T. and Yang, C., 2014. "Accurately Calibrate Kinect Sensor Using Indoor Control Field". In *Earth Observation and Remote Sensing Applications (EORSA), 2014 3rd International Workshop on*. IEEE, changsha, pp. 9–13. ISBN 9781479941841. doi:10.1109/EORSA.2014.6927839. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6927839&isnumber=6927831.

Horn, B.K.P., 1987. "Closed-form solution of absolute orientation using unit quaternions". *Journal of the Optical Society of America A*, Vol. 4, No. 4, pp. 629–642. ISSN 1084-7529. doi:10.1364/JOSAA.4.000629. URL https://www.osapublishing.org/josaa/abstract.cfm?uri=josaa-4-4-629 http://people.csail.mit.edu/bkph/papers/Absolute_Orientation.pdf.

Karan, B., 2013. "Calibration of Depth Measurement Model for Kinect- Type 3D Vision Sensors". In V. Skala, ed., *WSCG 2013: Poster Proceedings: 21st International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision in co-operation with EUROGRAPHICS Association*. Václav Skala - UNION Agency, pp. 61–64. ISBN 9788086943763. URL http://wscg.zcu.cz/WSCG2013/!_2013-WSCG-Poster-Proceedings.pdf https://otik.uk.zcu.cz/handle/11025/10626.

Pachidis, T.P. and Lygouras, J.N., 2007. "Vision-Based Path Generation Method for a Robot-Based Arc Welding System". *Journal of Intelligent and Robotic Systems*, Vol. 48, No. 3, pp. 307–331. doi:10.1007/s10846-006-9076-y. URL http://dx.doi.org/10.1007/s10846-006-9076-y http://link.springer.com/article/10.1007/s10846-006-9076-y#.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.