OBSTACLE AVOIDANCE FOR A ROBOT MANIPULATOR BASED ON VISUAL FEEDBACK

Fernando J. Mendiburu*, Marcos R.A. Morais*, Antonio M.N. Lima*

*Graduate Program in Electrical Electrical Engineering - PPgEE
Department of Electrical Engineering - DEE
Universidade Federal de Campina Grande - UFCG
Rua Aprígio Veloso, 882, 58429-900 Bairro Universitário
Campina Grande, Paraíba, Brazil

Emails: fernando.mendiburu@ee.ufcg.edu.br, morais@dee.ufcg.edu.br, amnlima@dee.ufcg.edu.br

Abstract— In this article a visual feedback control system and object occlusion handling implementation is presented. The proposed system integrates a trajectory planning method into the controller of a five degrees of freedom manipulator aiming to simplify object handling and obstacle avoidance. The system aims the intelligent handling of several parts, generating obstacle free paths whenever possible. The workspace information is provided by a redundant vision system: a RGB camera placed over the handler actuator and two fixed RGB-D sensors.

Keywords— Intelligent Automation, manipulator, path planning, RGB-D Sensors.

1 Introduction

The integration of a trajectory planner for a robotic manipulator with a stereo vision system capable of finding the obstacles pose and target objects it is very important for the safe manipulation of objects. The implementation of a robotic vision system is a theoretical and practical challenging problem, and some computational vision systems applications can be found in literature (Mendiburu et al., 2013). The research is relevant for social, academic and industrial aspects, and the development of automated systems do improve product quality, fosters mass production, and also reduces time and cost of production. Quality improvement is provided by the high accuracy and repeatability of the robots. The use of handlers enables the development of more efficient automation, leading to increased reliability to the processes.

In the proposed system we use the Pegasus 880-RA2-1-B as the manipulator robot. The redundant vision system is composed an IR camera, namely IR-Syntek STK1160, placed at the arm actuator, and two RGB-D devices, Kinect sensors, statically positioned external to the manipulator and perpendicular to each other to extend the platform field of view. The workspace is shown in Fig. 1. It can be seen the vision system integrated to the robotic platform: the RGB camera, the frontal and lateral RGB-D sensors. The system is robust as an obstacle free trajectory can be determined even when there are occlusions in the workspace.

The automatic workspace determination with computational vision only is a complex problem due to variations in scene illumination. The use of a RGB-D sensor as the Kinect simplifies the solution by capturing the scene depth using an infrared pair (CMOS camera/IR projector). The system shows robustness to changes in illumination that could otherwise degrade its performance. The emerging of RGB-D sensors like Kinect have promoted an evolution in vision systems due mainly to its low cost and remarkable technical features (Rakprayoon et al., 2011) and (Macknojia et al., 2013). After workspace determination, a path for task execution has to be established. For the routing, we used a PRM based algorithm, called SBL (Sanchez and Latombe, 2001), with random exploration and fast convergence already used in several robotic manipulators, such as in (Liu et al., 2009) and (Guernane and Belhocine, 2005).

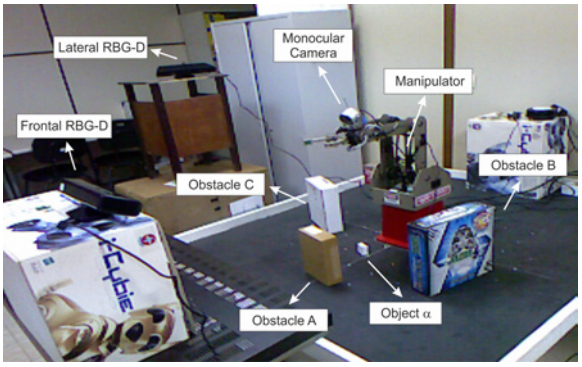

Figure 1: Workspace including the robot and vision system.

The article is organized as follows: Section 1 describes the platform used in the experimental tests. Section 2 discusses how platform configuration has been configured. Section 3 introduces the control strategy that exploits visual feedback for obstacle avoidance with occlusion of objects in the scene. Section 4 presents and discusses the results which demonstrates the feasibility of the proposed solution to deal with object occlusion and the integration of visual feedback trajectory planning for object handling and obstacle avoidance. Section 5 draws the conclusions.

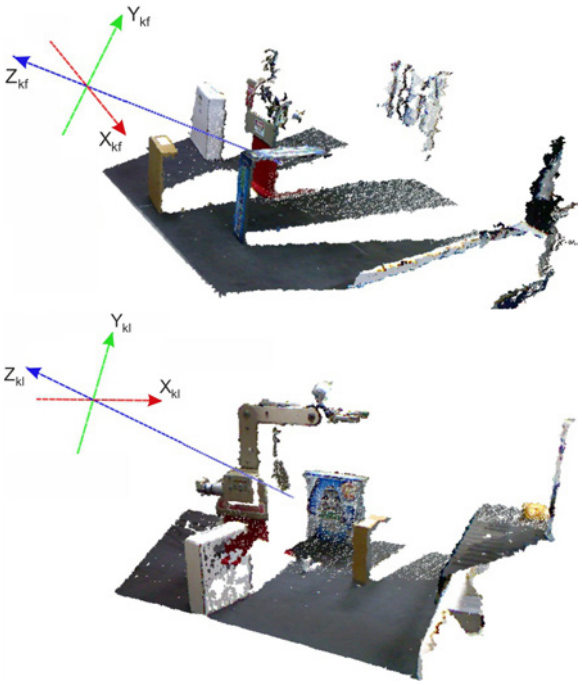

Figure 2: Frontal and lateral RGB-D acquisition.

2 Platform Set-Up

For precise measurements the monocular and RGB-D sensors models need to be obtained and calibrated. For the cameras the pinhole model was chosen. The calibration is important to relate the camera acquired information with the manipulator workspace. It gives the camera geometry and lens distortion parameters. Then, it is possible to correct the lens distortion and interpret the physical world more precisely.

After the calibration, the radial k and tangential p distortions parameters can be obtained. Next, a correction in the images is applied in order to reduce both RGB-D sensors and effector camera distortions, using the estimated distortion coefficients for each. Hence, the RGB-D sensor calibrated disparity (d_k) for each RGB-D device and the measurable values in Cartesian space (X,Y,Z) are obtained. Finally, using $H_{kf} = [R_{kf}|T_{kf}]$ ($H_{kl} = [R_{kl}|T_{kl}]$), each point in space acquired by the Kinect IR is reprojected into the Kinect RGB image and each pixel color is obtained.

The RGB-D sensors output are represented as a point clouds that characterizes the workspace. Those point clouds are specified as a function of each RGB-D sensors coordinate systems (O_{kf} and O_{kl}). See Fig. 2.

The point cloud of each sensor has to be represented in the manipulator reference system O_b with origin in its base. That is, the homogeneous transformation matrixes ${}^{O_b}_{\text{kf}}H$ and ${}^{O_b}_{\text{kl}}H$ relating those points from one coordinate system to another have to be found. This is necessary to the later arm path planning. With manual alignment of the two referencing system the homogeneous matrixes ${}^{O_b}_{\text{kf}}H$ and ${}^{O_b}_{\text{kl}}H$ were obtained and placed in the results section. During the alignment the Meshlab (*MeshLab*, 2012) software was used. A manual rotation and translation of the reference system O_b was applied up to three axis alignment with O_{kf} and O_{kl} system to the matrix ${}^{O_b}_{\text{kf}}H$ and ${}^{O_b}_{\text{kl}}H$, respectively.

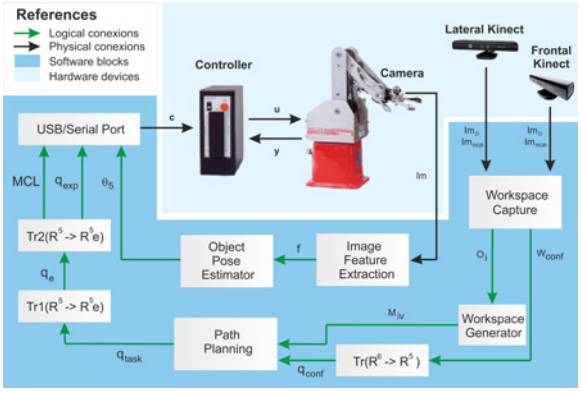

Figure 3: Control Scheme.

3 Control System

The control scheme proposed is represented in Fig. 3. An internal fast joint control ($[u, y]$ loop) and two slower externals loops for vision and planning are defined. Path planning, target object position and obstacle pose are first calculated by the RGB-D sensor loop among other features. Target object orientation is then calculated by the monocular camera loop when the effector is over it. The description and function of each block is explained in the following sections. These blocks are implemented in MATLAB script and C++ language.

3.1 Workspace capture

The RGB-D sensors captures the work scene where the objects lay in a planar surface. The block was implemented in C++ language where a function is called to characterize the workspace defining the robot, the obstacles and the target object. The implementation is based on Burrus (*RGBDemo*, 2011). The model acquisition is obtained with a single scene image from each RGB-D sensor. The algorithm estimates the planar equation of the table and then extracts the agglutinations (objects) using the PCL library (Rusu and Cousins, 2011). The estimation is based on the extrusion of the superior points in from the agglutinations up to the table, assuming that the objects are planar. The volume, color and center of mass are estimated too. The features extracted from each object allow to define the target object position (using the center of mass) and pose of the obstacles. The objects center of mass $\overrightarrow{\sigma}$ is calculated based on the workspace occlusions. When no occlusion occurs in any sensor, the nearest RGB-D sensor to the object is used to define the center of mass, allowing the error reduction in the effector pose. If partial or total occlusion occurs in one RGB-D sensor, the other sensor is used. With partial occlusion in both devices, based on the data obtained from both sensors one may estimate the center of mass $\overrightarrow{\sigma}$ by (1)

$$\overrightarrow{\sigma} = \overrightarrow{\sigma_{kf}} + (\overrightarrow{\sigma_{kl}} - \overrightarrow{\sigma_{kf}})\lambda, \qquad (1)$$

then, the mean point ($\lambda = 0.5$) of the straight line joining both centers of mass is used. The initial and final effector poses and the initial and final target object poses are defined in W_{conf}, a 4×6 matrix. Using the volume and color calculated the target object is determined, eliminating that way that point cloud from the scene modelling. For each object the point (X, Y, Z) with least euclidean distance d to the coordinate system O_b is calculated. With each object it is possible to eliminate the point cloud of the manipulator from the workspace since the manipulator will be the minimum distance to O_b. The point cloud of each obstacle i (o_i), is stored in VRML 1.0 files, which is the block output.

3.2 Workspace generator

This module takes the obstacles point cloud o_i and creates a workspace model M_{iv}, in VRML format, with those obstacles, the robot model and the floor model considered plain. The M_{iv} model is sent to the path planner. The point cloud of the obstacles o_i becomes a mesh by Delaunay Triangulation. That modeling is used because the polygons are quite simple to represent and have many favorable features. The vertex joint into triangles using that method was based on the program in (Mian, 2006). The o_i meshes (de Berg et al., 2008) are stored as triangles in VRML 1.0 files.

3.3 Tr($\mathbb{R}^6 \rightarrow \mathbb{R}^5$)

Program to calculate the initial and final effector poses (home pose) and the initial and final target object (W_{conf} is a 4×6 matrix) as a function of the manipulator configuration space (q_{conf} is a 4×5 matrix), given the equivalent joint angles to the spatial position. The angles are therefore transformed in the normalized form $\mathbb{C} = \left\{[0\,,\,1]^5\right\}$ to be given by the routing algorithm. To obtain those configuration it is used inverse kinematics of the manipulator. The three configurations in the configuration space q_{conf} are saved in a text file and used by the path planner to generate a route free of obstacles.

3.4 Path planning

The library in (Schwarzer et al., 2001) is used in this function. It implements the SBL algorithm (Sanchez and Latombe, 2001) that receives as input the workspace file M_{iv} and the configuration q_{conf} to create the final path free of obstacles (if it exists). The output q_{task} is a matrix of format $m \times 5$ where m is the number of normalized configurations that could implement the task. The output goes to a file where the task to be implemented by the manipulator controller is specified.

3.5 Tr1($\mathbb{R}^5 \rightarrow \mathbb{R}^5 e$)

A block that receives the normalized configuration space task q_{task} and transform those angles in encoder values q_e ($m \times 5$ matrix) to the motors of the arm joints. Those encoder values are used by the controller to perform the motions.

3.6 Tr2($\mathbb{R}^5 \rightarrow \mathbb{R}^5 e$)

A similar function which generates the target object pose of the manipulator in a varying degree attached to roll θ_5 for the 180 rotational positions. The block generate two files: the first has values of position encoders found in the planner added to the 180 remaining positions corresponding to the variation of joint θ_5 to obtain q_{exp}, a $(180 + m) \times 5$ matrix. The second file contains the programming language named MCL, which will make the driver perform the movements free of obstacles, defines the delays between commands, the communication via serial port with camera vision system mounted on the end-effector, when to pick up the object, release it, etc. Also this file allows interaction with the inputs and outputs of the manipulator controller: define when connect a belt, a pump or motor, among other applications. Finally an script to automate the API of the controller is opened (*AHK*, 2013). This is required since the application can not be controlled via the command system.

3.7 USB/Serial Port

This module has the function of performing logical-physical integration between the manipulator and the two vision loops through the USB port. The block receives the object angle θ_5 obtained in the monocular vision loop when the manipulator camera is positioned over the target object. The Tr2($\mathbb{R}^5 \rightarrow \mathbb{R}^5 e$) outputs are issued too. The serial port is a virtual port that allows sending the angle θ_5 to choose the target object pose among the 180 possible poses. The module output are the commands c to the controller perform desired movements.

3.8 Manipulator controller

The manipulator controller receives the information c and encoders measurements y. It use direct kinematics to generate the signals for each motor to move the joints.

3.9 Image Feature Extraction

This block aims at extracting the information from the image Im, giving a synthetic and robust description of the target object. It is captured by taking a top image using the IR-Syntek camera. The object position is provided by the RGB-D sensors. To obtain the object features f (in our implementation the target object area) we first apply a mean filter which allows the homogenization of the brightness and saturation of the image. Then, the threshold technique is applied to obtain a binary image and consequently extract the target object area. Also morphological open and close operation are used retaining the shape of objects and eliminating isolated values in relationship with neighboring pixels. The implementation of that block was made using OpenCV (Bradski and Kaehler, 2008).

3.10 Object pose estimation

This block provides the image interpretation. It is concerned with the measurement of image feature pa-

ptr	STK1160	kf_{rgb}	kf_{ir}	kl_{rgb}	kl_{ir}
$f.k_x$	714.33	514.10	561.68	518.45	572.23
$f.k_y$	634.56	514.37	561.02	522.74	571.91
c_x	349.85	325.94	317.21	323.13	329.22
c_y	275.47	251.14	227.74	255.86	238.46
k_1	-0.4492	0.2412	0.2065	0.2746	-0.2271
k_2	0.2842	-0.8139	0.8088	-0.8002	0.8834
p_1	0.0012	-0.0027	-0.0036	-0.0025	-0.0009
p_2	-0.0041	-0.0009	-0.0005	0.0012	0.0001
k_3	0.0000	0.8612	-0.9960	0.9313	-0.9726

Table 1: Intrinsic and distortion parameters for the cameras.

rameters from the image regions obtained. With the features f obtained from the previous block the orientation of the target object are calculated to be manipulated with the moments technique. The output is the roll angle orientation θ_5 which completes the pose W_{conf} provided by the RGB-D sensor.

$$\theta_5 = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}\right) + 90^{\circ}. \quad (2)$$

Using this expression one can calculate the orientation of the desired object as a function of the effector of the manipulator. This module was implemented in OpenCV (Bradski and Kaehler, 2008).

4 Results

To calculate the intrinsic parameters for arm camera, a manual calibration routine proposed in (Bouguet, 2011) was used due to the big lens distortion. Thirty pictures were taken and the twenty sharpest ones have been chosen. With the calibration performed the intrinsic and distortion parameters of the end-effector, lateral RGB and CMOS Kinect cameras can be seen in Table 1. It is observed that the tangential distortion can be neglected since it is much smaller than the radial distortion. Those parameters obtained are important and can be used to correct the image. In the case of RGB-D sensors a manual calibration was not necessary due to a good detection of the chess board edges. An automated routine proposed by (OpenCV, 2013) and implemented by Burrus (*RGBDemo*, 2011) was used to calibrate the sensor. The stereo calibration gives the orientation and translation H_{kf} and H_{kl} between the Kinect cameras.

$$H_{kf} = \begin{bmatrix} 0.9996 & -0.0001 & 0.0269 & 0.0247 \\ 0.0013 & 0.9989 & -0.0467 & 0.0017 \\ -0.0268 & 0.0467 & 0.9986 & -0.0022 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3)$$

$$H_{kl} = \begin{bmatrix} 0.9999 & 0.0002 & -0.0112 & 0.0242 \\ -0.0009 & 0.9992 & -0.0431 & 0.0022 \\ 0.0015 & 0.0554 & 0.9974 & -0.0015 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (4)$$

Frontal Kinect cameras (IR and RGB) have approximately the same orientation as it is shown in (3). Similar results was obtained in the Lateral Kinect cameras with a 2.5 cm baseline as it is shown in (4). The RMS reprojection error were 0.25 for the monucular camera, 0.19, 0.17 for the frontal kinect IR and frontal kinect RGB cameras and , 0.16 and 0.17 for the lateral kinect IR and lateral RGB cameras. Then, the ${}^{O_b}_{\text{kf}}H$ (${}^{O_b}_{\text{kl}}H$) homogeneous transformation matrix relating the points from O_{kf} to O_b (O_{kl} to O_b) is calculated as given by

$${}^{O_b}_{\text{kf}}H = \begin{bmatrix} -0.9997 & -0.0232 & -0.0049 & 0.1106 \\ 0.0044 & -0.3817 & 0.9243 & 1.0314 \\ -0.0233 & 0.9240 & 0.3819 & 0.5431 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5)$$

$${}^{O_b}_{\text{kl}}H = \begin{bmatrix} 0.1060 & -0.4482 & 0.8877 & 1.1448 \\ 0.9941 & 0.0684 & -0.0841 & 0.1377 \\ -0.0231 & 0.8913 & 0.4528 & 0.7892 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (6)$$

To evaluate the system performance a object occlusions workspace is presented in Fig. 1. For this scenario, a planning solution can not be obtained with only the frontal RGB-D sensor as in (Mendiburu et al., 2013). More general situations can not be covered with only one 3D capture device. The workspace captured by both sensors is show in the Fig. 2.

It can be noticed that the frontal RGB-D sensor could not detect the target object (object α) due to the occlusion by the obstacle in front of it. Three obstacles and the manipulator are captured by that device. However, the lateral RGB-D sensor detect the target object, the same three objects and the manipulator from a different perspective, adding information to the obstacles modelling. In the Fig. 4 it is presented the composition of both partial workspace visions (Fig. 2) that are represented in the manipulator base coordinate system using the matrixes ${}^{O_b}_{\text{kf}}H$ and ${}^{O_b}_{\text{kl}}H$. It can also be noticed the object occlusion elimination in the point cloud. Finally the extrusion and conversion from the point cloud into triangle mesh allows complete their models.

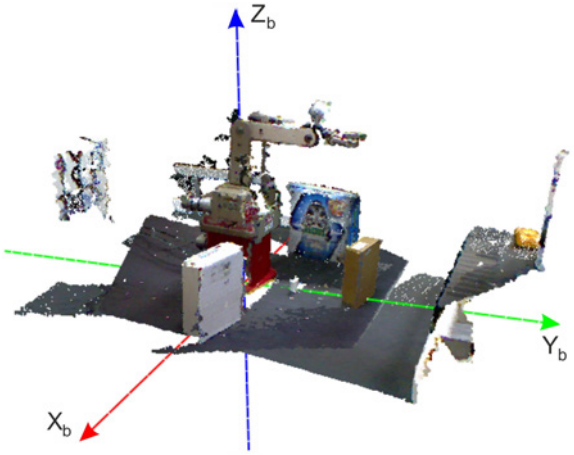

Figure 4: Composing the workspace.

In the test, the path planner should perform an arm routing between the configurations $q_{conf} = [\ q_{home}\ \ q_{inicial}\ \ q_{objetivo}\ \ q_{home}\ ]^{-1}$, beginning from its initial position q_{home}, load the target object at $q_{inicial}$, unload it at $q_{objetivo}$ and go back to q_{home}, finding a path between those configurations

that takes the robot between those poses without colliding with obstacles. It is a fail or error if that trajectory doesn't exist or not it is not found. The path planner parameters are defined as $\varepsilon = 0.012$, $\rho = 0.15$ and $s = 500$. In the Fig. 5 the simulation of the routing presenting the task is shown. The q_{task} configuration is a 14×5 matrix with 14 configurations to be performed. The Fig. 5.1 shows the start configuration q_{home} coinciding with q_1, the Fig. 5.2 presents the $q_{inicial}$ configuration that allows to position the effector right above the target object (q_5), the Fig. 5.3 shows $q_{objetivo}$, the desired configuration (q_{10}) and the Fig. 5.4 shows the final configuration q_{home} (equivalent to q_{14}). From those figures it can also be observed the intermediary configurations and segments that allowed to reach each of those configurations avoiding obstacles. The trajectory found by the path planner was the one with less poses in three executions of the SBL algorithm. The total time of execution to perform the task is $112,24\,s$ including the execution of the vision steps.

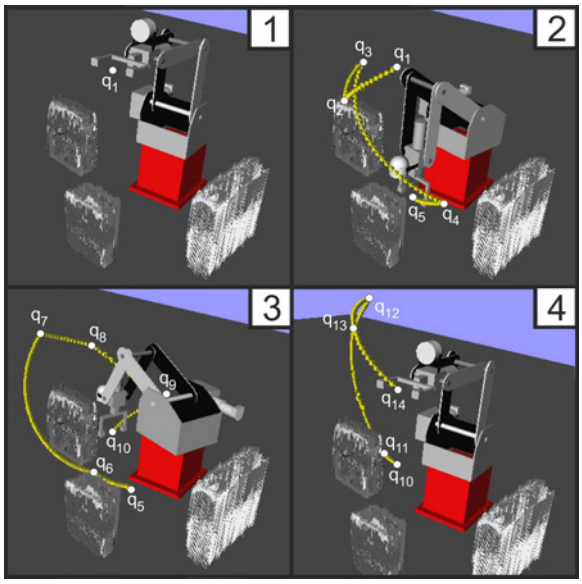

Figure 5: Path planning simulation.

In the Fig. 6 the implementation of the manipulator task is presented. The Fig. 5.1, 5.2, 5.3 and 5.4 are related with the configurations q_{home}, $q_{inicial}$, $q_{objetivo}$, and q_{home} of the Fig. 6, respectively. In Table 2 the values of the encoderes q_e to perform the task are presented.

The target object dimensions are $42 \times 20 \times 40\,mm$. The center of mass detected by the system was $\overrightarrow{\sigma_e} = \begin{bmatrix} 3.84 & 18.99 & 1,89 \end{bmatrix}\,cm$ related to manipulator base system. It can be observed that the effector is positioned above the target object with a predefined angle $\theta_5 = 0°$. The monocular vision system enabled the calculation of an angle $\theta_5 = 96°$.

The effector position error depend mainly on the vision system of the external loop when performing the center of mass calculation: if the point cloud is not captured accurately, the system performance can

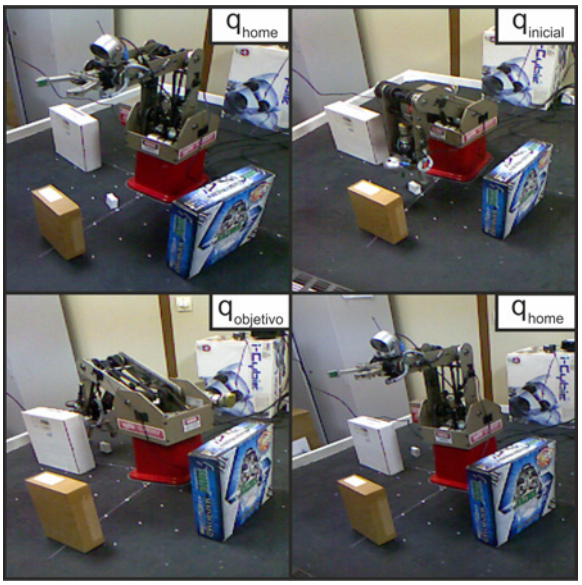

Figure 6: Path planning results.

q_e	$[\ \theta_{e1} \quad \theta_{e2} \quad \theta_{e3} \quad \theta_{e4} \quad \theta_{e5}\ ]$
q_{e1}	$[\ 0 \quad 23 \quad -1 \quad 0 \quad 0\ ]$
q_{e2}	$[97379 \quad -35392 \quad -91961 \quad -31426 \quad -12839]$
q_{e3}	$[181220 \quad 26263 \quad -49818 \quad -12982 \quad 4869]$
q_{e4}	$[-66795 \quad -119722 \quad -165278 \quad -53181 \quad 85283]$
q_{e5}	$[32026 \quad -160202 \quad -230103 \quad -73952 \quad 73952]$
q_{e6}	$[88977 \quad -155118 \quad -230276 \quad -83326 \quad 12400]$
q_{e7}	$[193406 \quad -113856 \quad 55806 \quad -4366 \quad 73711]$
q_{e8}	$[285797 \quad -100158 \quad 14657 \quad -52841 \quad 59735]$
q_{e9}	$[405171 \quad -165720 \quad -89663 \quad -92697 \quad 48032]$
q_{e10}	$[303172 \quad -182191 \quad -158068 \quad -73942 \quad 73942]$
q_{e11}	$[303136 \quad -216888 \quad -106992 \quad -39040 \quad 93102]$
q_{e12}	$[242615 \quad -26756 \quad 72255 \quad 62939 \quad 10644]$
q_{e13}	$[132917 \quad 3855 \quad 46747 \quad 36048 \quad -64]$
q_{e14}	$[\ 0 \quad 23 \quad -1 \quad 0 \quad 0\ ]$

Table 2: Effector configurations q_e [bits].

be impacted. The errors are also dependent on the calibration step for the reference system and less (negligibly) on the manipulator encoders.

To calculate the effector error pose we use the target object α and the obstacle A observed in Fig. 1. In the test, the error pose was calculated by executing only the step "Workspace capture" ten times. The error is calculated as $\varepsilon = \| \overrightarrow{\sigma_v} - \overrightarrow{\sigma_e} \|$. For the object α, the center of mass is $\overrightarrow{\sigma_v} = \begin{bmatrix} 3 & 19 & 2 \end{bmatrix}\,cm$. The obstacle A center of mass is $\overrightarrow{\sigma_v} = \begin{bmatrix} 0.3 & 40 & 10 \end{bmatrix}\,cm$. In the Table 3 the values of $\overrightarrow{\sigma_e}$ are shown. The target object α is completely occluded from the frontal RGB-D. However, the obstacle A is not occluded in any of the RGB-D sensors. It should be noticed the the obstacle is positioned at approximately $1.3\,m$ from the lateral RGB-D and $1\,m$ from the frontal one.

It can be seen that larger errors are observed when the distance between the object and the sensor increases. The target object was defined uniquely by the lateral RGB-D sensor, due to the total occlusion

Obstacle A $\overrightarrow{\sigma_e}\,(cm)$		Object α $\overrightarrow{\sigma_e}\,(cm)$
Sensor Kf	Sensor Kl	Sensor Kf
[0.22 39.87 10.25]	[0.37 40.56 10.30]	[3.64 18.99 1.86]
[0.15 39.93 10.33]	[0.26 40.51 10.34]	[3.53 19.05 1.86]
[0.21 39.92 10.32]	[0.33 40.49 10.36]	[3.55 19.07 1.81]
[0.20 39.93 10.31]	[0.23 40.49 10.37]	[3.57 19.01 1.72]
[0.18 39.97 10.34]	[0.28 40.50 10.36]	[3.61 19.03 1.78]
[0.14 39.95 10.35]	[0.25 40.55 10.31]	[3.59 19.12 1.83]
[0.13 39.90 10.29]	[0.29 40.53 10.35]	[3.62 19.07 1.71]
[0.19 39.91 10.27]	[0.28 40.47 10.34]	[3.62 19.04 1.82]
[0.12 39.92 10.26]	[0.23 40.52 10.29]	[3.56 19.08 1.81]
[0.26 39.94 10.28]	[0.26 40.48 10.31]	[3.54 19.09 1.77]
$\varepsilon = 0,3358\,(cm)$	$\varepsilon = 0.6118\,(cm)$	$\varepsilon = 0,6227\,(cm)$

Table 3: Distance to the center of mass.

in the other sensor. In the monocular vision system there are errors $\leq 1^{\circ}$, which is the size used when 180 rotational positions are defined.

5 Conclusions

A routing method was applied for a manipulator using a redundant vision system avoiding objects occlusion. Object detection, robot and obstacles and part transportation have been successfully performed with good results. The vision system errors have been considered and the computed center of mass included when there are total or partial occlusions for some RGB-D device. The overall control system response is adequate for finding an obstacle free trajectory that is implemented without hurdles. The system quite is robust to illumination changes and objects occlusion. Some improvements can be made by automating the alignment of reference systems between the RGB-D sensors and end-effector. Some form of joint control and the replacement of the monocular camera with another RGB-D sensor will be considered in future implementations in order to improve the precision in object manipulation.

Acknowledgments

The authors would like to thank PPgEE/UFCG, CAPES and CNPq for the financial support and research grants.

References

AHK (2013). http://www.autohotkey.com/.

Bouguet, J. (2011). *Camera Calibration Toolbox for Matlab.* http://www.vision.caltech.edu/bouguetj.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library.*, O'Reilly Media Inc.

de Berg, M., Cheong, O., van Kreveld, M. and Overmars, M. (2008). *Computational Geometry: Algorithms and Applications.*, Springer-Verlag.

Guernane, R. and Belhocine, M. (2005). A smoothing strategy for prm paths application to sixaxes motoman sv3x manipulator, *Conf. Rec. IROS* pp. 4155–4160.

Liu, H., Li, Y., Wen, H., Xia, J. and Chu, T. (2009). Hierarchical roadmap based rapid path planning for high-dof mobile manipulators in complex environments, *Conf. Rec. ROBIO* pp. 189–195.

Macknojia, R., Chavez-Aragon, A., Payeur, P. and Laganiere, R. (2013). Calibration of a network of kinect sensors for robotic inspection over a large workspace, *Conf. Rec. WORV* pp. 184–190.

Mendiburu, F., Morais, M. and Lima, A. (2013). Visual feedback trajectory planning for object handling and obstacle avoidance, *Conf. Rec. IECON* pp. 1–6.

MeshLab (2012). http://meshlab.sourceforge.net/.

Mian, A. (2006). http://www.csse.uwa.edu.au/.

OpenCV (2013). *Camera Calibration and 3D Reconstruction.* http://opencv.willowgarage.com/documentation/.

Rakprayoon, P., Ruchanurucks, M. and Coundoul, A. (2011). Kinect-based obstacle detection for manipulator, *Conf. Rec. SII* pp. 68–73.

RGBDemo (2011). http://labs.manctl.com/.

Rusu, R. and Cousins, S. (2011). 3d is here: Point cloud library (pcl), *Conf. Rec. ICRA* **12**(5): 651–670.

Sanchez, G. and Latombe, J. (2001). A single-query bi-directional probabilistic roadmap planner with lazy collision checking, *Conf. Rec. ISRR* .

Schwarzer, F., Saha, M. and Latombe, J. (2001). *Motion Planning Kit (MPK).* http://robotics.stanford.edu/ mitul/mpk/.