Adaptive Robotic Contour Following from Low Accuracy RGB-D Surface Profiling and Visual Servoing

Danial Nakhaeinia, Pierre Payeur, Robert Laganière
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, ON, Canada
[dnakhaei, ppayeur, laganier]@uottawa.ca

Abstract*—This paper introduces an adaptive contour following method for robot manipulators that originally combines low accuracy RGB-D sensing with eye-in-hand visual servoing. The main objective is to allow for the detection and following of freely shaped 3D object contours under visual guidance that is initially provided by a fixed Kinect sensor and refined by a single eye-in-hand camera. A path planning algorithm is developed that constrains the end effector to maintain close proximity to the surface of the object while following its contour. To achieve this goal, a RGB-D sensing is used to rapidly acquire information about the 3D location and profile of an object. However, because of the low resolution and noisy information provided by such sensors, accurate contour following is achieved with an extra eye-in-hand camera that is mounted on the robot's end-effector to locally refine the contour definition and to plan an accurate trajectory for the robot., Experiments carried out with a 7-DOF manipulator and the dual sensory stage are reported to validate the reliability of the proposed contour following method.

Keywords— contour following; contour detection; visual servoing; RGB-D sensors; eye-in-hand imaging; robotic control.

I. INTRODUCTION

Autonomous contour following in real time based on visual information is a delicate task often involved in industrial robot manipulation. Typically, the robot is holding a tool while closely following the contour of an object, either using previous knowledge about the shape and pose of the surface, or in an adaptive way. Furthermore, contour following can be accomplished with or without contact [1]. Contour following applications include cleaning, inspection, sealing, painting, part polishing, sewing, etc [2]. Previously reported approaches on contour following can be divided into three categories: contact-based, vision-based and hybrid.

In contact-based approaches [3, 4], a direct feedback, either from a force/torque sensor or a position/force sensor, is used to follow the object contours. An initial contact point between the tool and the object must be prescribed and the force controlled contact must be maintained. Therefore, due to the limited bandwidth of the contact or force sensor, the execution speed of the task is limited to prevent loss of contact and information [5].

In visual servoing approaches [6, 7], a camera is mounted on the end effector, in an eye-in-hand configuration, to find and track an object contour using image processing techniques. In contrast to contact-based approaches, visual servoing methods do not impose delays due to the limited contact point. However, they face latency due to computationally heavy image processing and massive data transfer. The time delay can be reduced by adding feedforward signals, but this may result in following wrong contour or misalignment between the tool center point (TCP) and the measured contour [8] when no prior knowledge about the object is available.

To address these shortcomings, recent work proposed hybrid control [5,8,9]. In these approaches a combination of vision and force (position) feedback are used to track and follow the object's contours. These approaches demonstrate a good performance in dealing with planar contour following but are not very successful over 3D structures because the depth of the object's contour is also required for precise path planning. Furthermore, without any global information about the object shape, as it is lacking from an eye-in-hand configuration, it is difficult to keep the contour in the camera field of view. The lack of global information can cause the robot to follow the wrong direction or contour, especially at corners and on curved surfaces.

This work proposes an adaptive contour following methodology without contact that operates in real time and is based on visual servoing. During the process, the tool is constrained to maintain close proximity to the surface while following its contour. Unlike previously reported approaches, this work combines visual cues from a fixed Kinect sensor, located behind the robot at a distance from the object to provide global shape and depth information, and an eye-in-hand color camera mounted close to the end effector that provides higher accuracy measurements on the contour location. The main advantages of the proposed method reside in the rapid localization and shape estimation of the surface that efficiently bring the end effector in proximity to the surface of the object, the precise control of the position of the end-effector on 3D surface contours, and the reduction in complexity and cost that result from the use of a multi-dimensional force sensor.

978-1-4799-4337-1/14 $31.00 © 2014 IEEE
DOI 10.1109/CRV.2014.15

II. PROBLEM FORMULATION

The main goal of this work is to design a reliable path planning method using visual servoing that ensures accurate tracking with a robot manipulator of the contour of an object that is not previously defined by CAD models but only acquired at high speed by affordable off-the-shelf RGB-D sensors, and therefore submitted to important inaccuracies. A Kinect sensor is used as the primary vision stage to provide full coverage and rapid 3D profiling of an object. The Kinect device is a priori calibrated with the robot's base reference frame during the setup procedure. As a result, the 3D points collected by the Kinect sensor are defined by Cartesian coordinates with respect to the robot's base reference frame. This representation provides global 3D information about the object to support the contour following path planning task, but at low resolution [10,11]. Since data acquired with Kinect sensors is not accurate enough to provide a high accuracy definition of the contour, it is used only to support an initial estimation of the depth of the object with respect to the manipulator's base and to approximately locate the contours. In addition, an eye-in-hand camera mounted on the robot's end effector provides closer and higher accuracy feedback about the object's contours (Fig. 1). The robot controller is designed such that the eye-in-hand vision system adds a feed forward signal which provides real-time look-ahead information in front of the tool center point (TCP) of the robot. Fig. 2 shows the proposed control architecture for contour following using the dual input vision stage combining Kinect sensor and eye-in-hand camera.

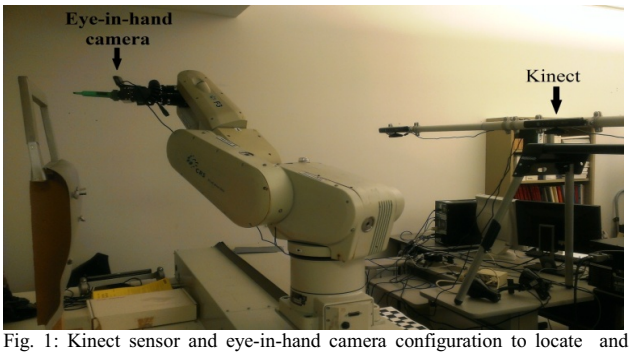


Fig. 1: Kinect sensor and eye-in-hand camera configuration to locate and follow the object contours.

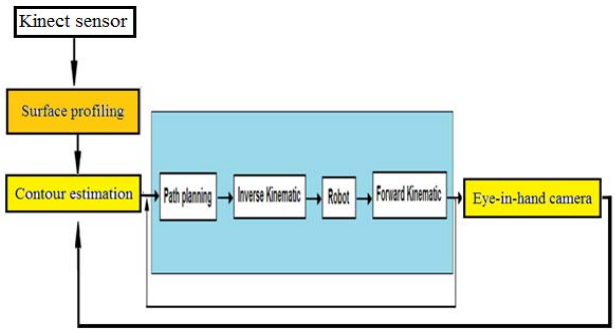


Fig. 2: System control architecture with position feedback (inner loop) and visual feedback (outer loop).

III. DESIGN OF THE CONTOUR FOLLOWER

In order to control the robot for it to accurately follow the contours of an unknown object using the global information from the Kinect sensor and the local information from an eye-in-hand camera, three major processing steps are considered, as detailed in the following sections.

1) Object Surface Localization and Shape Profiling

In this work the primary information about the object's shape, position and orientation is provided by a Kinect sensor. This RGB-D sensing technology is selected because of the rapidity with which it can provide a fair 3D reconstruction of an object of relatively large dimension, as found in numerous industrial applications. The Kinect sensor is used to capture color images of the scene and the depth corresponding to each pixel within the field of view. The data is processed to extract the 3D information about the object, that is its location in the workspace and an estimate of its surface shape. In order to coordinate the movement of the manipulator with the depth data collected by the Kinect sensor, the Cartesian coordinates of the object are defined with respect to the robot's base reference frame. An extensive procedure was proposed in [12] to estimate the internal and external calibration parameters of Kinect sensors, as well as their correspondence with the manipulator's reference frame. The method takes advantage of the depth and color information captured by the RGB-D sensor and achieves complete calibration within a distributed network of RGB-D sensors. To further reduce the shape profiling time, the calibration procedure is performed offline. Therefore, the calibration process does not impact the execution time in production.

The depth information about the object is processed to estimate its specific location, curvature, as well as extract the edges and corners of the object that define, with relatively low accuracy, the contours of the object. These locations are primordial to navigate the robot end effector while keeping the object boundary within the eye-in-hand camera field of view, and to maintain it in close proximity to the object [13], independently from the object's shape. For this purpose, the raw information from Kinect is stored in a textured 3D point cloud. The point cloud represents the shape and visual appearance of the surface (Fig. 3).

2) Border Edge Detection

In this work, we are focusing on following the outer contours of a surface. Therefore, border edges should be extracted from the 3D point cloud. Border edges can be extracted using the angle criterion method [14], convex hull method and splitting technique [15]. But these methods have some shortcomings in dealing with 3D surfaces or tend to be very slow. This work proposes an alternative efficient method to extract the border edges from a 3D point cloud. For this purpose, the surface is divided by N horizontal lines and M vertical lines separated by a distance of 1 mm from each other, along the Y axis and Z axis respectively (Fig. 5a). Horizontal lines go from Y_{min} to Y_{max} and vertical lines from Z_{min} to Z_{max}. The Y_{min}, Y_{max}, Z_{min}

and Z_{max} correspond respectively to the minimum and maximum values of the Y and Z coordinates in the point cloud (Eq.1).

$$M = Y_{max} - Y_{min}$$
$$N = Z_{max} - Z_{min} \quad (1)$$

where M is the number of horizontal lines, and N is the number of vertical lines.

After splitting the surface, the distance between the points which lie on a same line and the line's start point is calculated. The points with the maximum and minimum distances are selected respectively as the border edges. The points that have equal Y or Z coordinates in the point cloud belong to the same horizontal or vertical line respectively (Eq. 2). For example, if P[j] contains the list of points on the first horizontal line, two points which have a minimum and maximum distance from Z_{min} correspond to the border edges (Eq. 3)

$$For\ (i = 0{:}V_{num})$$
$$If\ |V[i].y - Ymin| < 1\ mm,\quad then\ P[j] = V[i] \quad (2)$$

$$Dist = |P[j].z - Zmin| \quad (3)$$

where V_{num} is the number of vertices in the point cloud, V[i].y represents the Y coordinates of points in the point clouds, P[j] map points on the same horizontal line, and Dist is the distance of each point P from $Z_{min.}$

The same procedure repeats for all M horizontal lines and N vertical lines and provides the border edges on the Y-Z plane. In the next step, the same process is performed for the Y-X and Z-X planes. These steps result in extracting all border edges with various depths. Since it is desired to follow the outer contour of an object in the Y-Z plane, given the configuration of the robot with respect to the object, as shown in Fig. 3, the edge borders extracted in different planes regardless of their depth value are compared and the edge borders with maximum and minimum Y and Z coordinates in each row and column represent the outer edge border of the object. These edge points provide the initial estimate of the surface contour, as shown in Fig. 4b. However, given the low resolution on depth information provided by Kinect sensors, these edge points are not accurate enough to precisely guide the robotic contour following operation.

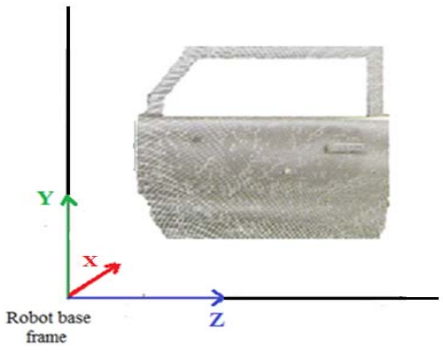
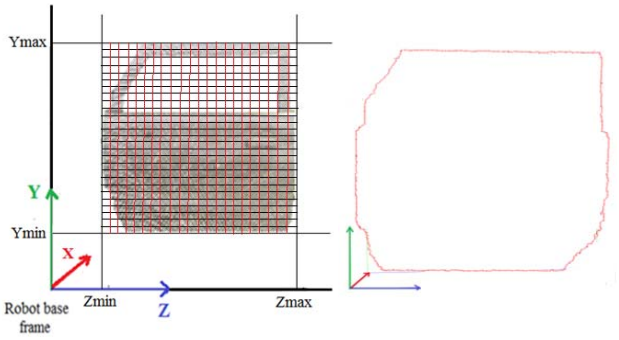


Fig. 3: Textured 3D point cloud captured by Kinect sensor.



(a) (b)

Fig. 4: a) Surface split by M horizontal and N vertical lines, and b) border edges extracted from the point cloud.

3) *Adaptive Contour Following*

The objective of the robot path planning under dual visual guidance is to allow the manipulator's end effector to first approach the surface using the initial RGB-D vision stage, and then to efficiently and precisely follow the object's contour, under higher precision visual servoing provided by the local eye-in-hand camera images. The proposed path planning design is detailed in the following subsections:

a) Contour starting point

A start point over the object's contour must first be defined. The start point can be any of the edge points identified in the edge map extracted from the Kinect sensor data. It provides the primary position and orientation for the end effector of the robot on any of those locations. These point coordinates are sufficient to initially guide the robot in close proximity of the object's contour, which avoids any human intervention or use of fixture to constrain the location of the object. This provides additional flexibility for operation in dynamic or unconstrained environments, as the robotic system is able to self-locate objects of interest. Once the approach phase is completed, the contour is detected and mapped with higher accuracy within the eye-in-hand camera field of view. The path planning and control of the robot is passed to visual servoing while the previously acquired overall model of the object remains as a validation level to assist with the global navigation of the manipulator.

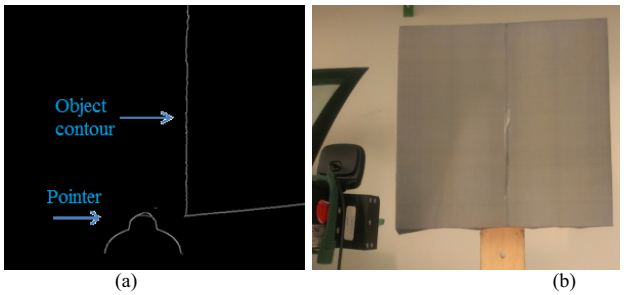
In our experiments, the lower left edge point of the panel is automatically selected as the starting point. For this purpose the closest point to the corner point with Y_{min} and Z_{min} coordinates is searched in the border edges list previously established, as shown in Fig. 4a (Eq. 4).

$$For\ (k=0{:}E_{num})$$
$$Dist = \sqrt{|E[k].y - Ymin|^2 - |E[k].z - Zmin|^2} \quad (4)$$

where E_{num} is the number of edge points, and $E[k]$ is the list of edge points.

The edge point which generates the minimum distance is considered as start point. As shown in Fig. 5, the point initially

selected on the contour of the object using the Kinect based information is typically not precisely aligned with the actual object contours due to the RGB-D sensor inaccuracies. But the robot reaches in close enough proximity to that contour to support an initialization phase, that is for the eye-in-hand camera to closely capture the actual location of the contour and compensate for the gap.



(a) (b)

Fig. 5: Misalignment of the initially detected contour from Kinect data only: a) eye-in-hand camera view at start point, b) third camera view showing the compensated end effector position with input from the eye-in-hand camera.

b) Image processing

After the contours of the objects are defined at low resolution via the Kinect data and the robot end effector has reached in proximity of the starting edge point, higher resolution images of the contour are captured by the eye-in-hand camera and processed in real time to finely locate the contour of the object in immediate proximity to the end effector. For this purpose, a Canny contour detection [16] is performed. The Canny detector generates a binary image which represents segments of the dominant edges with a bright line over a black background. Contours are obtained from these edges and represented by the sequence of neighbor pixels in the binary image that are connected using the chain code technique [17]. As the robot moves, the local information about the object contour is updated step-by-step. The contours detected at each step are then processed and used to compute the next end effector's position.

c) Pruning

Results from image processing define the local contours detected within the field of view of the eye-in-hand camera. However, among the contours that appear in the resulting binary image, there can be contour components originating from various objects present in the scene, beyond the object of interest. These extra contours tend to distract the robot from accurately following the contours of interest. To remedy this situation, the edge point coordinates of the desired object, previously identified from RGB-D sensor data, as described in section III.2, are used to prune the undesirable contours and noises in the local contour edge map provided by processing the eye-in-hand camera images. The estimated object's edge coordinates in the depth map are compared with the local contour coordinates.

However, since the local object contour coordinates are defined with respect to the eye-in-hand image frame, it is first required to transform these coordinates with respect to the robot base reference frame. The eye-in-hand camera being mounted on the robot end-effector, its transformation with respect to the 3end effector reference frame is physically measured and corresponds to a shift of 50mm along the Y axis. Since the distance between the end-effector and the object is kept constant during the contour following, a fixed scaling factor which converts pixels to metric units is used to estimate the local contour's coordinates with respect to the robot base frame, in combination with the known kinematic model of the manipulator. The image resolution captured by the eye-in-hand camera is 640x480 pixels which corresponds to a 100x75 mm field of view over the object. Therefore, a scaling factor of S=0.156 is used to convert the pixels in metric units.

The eye-in-hand camera visual feedback operates with 2D edges. The edge coordinates with respect to the robot base frame are calculated in order to support path planning and robot control with respect to the base of the robot. Since the camera is mounted on top of the end-effector with a shift along the Y axis from the end-effector centre, the binary edge map center is assumed as a reference frame origin to calculate the edges distance from the origin and the end-effector. First the edge point's coordinates with respect to the image centre are calculated. Then they are transformed to the end-effector frame (Eq. 6). Eventually, the point coordinates with respect to the robot base frame are obtained using Eq. 7.

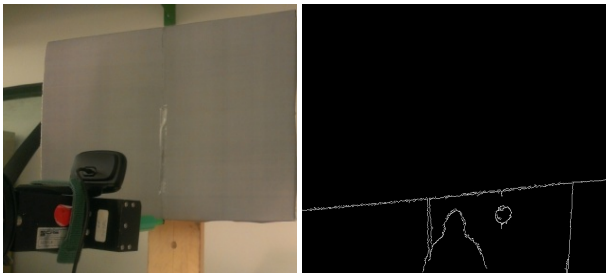
$$ {}^{I}P_{I} = \begin{bmatrix} Xi \\ Yi \\ 0 \end{bmatrix}, \quad {}^{B}P_{E} = \begin{bmatrix} Xe \\ Ye \\ Ze \end{bmatrix}, \quad Q_{CE} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 50 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.156 \end{bmatrix} \tag{5} $$

$$ {}^{E}P_{I} = Q_{CE}\, {}^{I}P_{I} \tag{6} $$

$$ {}^{B}P_{I} = {}^{B}P_{E} + {}^{E}P_{I} \tag{7} $$

where ${}^{I}P_{I}$ represents the edge coordinates with respect to the image center, ${}^{B}P_{E}$ is the end-effector position with respect to the base frame, ${}^{B}P_{I}$ is the edge coordinates with respect to the base frame, and Q_{CE} is the homogenous transformation between the eye-in-hand camera and the end-effector.

The local contours (edge points) detected in the binary image that are not in close proximity to the estimated object contours (list of edge points) extracted from the Kinect sensor data are discarded and not considered for the robot path planning. As shown in Fig. 6, where it is desired to follow the gray rectangular object contour, other components of the scene (e.g. the vertical wooden post) create extra edge points and corresponding sections of contour in the local eye-in-hand image. Using the proposed pruning technique these edge points are efficiently removed from the edge list and ignored for path planning.



(a) (b)

Fig. 6: a) Object original eye-in-hand image, b) local detected contours before pruning.

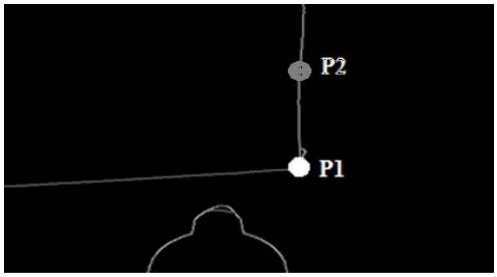


Fig. 7: Priority for the selection of the next tracking position according to the local edges detected from the eye-in-hand visual feedback.

d) Contour following

After pruning of the contours and defining the start point, the object's contour is followed by a pointing tip mounted on the end effector's plate. The motion direction is user defined as clockwise or counter-clockwise. It is assumed here that the motion direction does not change during the contour following operation around the entire piece or segment of contour to be tracked.

For each motion step, the distance to every detected edge pixel in the processed eye-in-hand camera image, representing a valid contour, from the end-effector center is calculated. For each step of the manipulator movement, the closest edge point with a minimum distance of r=30mm from the end-effector in the desired direction of motion is selected as the next end-effector position. The distance, r, can be adjusted according to the contour characteristics. Larger values of r accelerate the contour following operation but do not adapt well to contour following of surfaces with significant curves.

When contours are horizontal or vertical, the priority for choosing the next position is given to the edges on the same horizontal or vertical line with respect to the current position of the robot [18]. However, if a corner (on the surface to track) is detected in the eye-in-hand image, then this corner is selected as the next position for the end effector. Corners on the surface to follow the contours of are detected using a Harris corner detector applied on the image. For example, as shown in Fig. 7, although the distance of the end-effector to *P1* is less than 30 mm, *P1* is a detected as a corner on the same horizontal line with the current position of the end effector. Therefore, *P1* is chosen as the next position to reach. This criterion results in following a contour in a continuous and smooth manner and prevents undesirable oscillations of the end effector in between candidate contour edge points.

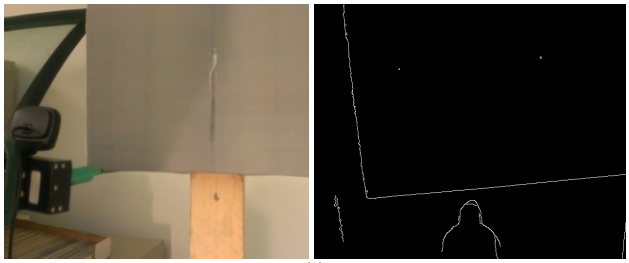
The end-effector and camera orientation do not change during the contour following and remain perpendicular to the surface. Since the eye-in-hand camera cannot provide depth information, the end-effector distance to the surface is adjusted using the edge's depth provided by the Kinect sensor information. The closest edge's depth, with respect to the robot base, among the border edges to the next end-effector position determines the depth in the X axis direction at every step along the contour following path.

IV. EXPERIMENTAL RESULTS

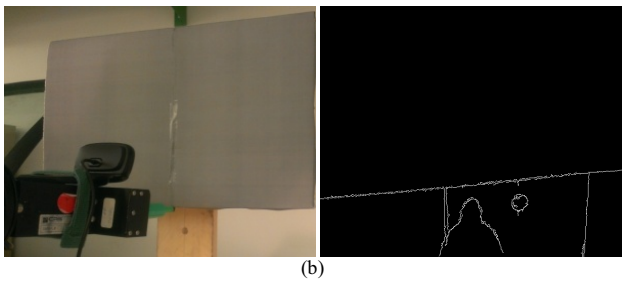
To validate the feasibility of the proposed path planning method, experiments are carried out with a 7-DOF CRS F3 manipulator. In these experiments, a distance of 120 mm is imposed between the robot end effector and the object during the contour following to increase the eye-in-hand camera field of view. A pointing tip with 100 mm length is mounted to the end-effector to determine the accuracy of the robot during contour following at each step (20 mm safety distance from the pointer and object is considered). The eye-in-hand camera used is a 5 MP webcam with 680x480 pixels image resolution.

1) *First Experiment: Following a planar rectangular close contour with sharp edges*

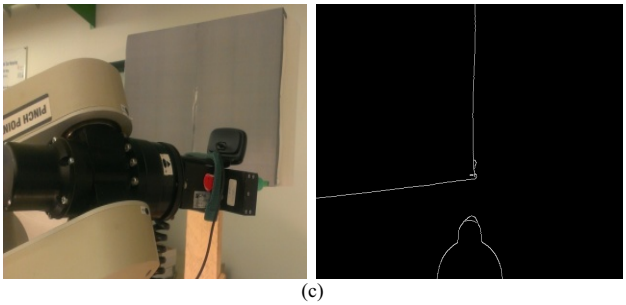
As shown in Fig. 5, a start point (lower left edge) on the contour of the object is automatically selected by the robot using the global information provided from the Kinect sensor, which is also used to locate the overall object and define the primary pose of the end-effector in close proximity of the object contour. The robot is driven to that location. However, at this stage the end effector is not accurately aligned with the contour because of the low accuracy of data collected with the Kinect sensor. The contour definition is then refined from the eye-in-hand camera to guide the robot and more precisely align the pointing tip with the contour (Fig. 8a). As shown in Fig. 8b, although other contours are detected, they are successfully pruned from the map of edge points of interest and the robot continues to follow the correct contours (rectangular gray object). In Fig. 8c, the robot pointing tip reaches the rightmost object corner. In the next step it changes its direction of movement to closely follow the vertical contour on the right side of the object, as shown in Fig. 8d. The smooth transition is made possible given that no further edge is detected on the same horizontal line in the selected direction of movement (here counter-clockwise). This process continues until the robot reaches back to a point close to the start point (lower left edge). In a more general case where the contour is not be a closed contour, the process ends when no contour points are detected from the Kinect sensor and the eye-in-hand camera in the selection direction of displacement.



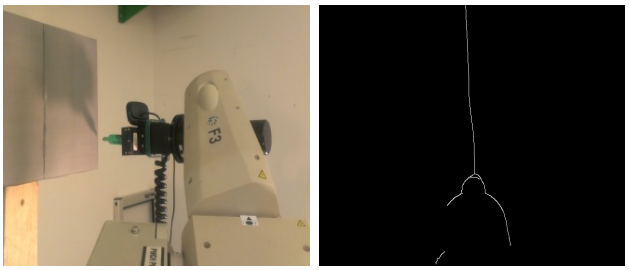
(a)



(b)



(c)

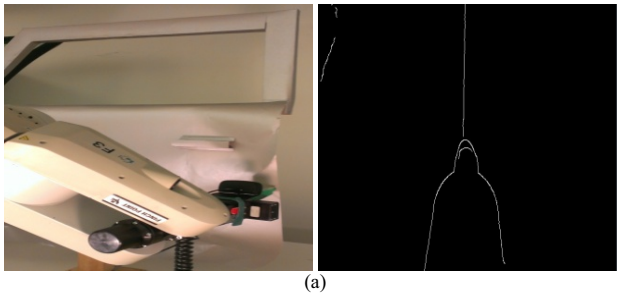


(d)

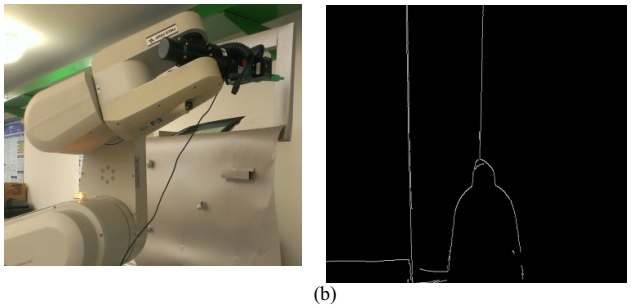
Fig. 8: First experiment: contour following on a planar rectangular object with sharp transitions.

2) *Second Experiment: Following a contour with variable orientations*

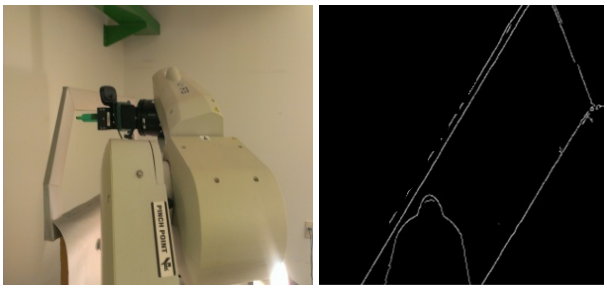
In the second experiment, a more generic object, shaped like an automotive door panel, is considered as it offers several contours with variable orientations, as shown in Fig. 9a-c. The robot successfully follows the contours on this car door model, but exhibits a slightly lower accuracy in some regions. In the upper part of the panel, surrounding the glass window area, two parallel contours (outer and inner) are detected (Fig. 9c). The detected local contours (from the eye-in-hand camera) are first compared with the detected global contours (from the Kinect sensors) to validate the presence of the two contours that are in proximity one to each other and prune away any less significant contour components. Moreover, the priority of selection for the next position to reach to be located over the same line (horizontal or vertical contour) with the current position forces the robot to continue following the contour over which is already lies, in the present case the outer contour of the door window area (Fig. 10b-c). This prevents the robot from transitioning in between the contours located in close proximity and ensures the smoothness of the trajectory.



(a)



(b)



(c)

Fig. 9: Second experiment: contour following on an object with complex shape and variable orientations.

Fig. 10 shows the estimated border edge map (blue contour) extracted from the Kinect data along with the superimposed robot trajectory (red points) when the robot navigates without the assistance of the eye-in-hand camera visual feedback. The robot properly follows the edges detected from the Kinect RGB-D information. However, the detected contour points reveal not to be accurately located over the actual object's contours, mainly because of the Kinect's intrinsic low resolution and of some residual error in Kinect/robot calibration process. Fig. 11 shows the Cartesian distance deviation that appears in between the pointed location and the closest actual location of the contour, manually measured at each step over the contour following operation based solely on Kinect sensor. The average residual error is about 2 cm in

absolute value, and varies in sign depending on the section of the contour that is tracked. This level or error is expectable as it corresponds to the typical error characteristics of the Kinect sensors [10,11]. We also observe a tendency for the end effector to oscillate around the contour as it progresses around the object.

When additional feedback is provided to the system as visual servoing with the eye-in-hand camera, the path of the robot pointer (red points) gets considerably refined, as shown in Fig. 12. It follows the locally detected contour (red contour), which offers a more accurate match with the actual contour of the object than the contour detected from Kinect data (blue contour). We also observe the increased stability and smoothness of the performed path in comparison with that shown in Fig. 11. The gap in between the blue and the red contour locus results from a more accurate localization of the actual with the eye-in-hand camera, which compensates for the limitations in the accuracy of the RGB-D sensor measurements, which tend to slightly distort and expand the object shape and size.

The results demonstrate that the proposed method overcomes the limitation of the previous works on contour following while relying on very affordable sensors for the far view. The robot autonomously detects the unknown object contours using the Kinect sensor information. The end effector is automatically located in proximity of the object to start with. Then the eye-in-hand camera refines the detected contours and further aligns the end-effector with them. Furthermore, the Kinect data is used for the robot to vary its depth in case of non-planar objects, such that the contour is accurately followed in 3D, and not only in 2D.

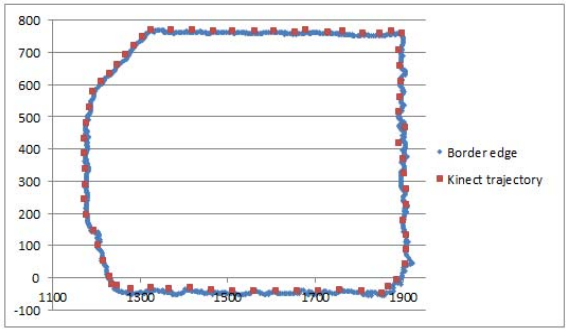


Fig. 10: Estimated border edges (blue edge map) vs robot trajectory without eye-in-hand camera (red dots)

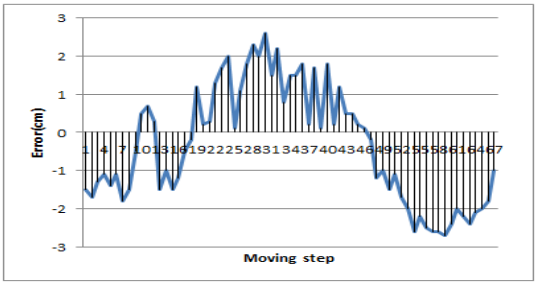


Fig. 11: Cartesian distance in between pointed and actual contour without eye-in-hand camera.

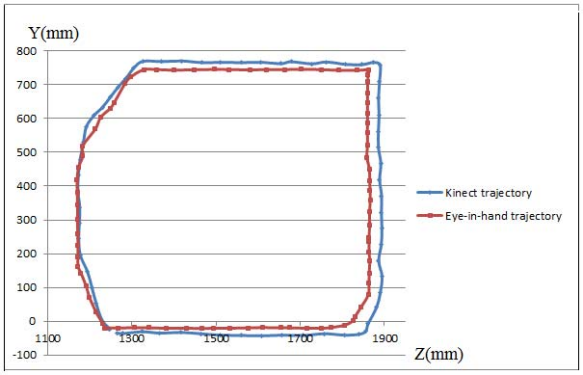


Fig. 12: Robot contour following using Kinect information (blue edge map) vs robot trajectory using visual servoing with eye-in- hand camera (red dots).

V. CONCLUSION

This work presents an adaptive contour following method based on low accuracy RGB-D surface profiling and visual servoing. The vision system consists of a fixed Kinect sensor and an eye-in-hand camera mounted on a robotic manipulator end-effector. The proposed approach builds upon the cooperation between a low-cost Kinect sensor and a standard color camera in order to lead to an accurate and efficient contour following operation with a robot manipulator. The Kinect sensor rapidly collects color images and depth information on the object which provides global knowledge to the robot about the object position, orientation and approximate contours location. This estimated contour provides the primary pose for the robot end-effector to ensure that the contours is positioned and remains within the eye-in-hand camera's field of view while also maintaining close proximity to the object. The eye-in-hand camera mounted on the end-effector provides local information with higher resolution about the actual contour location. A strategy is proposed to use edge points extracted from the eye-in-hand camera, which is looking ahead of the robot pointing tip in the current configuration, to generate feedforward signals to the robot controller under the form of the next contour edge point to reach. The proposed method is experimentally validated on a 7-DOF CRS F3 manipulator. The experimental results demonstrate that the robot successfully detects and follows object contours of various shapes and complexity.

REFERENCES

[1] L. Mi and Y.B. Jia, "High Precision Contour Tracking with Joystick Sensor", *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, vol. 1, pp. 804-809, Sendai, Japan, Oct. 2004.

[2] A.S. Prabuwono, S. Said, B. and R. Sulaiman, "Performance Evaluation of Autonomous Contour Following Algorithms for Industrial Robot", *Robot Manipulators Trends and Development*, A. Jimenez and B.M. Al Hadithi (Ed.), InTech, Mar. 2010.

[3] F. Lange and G. Hirzinger, "Stability Preserving Sensor-Based Control for Robots with Positional Interface",

IEEE Intl Conf. on Robotics and Automation, pp. 1700-1705, Barcelona, Spain, Apr. 2005.

[4] U. Martinez-Hernandez, T.J. Dodd, L. Natale, G. Metta, T.J. Prescott, and N.F. Lepora, "Active Contour Following To Explore Object Shape With Robot Touch", *World Haptics Conference* , pp. 341-346, Apr. 2013.

[5] J. Baeten and J. De Schutter, "Hybrid Vision/Force Control at Corners in Planar Robotic-Contour Following", *IEEE/ASME Transactions on Mechatronics*, vol. 7, no 2, pp. 143-151, June 2002.

[6] F. Lange, P.Wunsch, and G. Hirzinger, "Predictive Vision Based Control of High Speed Industrial Robot Paths," *IEEE Intl Conf. on Robotics and Automation*, vol. 3, pp. 2646–2651, Leuven, Belgium, 1998.

[7] C. Collewet and F. Chaumette, "A Contour Approach for Image-based Control on Objects with Complex Shape," *IEEE/RSJ Intl Conf. on Intelligent Robots and Systems*, vol. 1, pp. 751-756, Takamatsu, Japan, 2000.

[8] H. Koch, A. König, A. Weigl-Seitz, K. Kleinmann, and J. Suchý, "Multisensor Contour Following With Vision, Force, and Acceleration Sensors for an Industrial Robot," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no 2, pp. 268-280, 2013.

[9] W.C. Chang, "Hybrid Force and Vision-Based Contour Following of Planar Robots", *Journal of Intelligent and Robotic Systems*, vol. 47, issue 3, pp. 215-237, 2006.

[10] K. Khoshelham, "Accuracy of Kinect Depth Data", *ISPRS Workshop on Laser Scanning*, pp. 1437-1454, 2011.

[11] R. Macknojia, A. Chávez-Aragón, P. Payeur, R. Laganière, "Experimental Characterization of Two Generations of Kinect's Depth Sensors", *IEEE Intl Symp. on Robotic and Sensors Environments*, pp. 150-155, Magdeburg, Germany, Nov. 2012.

[12] R. Macknojia, A. Chávez-Aragón, P. Payeur, and R. Laganière, "Calibration of a Network of Kinect Sensors for Robotic Inspection over a Large Workspace," *IEEE Workshop on Robot Vision,*pp. 184-190, Clearwater, FL, Jan. 2013.

[13] R. Fareh, P. Payeur, D. Nakhaeinia, R. Macknojia, A. Chávez-Aragón, A.-M. Cretu, P. Laferrière, R. Laganière. R. Toledo, "An Integrated Vision-Guided Robotic System for Rapid Vehicle Inspection," *IEEE Intl Systems Conference*, Ottawa, ON, Apr. 2014.

[14] L. Linsen. Point cloud representation. Technical report, Faculty of Computer Science, University of Karlsruhe, 2001.

[15] G. Garai and B.B. Chaudhuri, "A Split and Merge Procedure for Polygonal Border Detection of Dot Pattern", *Image and Vision Computing*, vol. 17, pp 75–82, 1999.

[16] J. Canny, "A Computational Approach to Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no 6, pp. 679-698, Nov. 1986.

[17] D.H. Ballard and C.M. Brown, *Computer Vision*, Prentice-Hall Professional Technical Reference, 1982.

[18] D. Nakhaeinia, R. Fareh, P. Payeur, R. Laganière, "Trajectory Planning for Surface Following with a Manipulator under RGB-D Visual Guidance", *IEEE Intl Symp. on Safety, Security, and Rescue Robotics*, pp. 1-6, Linkoping, Sweden, Oct. 2013.