Image-driven Drawing System by a NAO Robot

Jose-Maria Munoz, Jose Avalos, Oscar E. Ramos
Department of Electrical Engineering, Universidad de Ingenieria y Tecnologia - UTEC, Lima, Peru

***Abstract*—Co-working applications are increasing in robotics due to the need of manipulating high volume of information in real time. This paper presents a methodology to develop a robot-based drawing system that allows the reproduction of an image shown by an operator. It incorporates cameras for real-time image processing, which uses filters, morphological operations and a topological skeleton to obtain the desired features. A system calibration technique is also proposed to calibrate the coordinates of the robot with respect to the image frame before using an inverse kinematics method that controls the robot motion. The speed of the drawing method depends on the network to which the system is connected. The results of the proposed methodology are fully obtained with a NAO robot.**

Keywords - Baxter robot, Kinect sensor, inverse kinematics, teleoperation.

I. Introduction

Common tasks such as drawing or painting possess a real challenge in robotics since it is not trivial for a robot to autonomously determine the proper path its end effector should follow. An image can be used to guide this motion by being able to reproduce the seen image as closely as possible. This reproduction could lead to autonomous drawing or painting without the need of hand coded trajectories. These tasks can be further applied to industrialized areas such as welding or painting based solely on a shown feature rather than on a pre-defined trajectory allowing for more flexibility in environments where different designs are needed.

Several attempts have been made to try to achieve robot drawing based on different path generators. One of the first attempts used the humanoid robot HOAP-2 [1], where the robot could draw anyone sitting in front of it using primitive techniques for face detection, front edge detection and edge extraction along with trajectory planning. Another effort was proposed in [2] in order to create artistic portraits using the ISAC robot. This robot had special soft hands which could mitigate the drawing of fine humans. It also used an artificial McKibben muscle as an actuator while its stereo vision helped in the 3D rendering of the object. In [3], a robotic system was proposed to draw the aesthetic human face more accurately than the aforementioned robots. This latter robot had only four degrees of freedom, with a camera placed on the body, and was specially designed for this purpose. Therefore, it had an efficient control of feedback and extraction modules of facial features. These methods used hardware specifically designed for the given task and their generalization to arbitrary robot configurations is not direct.

Contrary to previous approaches, this paper presents a generic methodology that can be applied to any type of robot regardless of its kinematic design. No special hardware is needed for the method to properly work. The only needed hardware is a camera and an arm which has enough degrees of freedom for reaching a certain workspace with a constant orientation. The main novelty of the approach consists in the generality of the approach by clearly separating the image feature extraction from the robot control, leveraging the computational power needed for each of these processes. The method is applied to the NAO robot, as shown in Figure 1 making it able to draw what it sees through its camera.

978-1-5386-2278-0/17/$31.00

Fig. 1: NAO Robot setup for drawing

II. Image Features Extraction

The NAO robot has two embedded RGB cameras which are used to obtain the desired images. However, the bare RGB images cannot be directly used to drive the robot motion, but they need to be further processed to obtain a proper format. This section presents the image processing methods used to obtain the desired image features that will be used as target points for the drawing process.

A. Detecting Edges on the Image

The first step in the processing process consists in obtaining edges from the acquired image. To this end, the RGB image needs to be converted to gray scale reducing the three channels to only one, but keeping the main image characteristics. The gray scale image contains details and too many features that cannot be properly handled by the controller of the robot hand, therefore it needs to be simplified. This simplification, which will be useful for drawing, consists in extracting the edges of the image.

To extract the edges, the Canny edge detector was used rather than other simpler edge detectors since it allows more control over the extracted features [4]. In general, it can provide good edge detection, good location and minimum response according to the chosen threshold parameters. The first step in the implementation consists in filtering the image with the first derivative of a Gaussian filter, followed by finding the intensity of the gradients and applying non-maximum suppression. Potential edges are then obtained by applying a double threshold and hysteresis.

After extracting the edges, spurious remaining noise needs to be eliminated and some edges might need to be enhanced. To this end, morphological operations are used to increase the features quality, in general. An opening operation is first applied to eliminate small noise points that appear in the image followed by a closing operation, which tends to undo the thinning of the previous step but without recreating the eliminated noise [5]. The specific mask is task dependent and is mainly experimentally obtained.

B. Topological Skeleton

After the morphological processing, and due to the nature of the image which is arbitrary, the edges can still have a width of more than one pixel. It is necessary to have one-pixel width since these points will be provided as target points for the robot hand motion. Because of this, a topological skeletonization was implemented, which takes the morphological output as input and outputs a single pixel edge [6]. This skeleton is a thin version of the shape and is equidistant to its boundaries, keeping the geometrical and topological properties of the obtained shapes.

After the skeleton is obtained, an initial point in the skeleton is randomly chosen and its pixel coordinates (x_o, y_o) are stored. This pixel is labeled as visited. Then the neighbors of the point are checked in a given direction (e.g. to the right) to determine the next coordinate point. The coordinates of this neighbor are stored as (x_1, y_1) and the pixel is labeled as visited. Once the sequence is closed, that is, the next neighbor is an already visited neighbor, or when no more neighbors are found, there will be a sequence of points (x_i, y_i) which describe the target points in the image frame. Then, another non visited point in the image is chosen and the process is repeated, to generate as many continuous edges as possible.

An important remark is that skeletonizing can add some additional noise to the image, such as single pixels appearing in some parts. To reduce this noise, connectivity is checked. Elements that have less than a given (small) threshold number of pixels are removed since they are considered to be noise.

C. System Calibration

The points (x_i, y_i) obtained in the previous steps are given with respect to the image frame and in pixels. However, the robot needs 3D Cartesian coordinates given in its own reference frame [7] (typically with respect to its torso). To convert from the image frame to the robot frame, the corners of the image are identified with a predefined workspace where the robot should draw. These 3D workspace corners are known values measured with respect to the robot reference frame. The ratio between the width and height of the rectangular 3D workspace, and the width and height (in pixels) of the image itself is used to scale the pixels adjusting them to the 3D world. The image is fitted to the 3D workspace plane as such, keeping the height of the plane constant. These new 3D coordinates will be referred to as $\mathbf{x}_{des} = (x_d, y_d, z_d)$ where z_d is constant.

III. Robot Motion Generation

After extracting the image features from the shown design, a set of desired points are given to the robot for it to follow. As previously mentioned, these points are close enough to guide the robot properly over the drawing surface, since a position-controlled scheme is used and there is no force feedback or arm compliance. Due to this structural restriction, the proper selection of the target points is critical. However, once they are properly chosen, the control scheme tracks them in Cartesian space avoiding motion discontinuities due to the differential method of the approach.

A. Differential Kinematics Scheme

The robot can move either the right or the left arm for reproducing the shown object by following the prescribed points in Cartesian space. To generate its motion, a differential scheme is used. Let $\mathbf{x}$ be the Cartesian position of the robot end-effector (i.e. its hand), and $\mathbf{q}$ be the vector containing the joint configuration. The forward kinematics of the end effector is given by

$$\mathbf{x} = \mathbf{f}(\mathbf{q}) \tag{1}$$

where $\mathbf{f}$ is the forward kinematics function and contains highly nonlinear terms. These nonlinear terms make it complicated to obtain a closed form solution. Some analytic solutions to this problem exist for specific robots, such as the NAO arm [8], but they cannot be generalized, and are robot dependent. Moreover, it is sometimes not easy to automatically choose among all the possible solutions.

Rather than obtaining the analytical solution, this work uses a differential kinematics approach since it is easily generalized to any type of robot once the forward kinematics is known [9] (considering the ease of obtaining the forward kinematics using the Denavit-Hartenberg parameters or specialized libraries that parse the robot model). The differential approach provides the following relation between Cartesian space velocities $\dot{\mathbf{x}}$ and joint velocities $\dot{\mathbf{q}}$:

$$\dot{\mathbf{x}} = J\dot{\mathbf{q}} \tag{2}$$

where $J = \frac{\partial \mathbf{x}}{\partial \mathbf{q}}$ is the end-effector Jacobian. This relation is linear an can be inverted as

$$\dot{\mathbf{q}} = J^{\#}\dot{\mathbf{x}} \tag{3}$$

where $J^{\#}$ denotes the pseudo-inverse of the Jacobian. This work uses the Moore-Penrose pseudo-inverse. In (3) The pseudo-inverse is used instead of the inverse since J is not always an invertible matrix, and is usually not even an square matrix. When the robot is close to a singularity, which can be

checked by continuously monitoring the rank of J, a damped least-squares solution is used to avoid large joint velocities.

B. Cartesian Space Reference

The desired points are provided in the Cartesian space whereas the differential scheme (3) uses $\dot{\mathbf{x}}$. To include the Cartesian information in terms of the velocity, an error needs to be first defined as

$$\mathbf{e} = \mathbf{x} - \mathbf{x}_{des} \tag{4}$$

where $\mathbf{x}_{des}$ is the desired Cartesian position and $\mathbf{x}$ is the current position obtained from the current joint configuration with forward kinematics using (1). Considering constant desired points, the derivative of the error is $\dot{\mathbf{e}} = \dot{\mathbf{x}}$ and therefore (3) can be rewritten as

$$\dot{\mathbf{q}} = J^{\#}\dot{\mathbf{e}} \tag{5}$$

which considers the joint velocities in terms of the variation of the Cartesian error. This variation can now be chosen so as to make the error decrease monotonically to zero as

$$\dot{\mathbf{e}}^{*} = -\lambda \mathbf{e} \tag{6}$$

where $\lambda > 0$ is a real value that guarantees exponential convergence of the error to zero. Then, this error reference (6) is applied to (5) to obtain the desired joint motion.

The final step consists in obtaining the joint positions from the joint velocities. Consider the joint configuration at time k to be $\mathbf{q}_k$, and the joint velocity to be $\dot{\mathbf{q}}_k$. The joint configuration at time $k+1$ is obtained as

$$\mathbf{q}_{k+1} = \mathbf{q}_k + \Delta T \dot{\mathbf{q}}_k \tag{7}$$

where ΔT is the control time. This control time depends on each specific robot but is typically less than 10 ms. Due to the fast control time, the integration scheme shown in (7) is usually a good approximation to the real joint configuration. If a slower control time needs to be used, another integration method would need to be used such as a Runge-Kutta integration, but this is not usual in practice.

IV. Complete Drawing System

The proposed drawing system needs to integrate both of the previously described sub-systems. The whole structure can be better understood with the flow diagram shown in Figure 2. As shown in this diagram, the process was divided in two parts: extraction of the desired points from the provided image, and generation of motion for the robot based on those points. The input image is acquired with a camera, such as the robot camera, and can be a previously drawn picture or a real 3D scene. The image is then pre-processed using the methods described in section II to improve the quality of the image and to obtain the points that will be used as reference. Once the image processing part is done and the Cartesian points of the operational space are obtained, the kinematics of the robot is computed, as described in section III leading to the desired joint configuration.

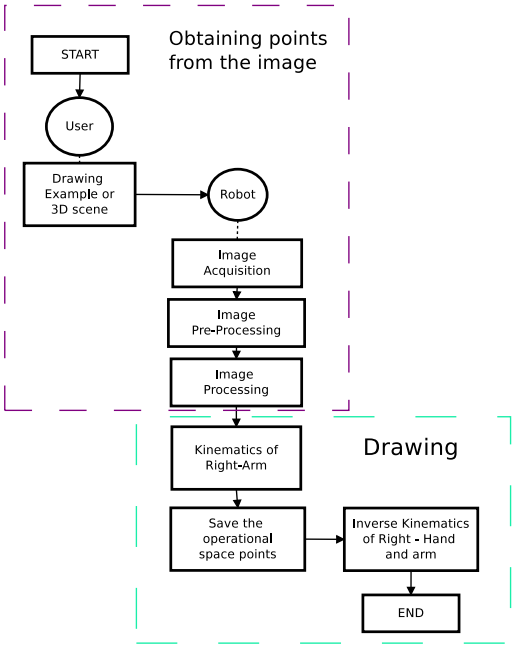


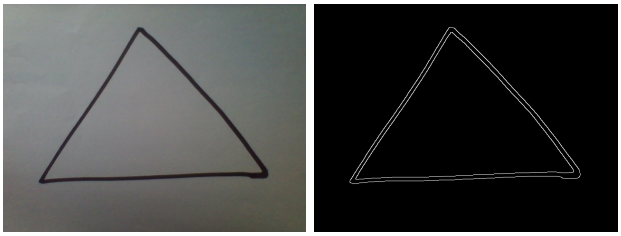
Fig. 2: Flowchart showing the steps that the system follows.

In this work, the framework that was used is ROS (*Robot Operating System*). This is not a limitation since plugins can be developed for any type of robot, and bridges for image acquisition exist. This framework was chosen since it naturally handles robot control signals that can be executed directly on a robot or on a dynamic simulator. Thus, there is no need to use a real robot for tests, although this has been the case here. Also, it can make the image processing and the robot motion completely independent so that software or hardware changes are transparent to each of the two layers.

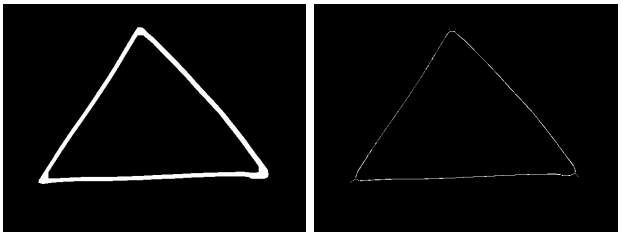
V. Results

The proposed methodology was tested on a real NAO robot. As previously described, ROS was used throughout this work. The image was first acquired with the robot frontal camera using OpenCV in Python through opencv bridge, which enables the use of OpenCV with ROS. This raw image was published in a ROS topic and another node processed the image as previously described. The resulting desired features are then sent to another ROS topic that stores them and updates them as needed. The motion control part is implemented in another node that reads the topic containing the desired points and sends the joint commands to the robot through the NAOqi bridge for ROS.

A result that shows the procedure for a hand made drawing is used as input to the camera is shown in Figure 3a. The results after applying a Canny edge detector with a proper threshold value and a morphological closing (dilation followed by erosion) are shown in Figure 3b, c. Figure 3d shows the resulting image after skeletonization. The desired points are obtained from this latter image, by following the neighboring pixels in a subsequent order, and are fed to the robot controller. The results for the processing of the real 3D scene are shown



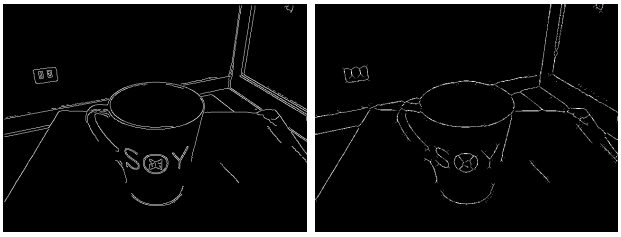
(a) Original Image (b) Edges using Canny



(c) Morphologically closed image (d) Desired skeleton

Fig. 3: Image processing steps for a 2D drawing

(a) Original Image (b) Edges using Canny



(c) Morphologically closed image (d) Skeletonized image

Fig. 4: Image processing steps for a 3D real scene

in Figure 4. As can be seen, the input is a tea cup on a table in an office. The processing steps are similar to the previous case and are shown in parts b, c and d. It can be observed in Figure 4d that in this case it is not straightforward to determine the path that the desired motion should follow. Different non connected paths are generated and the robot has to raise and lower the end effector (marker or pen) several times to achieve the desired result. This is an initial test and further study of the problem can potentially lead to better skeletons, although edge detection is not a trivial problem.

The obtained points are then sent to the robot controller. In this case, the robot kinematics is based on RBDL (Rigid Body Dynamics Library) which parses the URDF model of the robot and provides some basic forward kinematics. However, the rest of the control algorithm is completely developed using the methods shown in the previous sections. An example of the obtained robot motion is shown in Figure 5 where the green ball shows the target point and the red ball the current point. These points are with respect to the wrist since the desired points on the drawing surface are mapped to the wrist by a simple transformation.

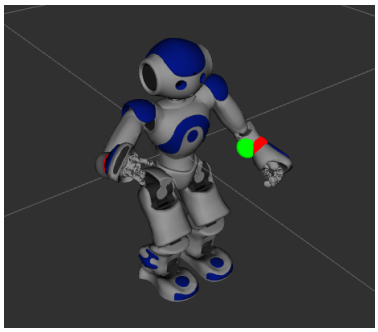


Fig. 5: Motion of the NAO robot in simulation

VI. Conclusion

This paper presented a methodology based on obtaining a sequence of points from a given image, and a inverse kinematics controller for the arm of a robot. Image processing based on edge detection, morphological operations and topological skeletonization was able to render an image from which points could be obtained. A numerical approach for inverse kinematics showed a real-time behavior without several delays. The motion of the NAO robot presents some inherent low-level control problems that deviate from the expected results. Future work will address these problems with a more robust control scheme and a cleaner image processing methodology. Also, writing with both hands at the same time will be explored as a capability that a robot can do but a typical human cannot.

References

[1] S. Calinon, J. Epiney, and A. Billard, "A humanoid robot drawing human portraits," pp. 161–166, 2005.

[2] A. Srikaew, M. Cambron, S. Northrup, R. Peters II, M. Wilkes, and K. Kawamura, "Humanoid drawing robot," 1998.

[3] P. Tresset and F. F. Leymarie, "Portrait drawing by paul the robot," *Computers & Graphics*, vol. 37, no. 5, pp. 348–363, 2013.

[4] P. Bao, L. Zhang, and X. Wu, "Canny edge detection enhancement by scale multiplication," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, no. 9, pp. 1485–1490, 2005.

[5] M. Lybanon, S. M. Lea, and S. Himes, "Segmentation of diverse image types using opening and closing," in *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, vol. 1. IEEE, 1994, pp. 347–351.

[6] P. Maragos and R. Schafer, "Morphological skeleton representation and coding of binary images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1228–1244, 1986.

[7] A. K. Singh, P. Chakraborty, and G. Nandi, "Sketch drawing by nao humanoid robot," in *TENCON 2015-2015 IEEE Region 10 Conference*. IEEE, 2015, pp. 1–6.

[8] N. Kofinas, E. Orfanoudakis, and M. G. Lagoudakis, "Complete analytical inverse kinematics for nao," in *Autonomous Robot Systems (Robotica), 2013 13th International Conference on*. IEEE, 2013, pp. 1–6.

[9] N. Kofinas, "Forward and inverse kinematics for the nao humanoid robot," *Technical University of Crete*, 2012.