TEACHING THE HUMAN SUPPORT ROBOT HOW TO DRAW A STICK FIGURE FROM OPENPOSE

Rafael Marques Braga

University of Miami, Miami FL, 33146, USA rxm403@miami.edu
http://www.rafabraga.com

Abstract. This paper focuses on the development of a drawing algorithm to be implemented in the Human Support Robot (HSR) as well as the use of OpenPose. The goal of the algorithm is to have Toyota's HSR draw a stick figure obtained from a picture of a human posing. OpenPose will process the picture to get the desired pose digitally and the HSR will then attempt to autonomously draw the same pose on a drawing board.

Keywords: Human Support Robot (HSR) · OpenPose · Drawing Robot · Humanoid Robot's behavior · Autonomous Systems.

1 Introduction

In 2012, Toyota Motors Company (TMC) released its Human Support Robot, or, in short, the HSR as part of its partner family of robots. This robot is Toyota's answer to the ever-expanding demand for senior care in aging societies such as Japan or the United States. [1] It features a cylindrical shaped body with a folding arm and omni-directional wheels. This enables the robot to have a compact and highly maneuverable body, which in turn, allows it to navigate through a complex environment and perform complicated tasks such as picking up pieces of paper, objects off the floor or items from the shelf, along a variety of other tasks. With its LIDAR sensor, set of cameras and microphones, force sensors, 11 degrees of freedom and motors as well as its powerful computing hardware of Intel CPU and Nvidia GPU, the HSR holds endless development possibilities [1].

Due to the HSR powerful hardware capabilities and open development tools, this robot is one of the standard platforms used by teams in the RoboCup Home league and World Robot Summit [1]. The same robot is also used for research purposes in multiple Toyota facilities as well as university research labs in the United States, Japan, and Europe.

As a human support robot, many of its required activities to be developed take the available technology to the limit and test the accuracy and precision of the robot's movement as well as the integration with new libraries and tools available for programming today. It is extremely challenging to develop functions to work accurately with the HSR so that it performs correctly in an ever changing, complex environment.

Functions such as teaching the TMC Human Support Robot to recognize and correctly pick up objects, interact in a natural manner with a human or perform tasks such as to clean up a room are just a few of the possibilities for its hardware. The development of these skills, however, require a huge number of resources and are achieved through small steps in order to make the end goal achievable. This paper will discuss how to achieve one of these smaller steps as well as the implementation of a state-of-the-art library, OpenPose, to the HSR.

As the HSR grows as a support robot, it needs to properly learn how to interact with its environment, with help of its vision, sensors and accurate movement. The goal for this project is to have the human support robot to be able to draw a stick figure from a picture of a person taken from its camera. The stick figure drawn should be in the same posture/ form as the person when the picture was taken.

This paper will discuss all the relevant frameworks used for this project as well as relevant published papers, what is already out in the industry for this kind of behavior function (drawing with a robot), how such behavior function was implemented and how it can be further improved in the future.

1.1 Motivation

The idea of having the Human Support Robot drawing a stick figure sounds incredibly simple as it just has to make the arm move in a straight line on a plane. This simple task, however, requires testing of the precision of the TMC odometry controls, it requires that the robot moves its arm in a straight line but also in a straight plane, so the pen does not come out of the paper when drawing the stick figure as well as other tasks.

The implementation of this idea will bring many possible future concepts as the idea originally came from teaching the robot how to draw Xs and Os for a tic-tac-toe game. By building up on the idea of just teaching the HSR how to draw Xs and Os for a simple game, the suggestion of making the HSR draw a human figure in different postures developed. This will force the development of a drawing lines in a plane function.

Furthermore, this idea gives the opportunity to implement a cutting edge, advanced library, OpenPose, to its ROS system so it can work simultaneously with the robot. Having the chance to implement such algorithms with the HSR brings the project an innovative idea that is yet to be implemented. Those are drawing with the HSR as well as the implementation of libraries in the forefront of technologies being used today.

This project takes into aim to step up the interaction with the robot as well as a great feature/ gimmick for the robot to have.

It is important to note that it is not included in the scope of this project to teach the robot how-to pick-up drawing tools such as a pen, to recognize the drawing plane automatically or to tell the robot how to hold a pen. In fact, it is already assumed the robot is holding up a pen/ marker parallel to its arm and the drawing plane is already pre-configured in its environment. Some discussion

on how to detect the drawing plane will be included in this paper, however, no implementation was done.

2 Related Works

The idea of having drawing robots is not new in the recent years. For decades we have had old and extremely capable drawing robots such as our everyday device – a printer. Printers, however, are only capable of printing in a 2D plane, which is a piece of paper. This paper already has pre scripted thickness and position. Drawing in 3D is a much harder task as this requires much more human like skills. Some of these skills required are to measure where a plane is to draw, and acting in a three-dimensional environment but only moving in a two-dimensional plane.

There are also a number of other drawing robots in the market today. A couple of examples are: "The Robot artist for kids", which is a tiny robot with two arms holding a pencil. This robot has the aim of teaching kids how to write and draw. Just like the printer, this robot only works in a 2-dimensional plane. This, however, has to move more motors in combination to draw correctly (2 arms). This will be a challenge to overcome in this project, when drawing with the Human Support Robot.

Having the transformation functions working together in order for the arm, hands, and body to move in synchrony to form a single line can also appear to be quite challenging, as this will require the movement of at least 4 motors to work together to draw a line (hands, arm angle and height as well as the "shoulder" motor.

As for research papers that discuss how to teach a human like robot to draw, some studies have been made including the paper from V. Mohan, "Teaching a humanoid robot to draw 'shapes'". [2] In this paper, the authors discuss a lot on how a robot should draw, including the crucial streams of learning for a computational framework. This includes (1) Motor babbling (self-exploration), (2) imitative action learning (social interaction) and (3) mental simulation. The authors in this paper discuss in detail how to make a humanoid robot draw different shapes and curves, including teaching the robot how to draw letters and other objects which a special, curvy path is necessary. [2]

In the mentioned paper, the authors focus on trying to transfer our human ability to perceive and synthesize 'shapes' to better interact with the world. The author's goal is to transfer the same ability to a humanoid robot, the iCub. The iCub is a one meter tall, child sized robot, developed by various European universities and built by the Italian Institute of Technology as a testbed for research into human cognition and artificial intelligence. Mohan's and Morasso's paper focus on a scenario of the iCub gradually learning to draw or scribble shapes of gradually increasing complexity. This is done after observing a demonstration by a guide and using a series of evaluations of its performance. [2]

In this paper the way the authors were able to teach the iCub how to draw was by having the robot observe the teacher's movement and trajectory while

drawing a shape. This was done by dividing the trajectory taken into a finite set of critical points. These are derived using catastrophe theory: Abstract Visual Program (AVP). These are then transformed into a concrete motor goal (CMG) for the iCub. The next step is then to learn how to synthesize a continuous virtual trajectory similar to the shape that was demonstrated by the teacher. This is done using the critical points defined in the CMG. Finally, an abstract motor program (AMP) is formed by deriving the 'shape' of the self-generated movement – done a forward model output. This model is illustrated in Figure 1. The final step is to then compare the models from the AMP and the AVP to generate an internal performance score and close the learning loop. [2]

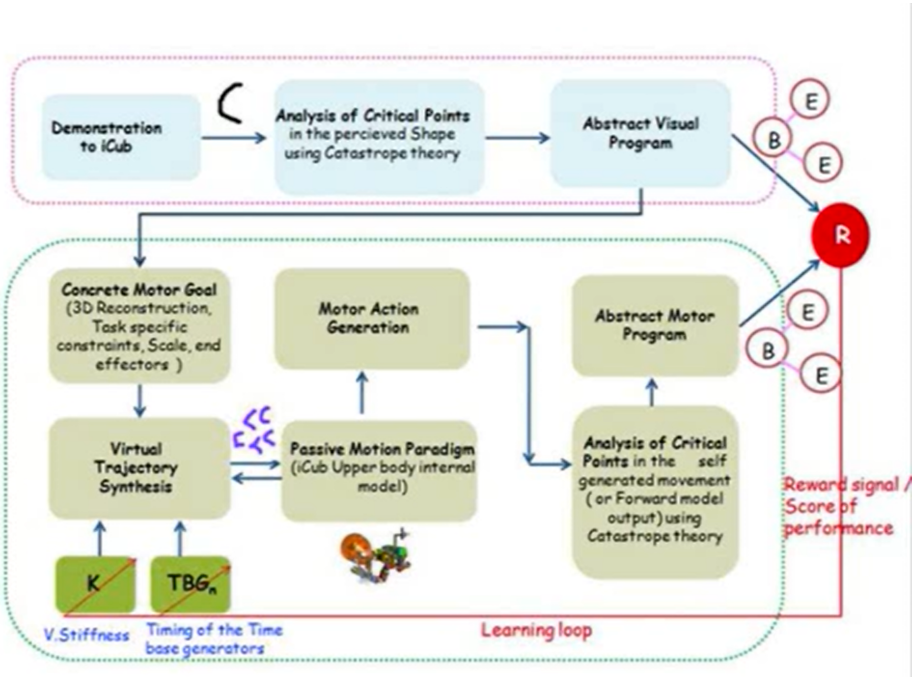

Fig. 1. Mohan's algorithm scheme for iCub's learning and drawing model. [2]

As for the results taken in this paper, it has shown that the iCub was indeed able to learn how to draw different shapes by observing the teacher drawing the same. Even though the results are not perfect, the results are still extraordinary as the system is shown to be robust and self-sufficient. [2]

Figure 2 shows a comparison of the results on how the robot performed after looking at the shape taught. [2] As seen the results are not perfect, but it is definitely an algorithm which can be improved, and lots can be learned from it. The most important take away from the method used in this paper is how making

a path for a robot to draw can be done by defining a finite set of points which when connected make the set shape. This becomes extremely more important when curves are started to be drawn instead of just straight lines.

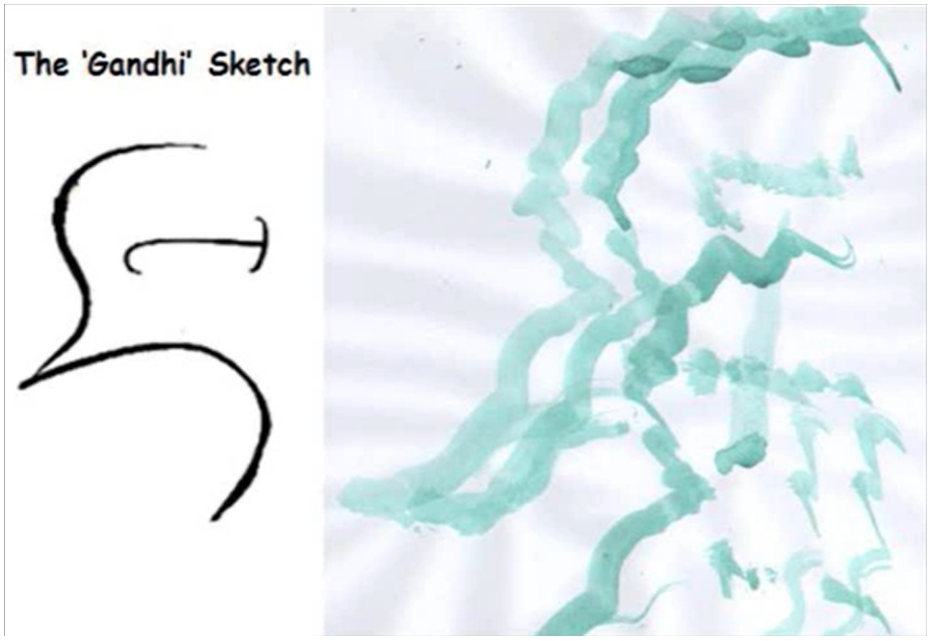

Fig. 2. On the left the drawing figure that is being taught the iCub is shown. On the right iCub's drawing of the same shape is shown.

As for the way Mohan's paper connects to the current research topic, the author was able to develop an algorithm which can make a humanoid robot draw curves and straight lines. This however needs to be taught and first observed from a teacher drawing the shape first. In this paper, the goal is for the robot (HSR) to draw an entire figure autonomously by detecting a human in its camera after receiving the command and then drawing the human's posture on a plane. Algorithms such as the AVP model however, can and should be used in the implementation part of such research topic due to its similarities in goal.

3 Approach

For the implementation of this project it was divided into 2 main parts, OpenPose and drawing. The OpenPose part of the project takes care of detecting the people's pose in a picture and formatting the results to be given to the drawing function. The drawing part of the project takes in the coordinates for the lines

that needs to be drawn and gets the HSR to draw such lines on a given plane. Figure 3 displays the steps taken in this project:

Fig. 3. Steps taken in the project.

3.1 OpenPose

OpenPose is a library that detects people and their poses in an image, video, or a live feed. It can detect up to 135 key points of a human body including the body's posture, hand, facial, and foot key points. It is an industry leading library used to detect 2D multi-person key point detection. When compared to other systems with the same capability of pose estimation (Alpha-Pose and Mask R-CNN), the default version of OpenPose shows speeds anywhere from 4 to 25 times faster, depending on the amount of people are per image. The more people there are in the image, the faster OpenPose is. [3]

OpenPose can also detect people's pose in an image in both 2 dimensions and 3 dimensions. The basic outputs for OpenPose are the basic input image with the key points of the people's poses being overlaid on top of the image as well as a JSON file saving all the key points as an array class. OpenPose has the support to add your own custom output code if necessary. A basic output image of OpenPose is shown below in Figure 4. [3]

As OpenPose is a cutting edge, innovative library, and outputs a perfect example of a "drawing", this library was the one chosen to be implemented with the HSR and to give the output drawing which the Human Support Robot will eventually have to draw on a board. For simplicity, the pose output format BODY_25 was chosen for the output file and detection. The hand and face model were not selected since these parts of the body did not need to be included in the drawing.

Since the outputs are given in a JSON file, this file will then be used as the key points for the drawing. The implementation results of OpenPose and the drawing will be discussed in the next section of the paper.

For the JSON file, the first step for the robot to interpret the results is to change the key points from the person from a JSON format to a data frame format. The JSON file outputs both the coordinates of each key point and its confidence level. [3] In the manipulation function to get the drawing points, the confidence number given in the JSON file was deleted since this number is not needed for the drawing function. Only the key points of the first person were taken out in this example.

Fig. 4. OpenPose sample result overlayed on top of a picture.
[3]

As for the results of the OpenPose, it is important to note that OpenPose gives out the JSON file with the coordinate system being the pixel location of the overlaid human's pose. [3] As an important step for the implementation of the OpenPose acquired posture, these points need to be normalized to the drawing limits set for the HSR. As it will be seen in section 3.2, the drawing plane for the drawing function in the HSR is set for ([-0.39, 0.39], [-0.39, 0.39]). Therefore, all the key points in the JSON output were normalized using the Min-Max equation (shown below) to have a -0.39 minimum and a 0.39 maximum coordinates. The X and Y coordinates were normalized to have the same minimum and maximum.

$$v' = \frac{v - min_A}{max_A - min_A}(newMax_A - newMin_A) + newMin_A$$

The points taken from OpenPose were then passed to the drawing function to draw in the same was as OpenPose reads them to overlay the points in the picture. This is as seen in Figure 5:

So the order of drawing the example above would be to connect point 17 to point 15 in the data frame, point 15 to point 0 and so on. The points number can be known by its row. As the data is gathered, normalized and the drawing order is set, the next step is the drawing function.

3.2 Drawing

In this part of the project, the goal is to make Toyota's Motor Company robot to draw in a straight line without the pen moving out of the plane. This function must take in 4 inputs: the starting and ending (X, Y) coordinates of the line – X1, Y1 are the starting points and X2 and Y2 are the ending coordinates of the line.

Drawing a straight line with the HSR will bring a number of difficulties to the task, some of them being:

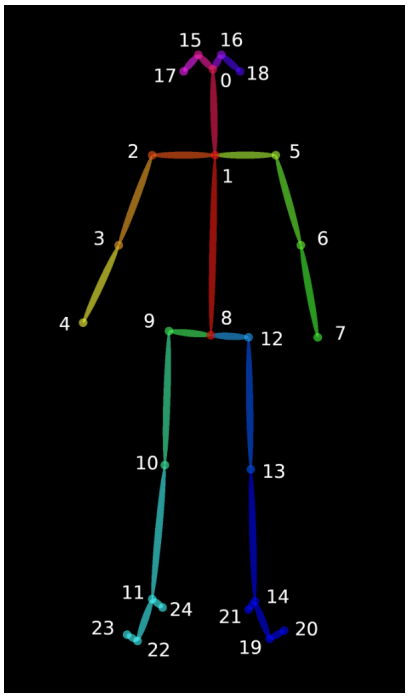

Fig. 5. Order OpenPose output points are drawn along a sample output file with no image overlayed [3]

- Naturally a robot's hand will want to move out of the plane, due to its cylindrical shaped body and natural movement.
- It is needed to make sure that the robot is using enough force to hold the pen on the paper/ board so that the pen actually draws.
- When drawing a stick figure from OpenPose's output, the robot will eventually have to lift its hand from the paper to draw multiple lines coming out of the same point. For example, both legs start at the same point but end at different points. Ensuring all the lines from the drawing figure are connected to each other and start at the same place will lead to a difficulty since it will be hard for the robot to come back in the same exact point in a piece of paper after drawing multiple lines from there. The severity of this problem will depend on how good the odometry system of the TMC's robot is.

To solve some of these problems, the first solution was to ensure a virtual plane was created in the location of the drawing board. This would ensure the robot could do all its transformations and movements based on the (0,0,0) coordinates of the board instead of its body or world coordinates. For example, when told to move to the position (1,0,0), the movement system would figure out a way to move from the initial coordinates to the goal coordinates based on the frame of the board instead of the robots or the world the robot is in. Using this coordinate system solution, the z-plane would be the depth plane and was set for 0.1 depth to be when the robot has the pen on the paper (so force is being applied) and -0.1 to be when the robot lifts the pen from the paper. The (0, 0, 0) coordinates are set as the center of the drawing board with (0.39, 0.39, 0) being the top right edge of the squared shaped drawing plane.

This ends up making the drawing function extremely easier as TMC's standard movement function can be used. The movement function used to set the robot move in a straight line within the piece of paper was HSRB's move _end _effector _pose function. This function took 4 arguments, the x, y, z coordinates as well as the frame for the robot to move in relation to. The move _end _effector _pose function was created by TMC for the exact purpose to move parts of the robot, including the hand, in a straight line, fitting the use of this function to exactly what is necessary in this project.

The drawing function was composed of 4 move_end_effector_pose callbacks for the straight line that must be drawn. The first one was to move the hand to the starting points with the Z-Coordinate set to -0.1 as the hand should be out of the paper when moving to the starting point of the line. Then the 2nd callback function of the move_end_effector_pose is to put the pen on the drawing board, giving the same starting X1,Y1 coordinates but changing the Z-coordinate to 0.1. The 3rd callback function was to actually draw the line, sending as inputs the (X2, Y2) coordinates moving along the 0.1 depth Z-coordinate. Lastly, a move_end_effector_pose function was called to get the robot to take its pen out of the paper. Just telling to move to the (X2, Y2, -0.1) coordinates was enough for the task. Calling the functions as explained using the HSR's simulation results resulted in the robot drawing a perfectly straight line on a single plane as planned.

It is important to note the robot's body linear weight is set to the maximum (100), so it moves the body as little as possible and tries to move its hands the most so it does not hit any objects nearby. The looking_at_hand constraint is also set to true so the robot looks natural when drawing, actually looking at its hand as a human would do.

4 Results

Above, the implementation of each function used was explained. Next, the final results the robot drew are displayed.

Starting with the picture used as the input, figure 6 (left side) was used with Open Pose's processed output shown to its right:

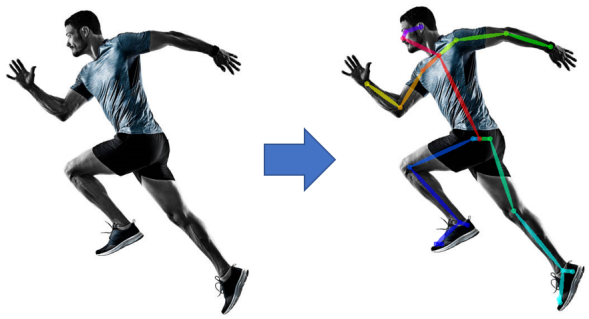

Fig. 6. Picture used as input for Openpose shown on the left and OpenPose processed output on the right

Having the right keypoints of the humans posture, the next step is to get the JSON file with just the points and get the robot to draw OpenPose's output. The final drawn output from OpenPose (left) and the robot (right) are shown in the image below.

As seen, the HSR was able to draw the picture from the left almost exactly as displayed, showing pretty accurate results. Taking just around 1 minute and 36 seconds to do the entire drawing, the movements are not slow either. This method was not yet tried on the actual HSR to see how accurate the drawings are.

For visualization purposes, a line marker was drawn to show the movement of the pen in the board and to see the actual results of the drawn image. The line, in green, showed the same movement as the robot's hand since the same starting and ending coordinates were given.

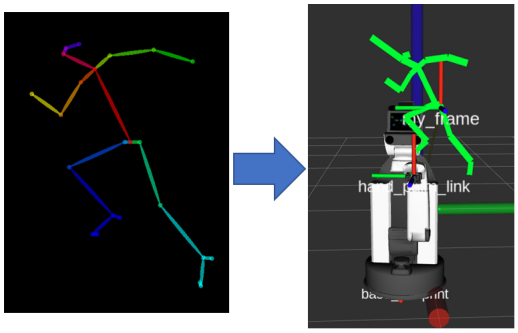

Fig. 7. Only OpenPose's result shown on the left. HSR final drawing of the same pose shown on right

5 Conclusion and Future Improvements

As seen, using today's technology, it is possible to get Toyota's Human Support Robot to draw a stick figure taken from a picture or from a live video using its camera using OpenPose. This project, however, still has multiple weaknesses and multiple steps which can and should be improved.

The next step that should be naturally taken when continuing with this project is to actually implement the algorithm developed in the HSR. This would highlight the problems and bugs in the algorithm.

Another natural improvement that can be done is on how the robot is drawing a straight line. If the line ends up not being straight when the algorithm is implemented, a similar solution as the one seen in Mohan's paper studied can be implemented by selecting some key halfway points in the line. By dividing the line into further key halfway points and connecting them in the end into one line would make the robot draw in straighter line.

By further developing this algorithm, there are also a number of other functions and implementations that can be done. Some of which are having OpenPose work in the actual HSR, having the robot actually interact with the user or further develop the drawing algorithm so it can draw Xs and Os, curves and eventually full scale drawings.

Even with all the improvements that could be made, this project was an excellent opportunity to explore Toyota's Human Support Robot capabilities, the complex ROS interface and the innovative OpenPose library. The final results showed an excellent gimmick and an awesome feature for the HSR to have and naturally interact with humans.

References

1. Spectrum, IEEE. "Human Support Robot." ROBOTS, 22 Oct. 2019, https://robots.ieee.org/robots/hsr/.

2. Mohan, V., Morasso, P., Zenzeri, J. et al. Teaching a humanoid robot to draw 'Shapes'. Auton Robot 31, 21–53 (2011). https://doi-org.access.library.miami.edu/10.1007/s10514-011-9229-0
3. Z. Cao, G. Hidalgo, T. Simon, S. -E. Wei and Y. Sheikh, "OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 1, pp. 172-186, 1 Jan. 2021, doi: 10.1109/TPAMI.2019.2929257.