# Autonomous System for Performing Dexterous, Human-Level Manipulation Tasks as Response to External Stimuli in Real Time

Ana Neacşu(✉), Corneliu Burileanu, and Horia Cucu

Speech and Dialogue Research Laboratory, University Politehnica of Bucharest, Bucharest, Romania
{ana.neacsu, corneliu.burileanu, horia.cucu}@speed.pub.ro

**Abstract.** The system solves a complex puzzle, namely a Pyraminx (Rubik's pyramid), demonstrating the degree of movement complexity that the Kinova robotic arm can achieve. The system is composed of three important parts: the first one's main purpose is to capture real time images from the Kinect sensor and to process them into input data for the second module. The second part, the core of the system, performs all necessary computations in order to make a movement decision based on the available data. The third part represents an interface with the robotic arm, transposing the decision from the second block into pure movement data, passed to the Kinova's controller.

**Keywords:** Image recognition · Shape recognition · Color detection
Kinect · Robotic arm · Clustering · Autonomous system

## 1 Introduction

Nowadays, society tries to accept and integrate persons with various disabilities. They comprise an estimated population of one billion people globally, of whom eighty percent live in developing countries and are overrepresented among those living in absolute poverty.

The core of this project is the robotic arm created by Kinova robotics to aid people battling disabilities. For someone with severe motion disability some trivial tasks, such as picking up a glass of water may represent a challenge [15]. In this context, it is imposed to find an efficient solution to give these people some independence. Hence, the need to develop a system capable of performing human level motions.

The paper is organized as follows: Sect. 1 presents the motivation, the objectives and the outline of this thesis, along with some technical details about the hardware used to develop this project. Chapter 3 is the first chapter that illustrates contributions of the author of the thesis. It describes the data acquisition process and the image processing methods that were approached. Chapter 4 deals with the development of an algorithm that solves the Pyraminx puzzle. Chapter 5 focuses on the implementation the actual moves of the robotic arm. Finally, Chapter 6 summarizes the main conclusions of the thesis.


O. Fratu et al. (Eds.): FABULOUS 2017, LNICST 241, pp. 246–252, 2018.
https://doi.org/10.1007/978-3-319-92213-3_36

### 1.1 Objectives

In this context, this thesis aims to present a system capable of performing a given task without any kind of human intervention. We want to achieve human-like dexterity – the Kinova robotic arm is equipped with six degrees of freedom, having the capacity to accomplish manipulation tasks that require high dexterity. In order to prove this point, we will use the robotic arm to solve a Pyraminx, as response to external stimuli, represented by images captured with a Kinect module in real time.

### 1.2 Hardware Technologies

In this paper, we use the Jaco 3 model, produced by Kinova Robotics. Launched in 2010, JACO is a six-axis robotic manipulator arm with a three-fingered hand. This device significantly improves the lives of persons with reduced mobility by assisting anyone with an upper body mobility impairment to perform complex actions. This robot has a total of six degrees of freedom, it is made of carbon fibre structure, it is light weighted and it can reach the floor with standard installation on wheelchair. The gripper offers the option of using two or three fingers. Each finger is covered with high friction rubber pads, making grasping objects easy [1, 2].

## 2 Modeling the Initial Setup of the Pyraminx

Pyraminx is a puzzle in the shape of a tetrahedron working on the same principle as a Rubik's cube. It consists in 4 axial pieces, that have octahedral shape, and they can only rotate around the axis they are attached to, 6 edges, that can be permuted in any direction and 4 trivial tips, that can be twisted independently. To permute its pieces, the Pyraminx must be twisted around its cuts [4].

### 2.1 Shape Recognition and Color Detection – Method I

The purpose of this module is to create a matrix of colors that describes the initial state of the puzzle that will serve as input for the solving algorithm. Figure 1 illustrates how the system works:

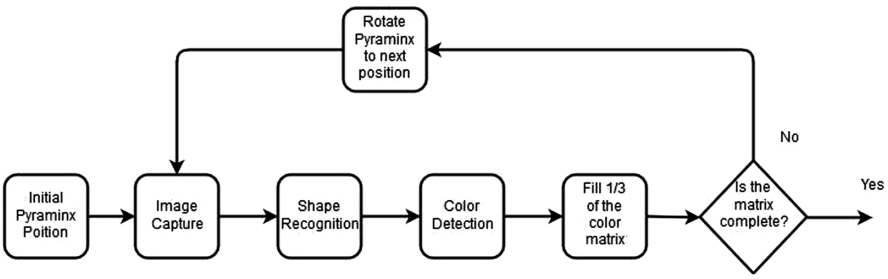


**Fig. 1.** Modeling initial setup

### A. **Image Capture**

To be able to create a matrix that contains all the colors from the pyramid, is not enough to have only one image of the puzzle. As our objective is to create an autonomous system, we didn't consider the option of moving the Kinect sensor manually to take pictures from different angles. The solution that we've implemented consists in rotating the stand that is holding the pyramid using a servo-motor and a Hall sensor connected to an Arduino board, as seen in Fig. 2 [14].

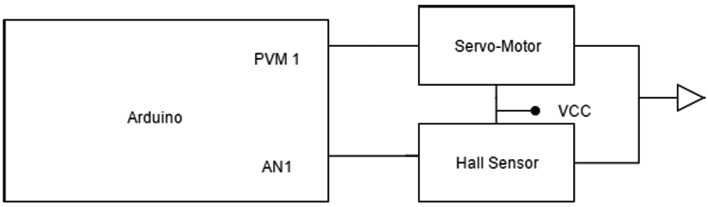


**Fig. 2.** Servo-motor configuration

The system works as follows: the Kinect sensor takes a picture, the servo motor rotates the pyramid at 2.09 rad, then a second photo is taken. To be able to obtain all the colors from the puzzle, three photos are needed, one for every lateral face (the top face appears in all of them).

### B. **Shape Recognition**

Object detection and segmentation is the most important and challenging fundamental task of computer vision. It is a critical part in many applications such as image search, scene understanding, etc. However, it is still an open problem due to the variety and complexity of object classes and backgrounds [13].

The images obtained at the previous step must be processed, to extract the color of every triangle. In order to do that, we used a shape recognition algorithm that detects the vertexes of all the triangles from the image. The shape recognition is implemented using OpenCV library and consists of:

- HSV conversion – the image we capture using the Kinect is in RGB. We transposed it to HSV, because in this domain it is easier to perform color detection. The biggest problem was to detect the yellow color, because it was very close to the white edge. We obtained the best results using the Hue component [10].
- Canny edge detection - it is a multi-stage algorithm to detect a wide range of edges in the image [1].
- Triangle detection – we filter the edges found on the previous step and keep only the forms that can be approximated with a triangle [13].
- Filter out duplicate triangles – there is the possibility to have some duplicate vertexes, so we filter them to obtain the number of triangles that we need.

### C. Color detection

After the previous step is performed, color detection becomes a simple task: using the coordinates of the three vertexes of all the triangles, we can find the middle point of each of them. We perform color detection on a small area around the middle of each triangle, using the HSV conversion [7]. Finally, after all the processing the program will return a $5 \times 12$ color matrix. Each row of the matrix represents a row of colors on the pyramid, and each 3 rows form a face of the puzzle.

**Remark 1.** Using this method, we have obtained good results only in certain conditions: natural light and white background. Because the colors of the puzzle are not matte, if there is too much light, it will be reflected; in this case the recognition task becomes much more difficult and the error rate increases significantly.

From the experimental point of view, we have observed that this algorithm does not distinguish between the yellow color of the triangle and the white contour of the Pyraminx, which lead to frequent and significant errors in the recognition task, even in optimal light conditions. Hence, we decided to try another method, presented in the following sub-chapter.

## 2.2 Shape Recognition and Color Detection – Method II

To overcome the disadvantages from the previous method, we tried a totally different approach, based on machine learning techniques, starting from the observation that, depending on the light and angle, the colors of the pieces have very different properties, covering a wide range of shades. So, for each color we defined several classes, to cover all the possibilities. Figure 3 shows how we obtained the color model. The idea is to train a system that can separate the colors form the Pyraminx, considering the colors to be discrete random variables. The separation is done based on two parameters of the image: the mean and the standard deviation [11]. In this method, the whole image is analyzed. Afterwards, the color detection is performed and the last step consists in shape recognition.

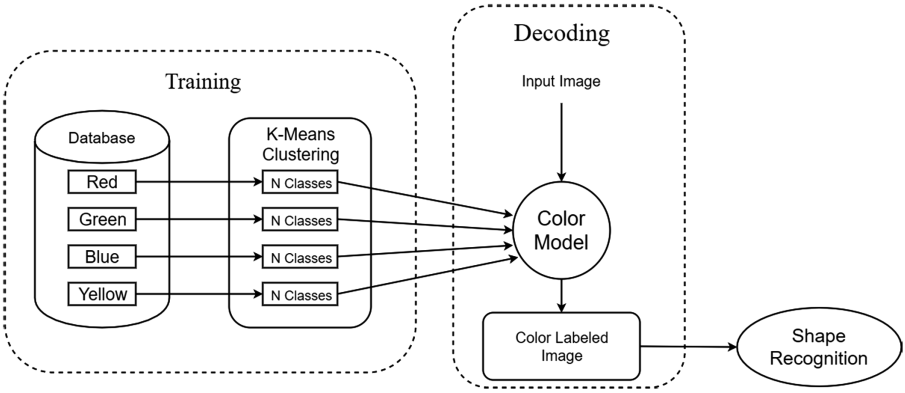


**Fig. 3.** Color detection algorithm

### 2.3 Data Acquisition and Clustering

To apply such an algorithm, we needed a database consisting in different images with the Pyraminx, which we created using the Kinect Module. The database contains 100 pictures of the puzzle, taken in different light setup (natural light and different neon light). Next, we have separated all the pieces from all the images based on their colors, using Paint.net, resulting in four images (one for each color) containing all the useful information from the database. These images were used for training a system that discriminates colors, based on two parameters: Mean and Standard Deviation [5].

We used K-means algorithm as a clustering method to separate each color in different classes. This vector quantization process derived from the signal processing domain. In data mining the method is very popular for cluster analysis. Parsing n observations into K groups named "clusters" is made based on the proximity of the observation to the prototype of the array [6].

**Remark 2.** Using this method, we have obtained better results than with the previous one, but still there were problems regarding the recognition of the yellow color, because it reflected most of the light. Hence, we decided to replace the shiny stickers from the Pyraminx with matte ones and use a unicolored background when the Kinect module takes the pictures. This way, the color recognition works perfectly, regardless the light conditions.

## 3 Solving Algorithm

In order to permute all the pieces, using a single arm and having the pyramid fixed on a stand, we developed an algorithm that will solve the puzzle using only 4 main moves, as showed in Fig. 4. We defined the moves in only one direction (clock-wise for move 1 and 2 and counter-clockwise for move 3); the reverse move is equivalent with two successive moves.

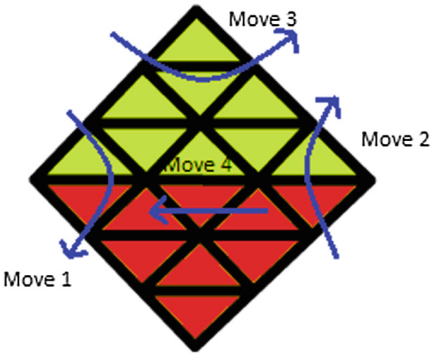


**Fig. 4.** Pyraminx moves

Using only these moves in different combinations, one can solve the Pyraminx in all situations. The algorithm we developed works as follows:

- *Align the Centers and Corners* - The first step in solving the Pyraminx is to know, even if the pieces are scrambled, what color corresponds to each face. There are two easy ways to figure out what is the color of one face: if one can match three center pieces having the same color on one face, that is the color of the face. The other solution is concentrating on the corner pieces. All three corner pieces from a face can be positioned to match color, and that is the color of the face. The next problem is to correctly position the center pieces, and that is done using a simple function. The verification must be done over just one face. If the centers are correctly positioned on one face, all the other center pieces will be in place. The same principle is applying to corners [3, 12].
- *Solve the Top Face* - After all the centers and corners are in place, the next step is to completely solve one face. This is done by placing the 3 edges corresponding to the color of the face we want to solve on their correct position. There are 12 different possible solution for each piece. The first possibility that we are considering in this algorithm is the one in which the desired side is placed on the top face, but not in the right position. In this case, the piece must be lowered on the second layer, and later positioned correctly on the first Layer. Depending on its position, a different combination of moves is required to lower a certain piece. After all the sides are placed somewhere on the second layer, we defined some fundamental sets of moves, that used to move a certain side into the right position [8].
- *Placing the Lower Edges* - After the top piece is solved, there are only three lower edges that can be out of place. Depending on their position, four different cases can occur, and for each a specific algorithm will be applied:
  - Case A: In the most fortunate case, after solving the top face, all the lower edge pieces are in the right place and the pyramid is solved.
  - Case B: one edge piece is in the correct and the other two need to be flipped
  - Case C: one edge pieces is wrong and the other two have one color that lines up with the adjoining side.
  - Case D: All three edge pieces are completely wrong but when you turn the top corner, they will line up with the bottom perfectly [9].

## 4 Performing the Moves

This part represents the interface with the robotic arm, transposing the set of movements computed by the algorithm into pure movement data, passed to the Kinova's controller. The input will consist in a string of numbers, each number being correlated with one of the four moves.

Since all the moves are complex and require high precision, the position of all the actuators are updated during a move, so the action is smooth and as natural as possible.

Controlling the robotic arm requires two steps: manipulate the arm using the joystick and then program the hand to automatically perform the action, using functions from the *.dll* file provided by Kinova, which receives as parameter an angle (how many degrees the actuator rotates from its previous position).

For the movement to look natural and coherent, the actuators are updated at different moments of time, synchronization representing one of the greatest challenges of this project. For each move, the arm starts from a default position, called home position and the final position of the actuators was experimentally found, using the joystick and the monitor function of Kinova Development Center API. By subtracting this new coordinates of each actuator from the values from home position, we deduced how many degrees each actuator must rotate, which is the parameter we need in our code.

## 5 Conclusion and Further Development

Our project represents just a small example of what one can do using a robotic arm as complex and efficient as the one from Kinova.

In the future, we want to develop a system that reacts to different types of real time stimuli: control the arm with vocal command.

Another improvement we would like to implement in the future implies equipping the arm with various sensors to improve furthermore its precision and autonomy. Interconnecting two or more such robotic arms, one can create a robot capable of performing almost all the moves that a human being can do.

## References

1. Kinova robotics website. http://www.kinovarobotics.com
2. Kinova robotic Jaco User Guide. http://www.kinovarobotics.com
3. Pyraminx Puzzle. http://www.nerdparadise.com/puzzles/pyraminx
4. Rubik's Triangle. http://www.ruwix.com/twisty-puzzles/pyraminx
5. Hartigan, J.A., Wong, M.A.: Algorithm AS 136: a k-means clustering algorithm. J. Roy. Stat. Soc.: Ser. C (Appl. Stat.) **28**(1), 100–108 (1979)
6. Kanungo, T., Mount, D.M., Netanyahu, N.S., Piatko, C.D., Silverman, R., Wu, A.Y.: An efficient k-means clustering algorithm: Analysis and implementation. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 881–892 (2002)
7. Sural, S., Qian, G., Pramanik, S.: Segmentation and histogram generation using the HSV color space for image retrieval. In: Proceedings of the 2002 International Conference on Image Processing, vol. 2, pp. II–II. IEEE (2002)
8. Frey, A., Singmaster, D.: Handbook of Cubic Math. Lutterworth Press (2004). ISBN 9780718825551
9. Beasley, J.: The Mathematics of Games. Dover Publications (2006). ISBN 0486449769
10. Bear, J.H.: What is the HSV. https://www.thoughtco.com/what-is-hsv-in-design-1078068
11. Li, F.-F.: Machine learning in computer vision. http://www.cs.princeton.edu/courses/archive/spr07/cos424/lectures/li-guest-lecture.pdf
12. Joyner, D.: Adventures in Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys, 322 p. Johns Hopkins University Press (2008)
13. Forsyth, D., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall, Upper Saddle River (2011)
14. Han, J., Shao, L., Xu, D., Shotton, J.: Enhanced computer vision with Microsoft kinect sensor: a review. IEEE Trans. Cybern. **43**(5), 1318–1334 (2013)
15. Hendrich, N., Bistry, H., Zhang, J.: Architecture and software design for a service robot in an elderly-care scenario. Engineering **1**(1), 027–035 (2015)