CS 684 Course Project

Rangoli Bot

Team 1

Sarbartha Sengupta (10305903) [Team Leader]
Kaustav Das (10305024)
Kishaloy Halder (10305022)

Under the kind guidance of
Prof. Kavi Arya and Prof. Krithi Ramamritham

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai-400076

13th November 2011

Table of Contents

1 Introduction

Rangoli is a traditional folk art of India. In festive seasons colorful rangolis are drawn on floor at almost every place across India. This project is highly motivated by the technique of making rangolis. Currently in this project we aim to make the robot draw very basic rangolis which can be subsequently extended to print posters, flexes. Our technique of drawing rangoli is highly inspired from the way an artist draws or the way a painter paints on the canvas. When an artist draws something she knows where exactly her hand is and what is to be drawn at that position. In our methodology also we have introduced a negative feedback system in order to continuously determine the position of the bot and paint accordingly. Our autonomous rangoli making robot can be thought as an abstraction of a printing machine, and is proposed to create large rangolis in malls etc.

1.1 Definitions, Acronyms and Abbreviations

FB5, ATMEGA, Scilab, Esterel, ZigBee.

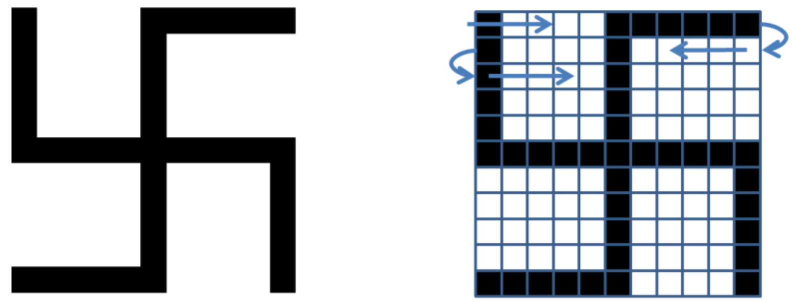
2 Problem Statement

The main objective of this project is to draw a given image of rangoli on floor using colored powder. This bot can be very useful where we have to draw same rangoli in different places. The overall solution approach is discussed below.

A Scilab program takes an input image (Figure 1(a)) and converts it into a binary image. This binary image (a matrix of 1's and 0's) is sent to the robot. The bot reproduces the binary matrix on the floor by dropping rangoli powder on the floor. The bot draws one row at a time in a to-and-fro motion.

The movement of the robot is such that it should not disturb the part already drawn. The movement is a row-wise matrix traversal of the image matrix (Figure 1(b)). The bot will draw same colored consecutive pixels in a row with one line segment (Figure 2).

The bot should draw each row parallel to each other. In order to do that the bot takes feedback from a webcam located at the top of the arena about the current position of the robot.



(a) Input Image (b) Row-wise Matrix Traversal

Figure 1

Figure 2

3 Requirements

3.1 Hardware Requirements

- Firebird 5 (x1)
- ZigBee Module (x2)
- Color dispenser (x3)
- Vibrator motor (x1)
- Resistance 100 Ω (x1)
- Web-camera (x1)
- Servo motors (x3)
- Aluminium Frame
- Colored Paper (Black, Red and Blue)
- Acer laptop – with Windows XP (x1)

3.2 Software Requirements

- Winavr 4.0
- Esterel Compiler
- Scilab 4.0
  - SIVP toolbox 5.0.2
- X-CTU
- Oracle Virtual Box

4 Implementation

4.1 Functionality

1. **Representation of Image -** A jpeg/bmp image will be given to the Scilab program (inputImageProceesing.sce) which will convert the image into a 17x7 matrix with run length encoding containing only 1's and 0's.

2. **Sending the Image -** The run-length encoded image matrix is transferred to Bot using the ZigBee module.

3. **Drawing the Image -** The bot will start drawing the image on the floor in a rasterized to and fro fashion as described in section 2.

4. **Drawing line segments -** The run length encoding algorithm is such that if it finds multiple consecutive pixels of the same color along a row then instead of drawing it pixel-wise it will draw a smooth line going through all of them.

5. **Ensuring Smooth flow of Colored powder -** We have installed a vibrator motor on the body of the color funnel in order to ensure smooth flow of the colored powder whenever the lid of the funnel is opened. The vibrator motor (figure 10) requires a DC voltage of 5V which was taken from a power-slot provided on the bot itself just beside one of the IR sensors. We had to use a carbon resistor of 100Ω in series with it in order to reduce the vibration.

6. **Determination of co-ordinates of the bot –** We have used a camera hanging on the top-centre of the arena looking for the exact position of the bot within the arena. As seen in figure 7 there were two markers of colors respectively red and blue on top of the bot to detect its position as well as orientation. Depending on the resultant orientation the required movement signal is sent to the bot using a ZigBee module (refer to negativeFeedbackSystem.sce). This works as a virtual **Global Positioning System** (GPS) to guide the bot in the course of drawing. Different signals sent by the ZigBee module are demonstrated in figure 9.

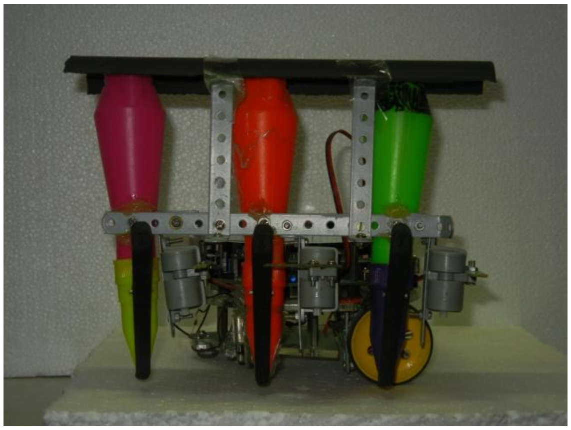


Figure 3: Position of Servo motors and color dispenser

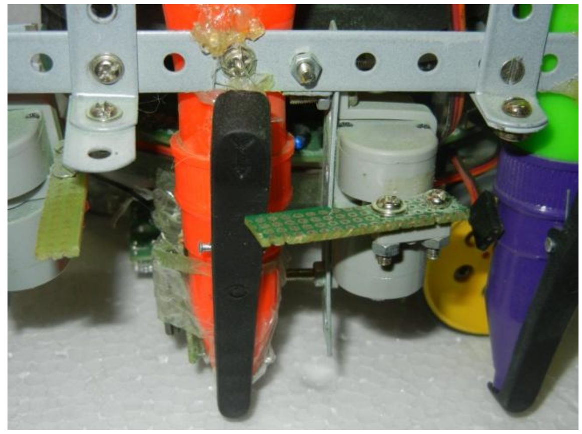


Figure 4: Funnel Blocking Mechanism using Servo Motor

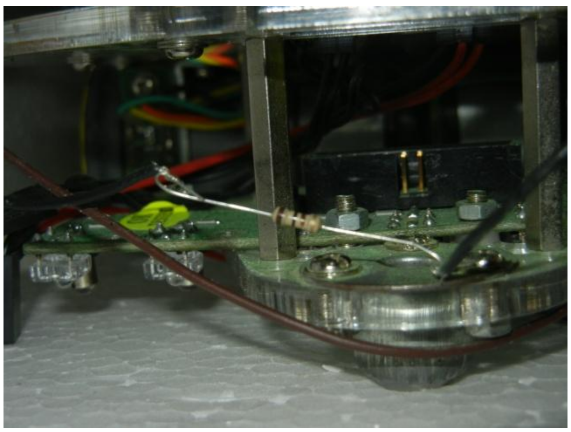


Figure 5: Resistance to control vibration of motor

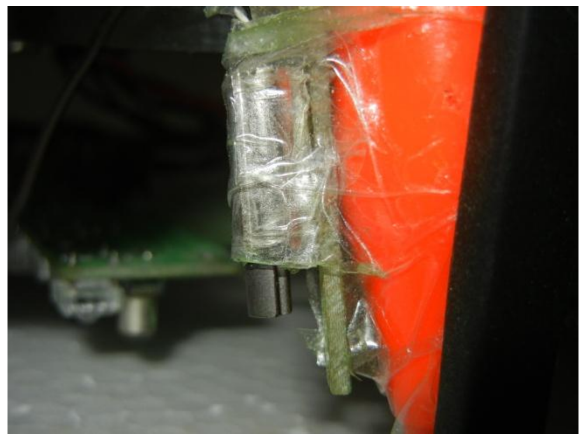


Figure 6: Installation of Vibrator motor to ensure smooth flow of colored powder

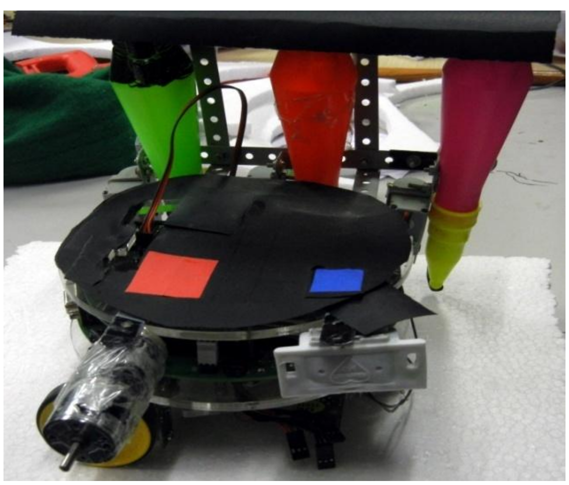


Figure 7: Position of colored markers on top of the bot

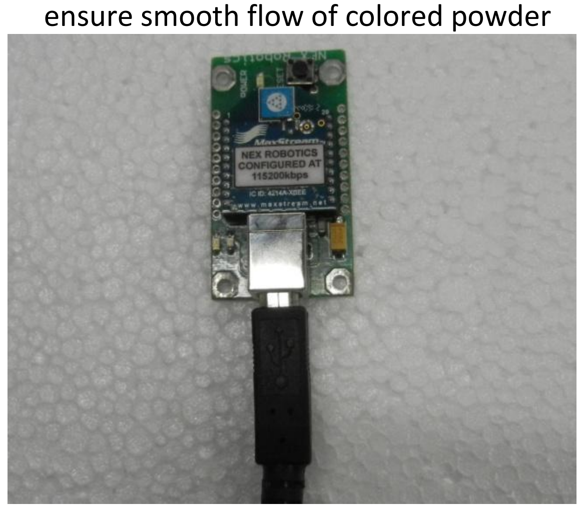


Figure 8: ZigBee module

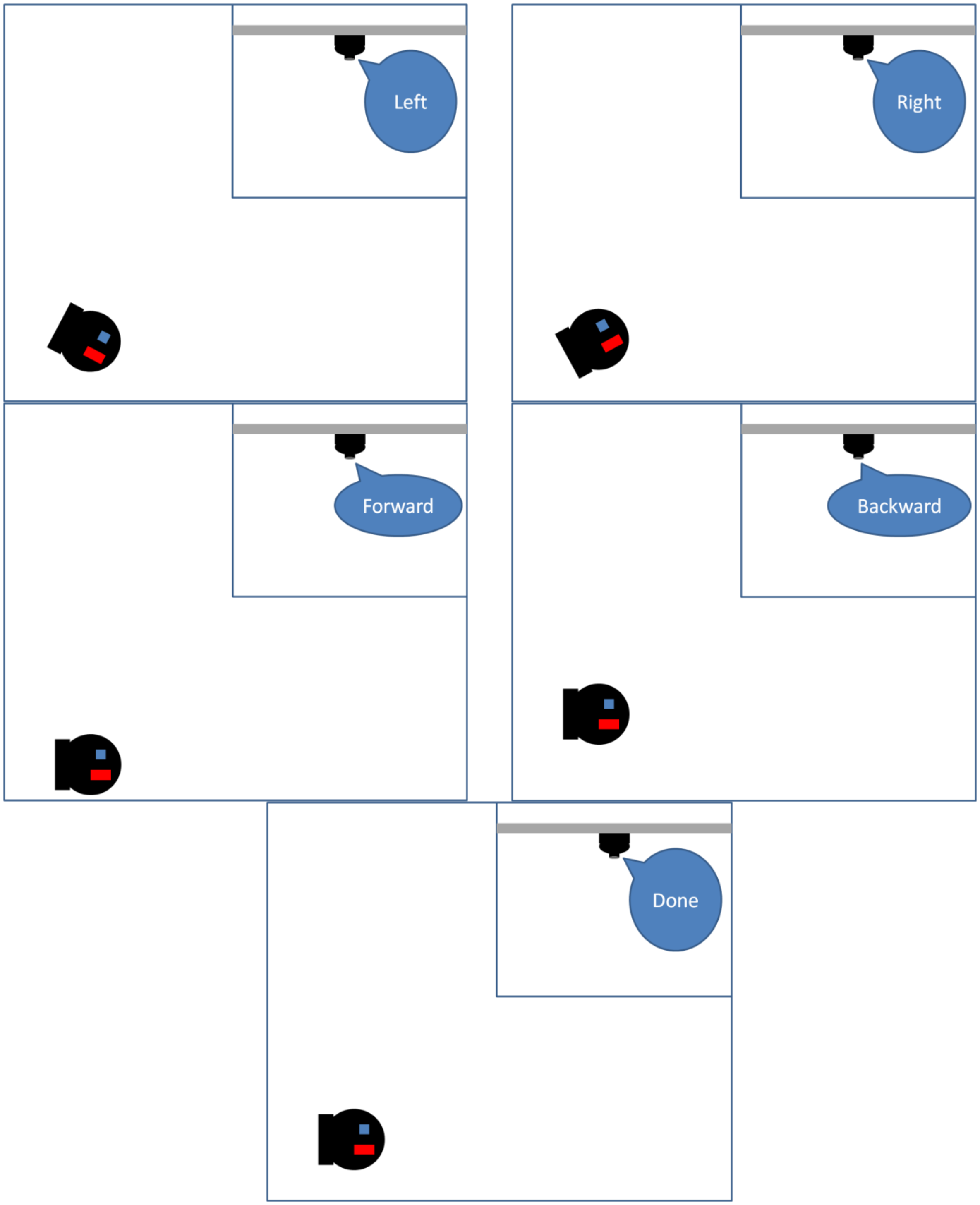


Figure 9: Different orientations of the bot and corresponding signals from the camera

5 Testing Strategy and Data

We have tested with simple binary images and observed the behavior of the bot. Following are the observations.

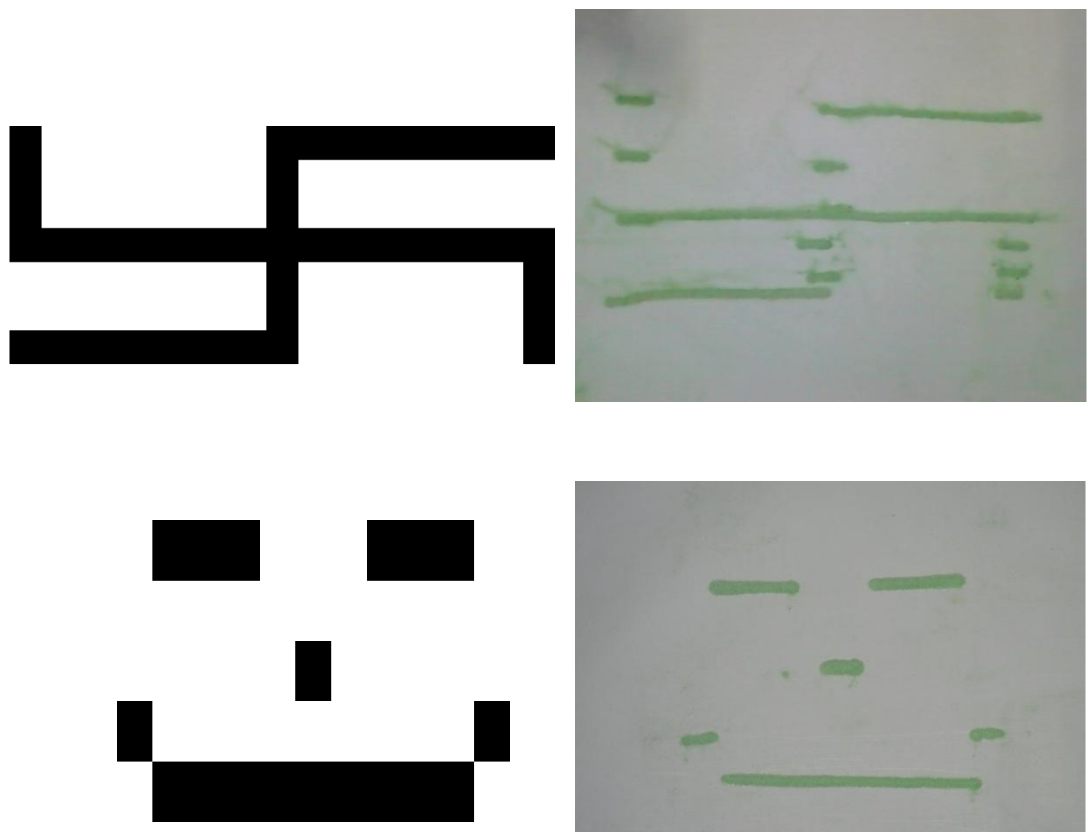


Figure 10: Left: Input images, Right: Images drawn by the bot

We observe that the obtained images resemble closely the input images. The notable difference is that the distance between two consecutive rows is greater in the drawn images than that of the input. This was done in order to eliminate the chance of treading on the portion already drawn. Small distances resulted in overlapping of consecutive rows and thus distorting the entire figure.

We also tried to draw rangoli changing the background color of the floor. But due to some automatic white-balancing taking place in the camera, the pixel color values of the colored markers were getting jeopardized leading to incorrect detection of the orientation of the bot.

6 Discussion of System

6.1 Things worked out as per plan/expectation

- **Suitable color dispenser** – a color dispenser with small nozzle and a simple locking system (figure 3).
- **Funnel Blocking Mechanism** – a servo motor with a small lever installed on the shaft of the motor. Turning the servo motor to 'on' position opens the funnel and turning it 'off' closes the funnel (figure 4).
- **Height of the nozzle** – an optimal height of the nozzle in order to ensure sharp flow of color as well lowering the chance of getting into contact of the ground. This was resolved using a customized aluminium frame to hold the funnels in position (figure 3).
- **Smooth flow of colored powder** – A vibrator motor was installed on the body of the funnel in order to provide a constant vibration to make powder fall smoothly whenever the lid is opened. Initially the vibration of the motor was so violent that it was affecting the movement of the bot. We investigated the case and finally decided to use a resistance of 100Ω to bring the vibration to an optimal level (figure 6).
- **Detection of colored markers** – We covered the entire top surface of the bot with black paper to eliminate all the color noises deriving out of the various LEDs installed on the bot as well as distinguishing the colored markers as prominently as possible (figure 7).
- **Weight balancing** – As the aluminium frame was installed on the left side of the bot, rotation of the left wheel was affected by the excess weight. As a consequence of that the bot was unable to provide a linear trail initially. We compensated this by putting some extra weight on the other side of the bot. We had to go through several iterations using different objects to match the exact weight of the frame.
- **Height and Resolution of the Camera** – There was a trade-off between the height and resolution of the camera. A very small height was unable to provide view of the entire arena whereas it was hardly possible to clearly distinguish colored markers with a greater height because of the poor resolution of the camera. Ultimately after several iterations we were able to achieve an optimal height serving both the purpose.

6.2 Things that didn't work out as per plan/expectation

- **Using LEDs as colored markers** – Detection of the colored markers is heavily dependent on the external light sources. Each time lighting condition changes the threshold pixel color values for the markers were to be adjusted. We thought of installing two colored LEDs on top of the bot so that the camera could easily view it despite the lighting conditions. In fact if this policy had worked out then the bot could even draw rangoli without any external light source i.e., in the dark. But unfortunately due to the poor resolution of the camera it was hardly possible to distinguish the color values of the markers. All that the camera was viewing was two white dots on the bots. As a result we had to ultimately give up this idea.

- **Bug in the Esterel compiler** – There were some issues with setting the variables through esterel. By default esterel compiler assigns a '1' value to the flag variables used which results in an incorrect *if then else* clause in the resulting C-code. There is no option in the esterel compiler by which we can assign a '0' value to a mutually exclusive flag variable. Ultimately we had to change our esterel code to get rid of this.

7 Future Work

7.1 Tri-Color Rangoli:

We have built the hardware required to extend the project to make the bot draw multicolor rangoli. There are three servo motors installed on the aluminium frame of the bot. The power cables of the three motors need to be connected to the three servo motor power slots (named S1, S2, S3 respectively) provided on top of the bot. The esterel code needs to be extended with a few other signals. The scilab code for converting the image into collection of run-length encoded rows has to be modified to append a few more guard columns in beginning or the ending.

7.2 Vectorized Rangoli drawing:

In our project the bot can only draw in a rasterized fashion. But it is possible to extend this project to make the bot draw vectorized components of the input image. The only constraint of doing that is to ensure that the bot doesn't tread on the portion already drawn. In order to do that one possibility is to use a strong net with sufficiently large holes in it. In that not only the bot can move freely on top of it but also the colored powder will fall freely through the holes of the net onto the ground to create the rangoli impression and there will be no chance of going over one portion.

7.3 Entirely Negative feedback Controlled System:

Currently the entire system is basically a hybrid control system i.e., part of it is positive feedback controlled and part of it is negative feedback controlled. The controlling through camera and image processing is negative feedback controlled whereas once the bot starts drawing the image on the floor it takes the control and acts as a positive feedback system. Due to the inefficiencies of Scilab we had decided to keep like that only as frame capture rate in Scilab is very low. As it improves with later versions we can control every movement of the bot through negative feedback system to make more accurate rangolis. In that case we will not have to depend on the accuracy of the bot itself.

8 Conclusion

The experience of doing this project was quite enriching as it gives a hands on experience on embedded systems. It also gives an insight about the uncertainties and limitations involved in all aspects of physical embedded systems. It is worthy to mention that we could deploy many concepts learned in the lectures directly into our project e.g., negative feedback system, event-driven programming etc. We have successfully implanted a rangoli drawing bot and identified all the potential areas of betterment. Last but probably the most important aspect of our project is that it is entirely based on open source tools and software e.g., esterel, Scilab, SIVP toolbox.

9 References

1. www.telit.com/module/infopool/download.php?id=542
2. http://en.wikipedia.org/wiki/ZigBee
3. www.cse.iitb.ac.in/~cs684/esterel/Programming_Firebird_Esterel_manual.pdf
4. http://www.scilab.org/
5. http://sourceforge.net/projects/opencvlibrary
6. http://atoms.scilab.org/toolboxes/IPD/2.0