Intelligent Machines Design Laboratory

EEL5666
Summer 2002

Autonomous Dice Recognizing and Manipulating Robot

Sensor Report
Thursday, July 11, 2002

Cyrus Harrison

Table of Contents

Introduction:

This paper serves to describe the components of my robot that allow it to sense its environment and change it.

Sensors:

This section shows the sensors used to collect environmental and die information necessary to recognize die faces.

Image Recognition with X10 Camera:

Camera Mounting:

The X10 Camera is mounted on the front of the mobile platform using a thick paper tube to isolate any environmental light. The light source for dice staging is a white led. Using a 100 Ohm resistor the light is very bright and illuminates the dice well enough for the image analysis algorithm to work. Fooling with many distances of the X10 lens, I found that the best image was produced when placed around 2 inches.
At this distance the camera focuses on the die well and external light from the bottom of the staging area is causes minimal interference.

When the IR Bream Beak is triggered (die has been found), a small servo lowers the camera tube down so its bottom touches the dice platform. From here a clear image of a die can be obtained.

To power the X10 with batteries an X10 battery pack is needed. The battery pack contains a small circuit which amplifies four AA batteries to the voltage necessary for the camera. Since the Battery back is very bulky, I removed this unit from the plastic molding and attached it to a smaller battery pack. I also shortened the X10's Power Cord because it was a nuisance.

Image Analysis:

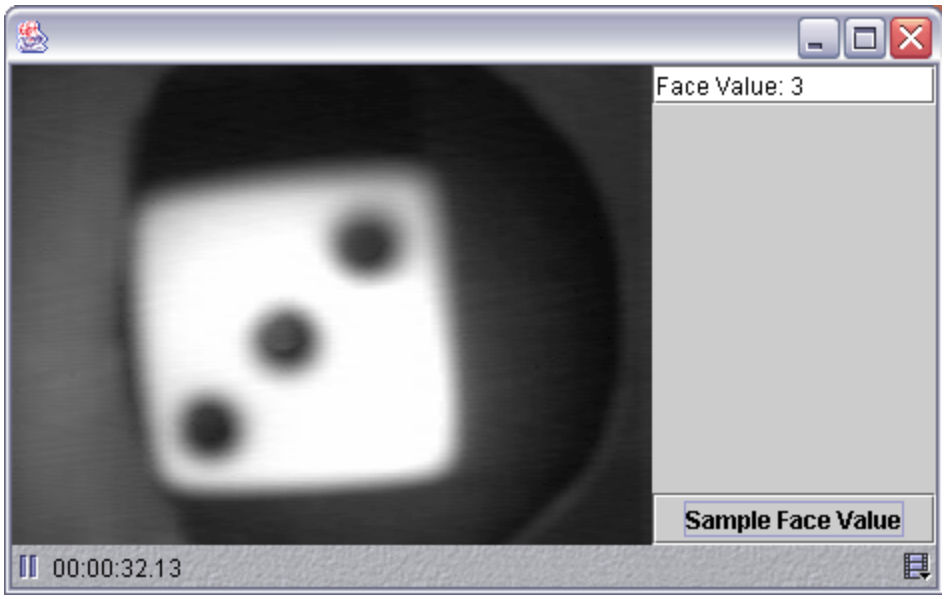
The algorithm looks for similar regions in the image and counts how many of these separate components exist. The camera area is staged so that any part of the image that is not part of a die should be dark. This means counting the components will result in one more than dots are on the die observed.

The image analysis is done in three steps using connected components labeling algorithm as a base.

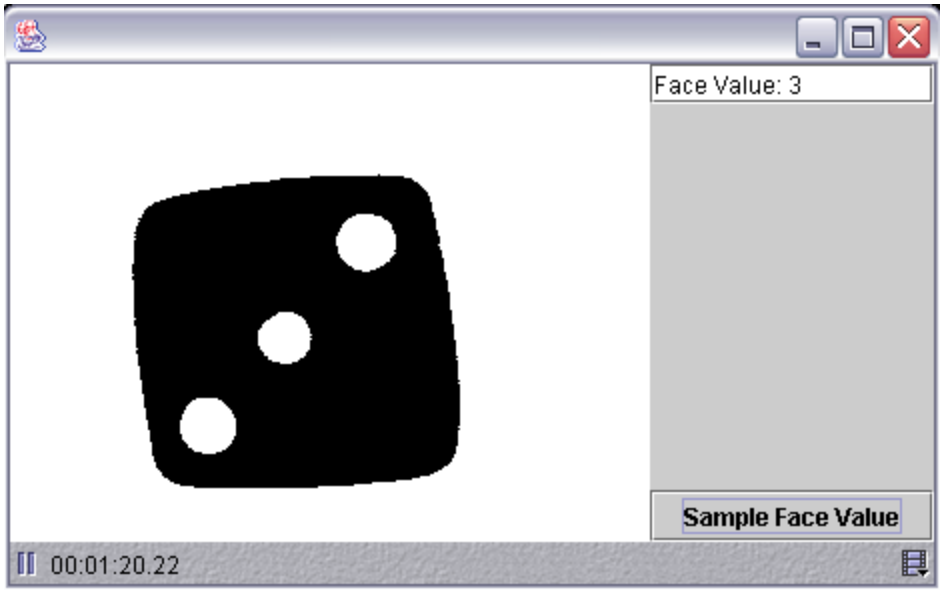
Low Light Filter:

Examines each pixel of the image and filters out anything that is not "dark". This gives a new image which has the same resolution. Each new pixel is binary to represent if the pixel was dark or light based on the filter threshold.

Before LLFilter:



After LLFilter:



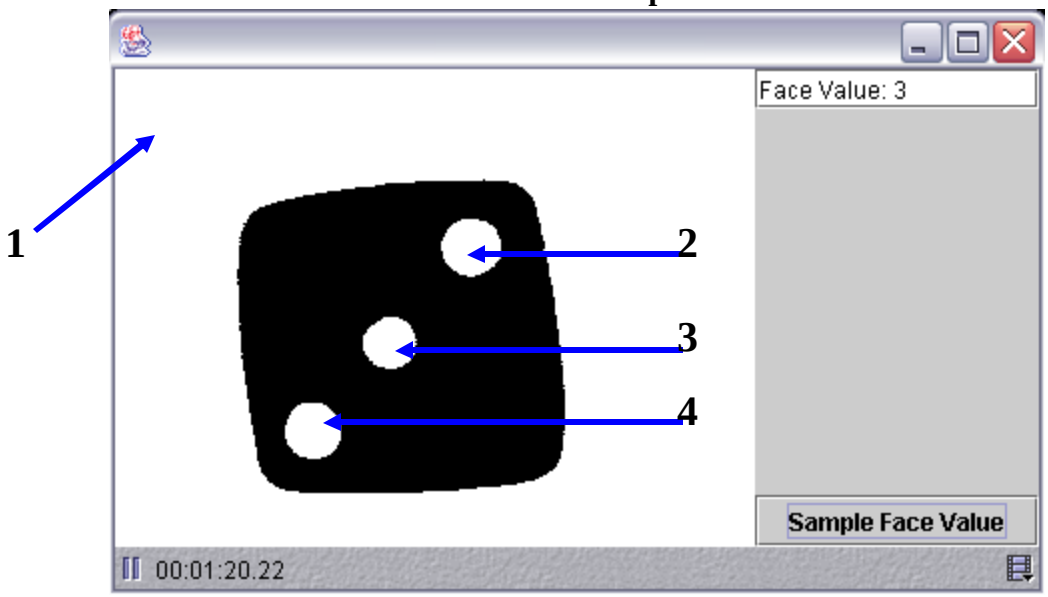
Labeling of Components:

Examines each pixel to see if its adjacent pixels are of the same nature. With a binary image if a dark pixel has neighbors that are dark, then they will be labeled as part of the same component. If a dark pixel is newly discovered it is considered a new component. This scan is done left to right, top to bottom on the image to discover how many distinct components.

Cleaning of Components:

Cleans the discovered components to avoid errors from a messy signal. This process eliminates errors in the process because of a jagged image, and throws out any components that are to small to be considered dots. Several samples are taken and the mode of these sample is used for the end result.

End result of Components

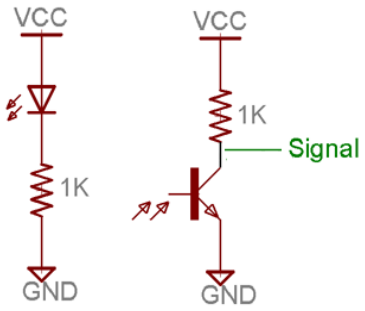


One is subtracted from the found components.

Sharp GP2Y0A02YK Distance Measuring Sensor:

The Sharp GP2Y0A02YK despite the ridiculous name is a very easy to use IR Proximity Detector. It has a simple three connection interface (Signal, GND, VCC) which can be plugged directly into the ADC header on my extension board. The Sensor Datasheet says it can accurately measure between 20cm and 150cm. This range is much more than needed. The sensor voltage raises the more its emitted IR is reflected back. Though experimentation I found that when something is close enough to cause concern, the value is about 2 Volts (0x190 when read by 10bit ADC). When the sensor is very very close (collision will be shortly unavoidable) the sensor saturates and the value begins to drop. This choice of the 0x190 gives a handle on the proximity of an approaching object and also will snag it if the sensor saturates.

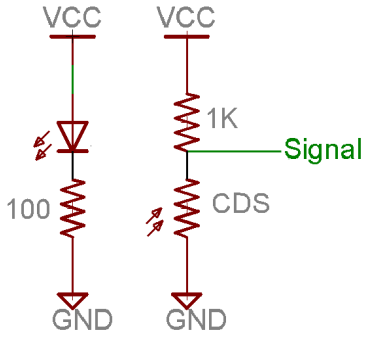
IR Break Beam:



The IR Break Beam is used to detect when a die has entered the image recognition staging area. It is a simple IR Emitter LED and IR Detector Transistor pair. I used heat shrink tubing to collimate the light from the emitter and aim it directly at the detector. When the beam is broken the voltage raises from 3 volts to almost 5 volts.

As long as there is not a very bright light directed on to the detector (easy enough to avoid) this characteristic will prevail.
The Break Beam detection takes precedence over IR collision detection, and causes the platform to stop and lower the Camera. After it receives the value from the computer via RF receiver, Ranzor will lift the Camera and resume normal operation.

Edge of Table Sensors:



To create the Edge of Table Sensors I used Cadmium Sulfide Cells and bright white LEDs. The White LED is connected with a 100 Ohm Resistor exactly like the one used to light the die staging area. I used heat shrink tubing to collimate the light into a small bright circle, and placed the CdS Cell in series with a 1K resistor next to it. The sensor is very similar to Vinh Trinh's from Spring 2002.
The concept for edge detection using this sensor is pretty simple. When the sensor is on a surface, the CdS Cell will see the reflection from the white led and change resistance accordingly. When the sensor is placed over nothing (over the edge of the table) the light will not affect the CdS Cell. By examining the voltages produced in these situations the robot can prevent itself from falling off of the table. I have tested it on different colors of surfaces and it should work for any surface light or dark.

To ensure that external light does not reach the CdS Cells, I encapsulated the pair in a plastic film canister. I cut and added to the film canister so when mounted on the front of the platform it would extend to the table, and shield the sensor from external light.

I created these sensors and they worked well, but I found that just using the Sharp GP2Y0A02YK worked just as well if not better, so I decided on just using these for the final design. If you are doing table edge sensing in the future, I would recommend you considered using the Sharp Sensors because they are very easy to use. Just mount them high enough off the table so they don't become to saturated, then you simply look for a drop in the analog value to indicate the absence of the table.

Actuation:

This section describes the types of actuation in Ranzor. These provide the means necessary to collect, recognize and roll dice.

Locomotion

Servos

Locomotion is created via two prehacked servos. I choose the prehacked servos for ease of use. Plug them in and experiment with PWM timing and you are ready to roll. They are not very high quality compared to other servos I have used, but they get the job done. I have noticed that they go a little faster forward than backward. Because they have to be mounted in reverse on opposite sides, this causes the robot to turn a bit when attempting to move forward. It has not caused any problems; actually this helps my robot get out of possible traps when both IR are switching on reporting obstacle proximity.

Wheels

I purchased these wheels because they could plug directly into a servo, and they were the correct size for my platform. They are nice wheels made specifically for hobby robotics and I am very happy with them. Using the rubber bands they slip very little on normal surfaces.

Camera Lift

Servo

For lifting my camera from up and down the staging area I needed something with a small profile. I original planned to lift the dice up to the camera with a platform, but this was unnecessary. I selected something much simpler. Because I wanted to have an aesthetically pleasing solution I chose the smallest servo I could get, and hoped the torque would be adequate. It turned out to work fine. This tiny servo is around the size of a nickel.

Die Rolling

Solenoid

To roll a die I will be using a solenoid. I originally planned on using the robot's motion to knock the die over. Uriel talked me out of this, using a solenoid seems easy enough so I ordered one. I will use a continuous solenoid so no PWM is needed.

Voice

My original plan was to use a SPO256 Speech Generator Chip. This chip was found in early video game systems, and sold at Radio Shack in the 1980's. The chip is nice because you can create any word needed combining a series of 59 phonically sounds. The only problem with this chip, is it is not sold any more. I purchased on Ebay from a private seller. I created a board from diagrams online for this chip, and tested it. I was only able to obtain some gibberish. The timing sounded correct for the words but I had no real luck.

Since I have no real way of knowing if the chip ever worked I am going to make another attempt at voice with a play back chip. The chip I choose is an ISD2500 ChipCoder from Winbond. Using the documentation it was pretty easy to wire up. I also used a LM386 Op Amp to amplify the Audio. The following diagrams show the ISD2500

The only regret I have with this chip is that I chose the 120 second Version. All of the ISD2500 Series Chips have the same amount of memory for playback, the only difference is the sample rate, so the 120 Second version sounds kind of crappy. It turned out I did not need that much play back and I wish I had better voice quality.

Conclusion

This concludes my explanation of sensors used and type of actuation for my IDML robot. Combining these items, I am able to recognize die, and I am working on integrating behaviors to have a complete robot.