Android based Object Detection and Classification:

Modeling a Child's learning of what's Hot/Cold

Matthew Weber
CPRE 585x - Project Proposal
March 10, 2011

(Image from http://www.dhyra.com/2011/01/android-ice-cream-can-be-enjoyed-on.html)

Abstract

As part of a young child's learning process, there are specific items that are discovered through the embodied experience. It's been shown that an infant is going to using their hands and feet to touch things and characterize how they “experienced” the objects. The goal of this project is to model that learning, by organizing the characteristics of learned objects, allowing a prediction of what new similar objects might be like. Success would be the software/robot determining if a object is going to be hot or code based on previous experience.

Introduction

The concept is to attempt to model a child's learning process, when related to predicting what new objects are like based on previously learned objects. A object is defined for this project as something like a ice cream cone or a pizza. It's something that consists of color and a specific set of environmental traits (hot/cold, luminous, etc). This experiment takes in those traits via imagery and environment sensor data to define what an object is like. That data is then used to form a model of relationships of similar objects that can be later compared to a new object. When a new object is discovered, the prediction process would try to fit it to an already learned object. Even if the new object isn't predicted correctly, there is still a valid path in it's process to learn that new objects characteristics. If that path is taken, the algorithm performs a confirmation step where a correction could be made to then “correctly” learn that new object's characteristics. This concept is relying on using multiple sensor data sources to decrease error in the learning/predicting algorithm, but may still run into cases where the “noise” is to high to get an understanding of what is being analyzed. Understanding this “noise” is going to be part of the experimentation process and may limit the initial project inputs to controlled background colors and images that appeal to the available image detection algorithms.

This concept strives to provide the user, without an embodied robot, a method for doing sudo-embodiment through the use of “hand” sensors and “eye” camera. I've only selected to use a subset of sensors available, but the design allows enabling of more sensors on the hardware as needed to enhance/supplement the existing sensory data. Things like the camera could also be enabled to do video instead of the initial still shots currently planned. Since the Android and communication protocols being used are based on open standards, the interoperability of this design with existing hardware could definitely be leveraged.

This project is targeted to students who have simple technology like their cell phone and micro-controller kits available to run experiments on. With the end goal of providing a framework that allows object and sensory input learning to be organized with relationships. This framework could be then used to take a Android based device and make it a “brain” for a robot. There have been some attempts to use a Android device in a “cellbot” (http://mashable.com/2010/03/06/cellbot/), but the “cellbots” seem to rely on preprogrammed functionality. Compiling the existing camera inputs with computer vision processing, would make the cellbot a powerful platform for prototyping autonomous movement.

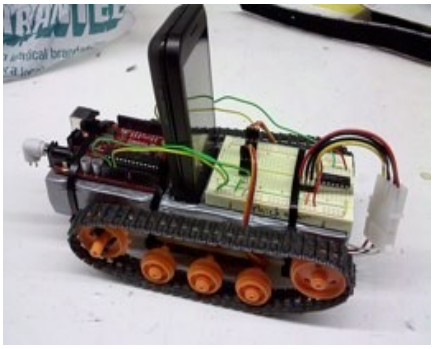


Illustration 1: Cellbot (http://mashable.com/2010/03/06/cellbot/)

When researching the resources available for doing a computer vision project and understanding sensor data, there weren't an open source examples or platforms available as a good starting point that already brought that information together to easily form predictions. When this project is completed, an Android application will exist that processes inbound sensory data and creates the necessary predictions for learning of new objects.

Approach

Since I don't have much experience in the area of robotics, I've fell back on my work experience with micro-controllers and Android. I will have to do some research to figure-out the computer vision and the neural net learning algorithms available, but initial searches have found a number of projects and examples that look promising. Items like coding up a socket transport to move the sensor data from the sensor array isn't an issue. Along with the neural net examples seem to show a easy representation of do the data inputting and organizing. The difficult part looks to be with the image processing and similarity predictions using the neural net.

When looking at this project from the perspective of a 2-year-old child, usually they can start to learn characteristics of objects experience. So attempting to do basic learning of objects based of similarity characteristics doesn't seen beyond reach. The challenge is going to meeting the capability(error rate) that the child would have for recognizing new objects based previous experience.

At a high-level, I've broken the development into the following tasks (Design, Code/Test, Integrate, Document). The design phase will consist of growing this document with more detailed sequence diagrams, algorithm information, and test cases / expected results. The code/test phase will consist of a few steps.

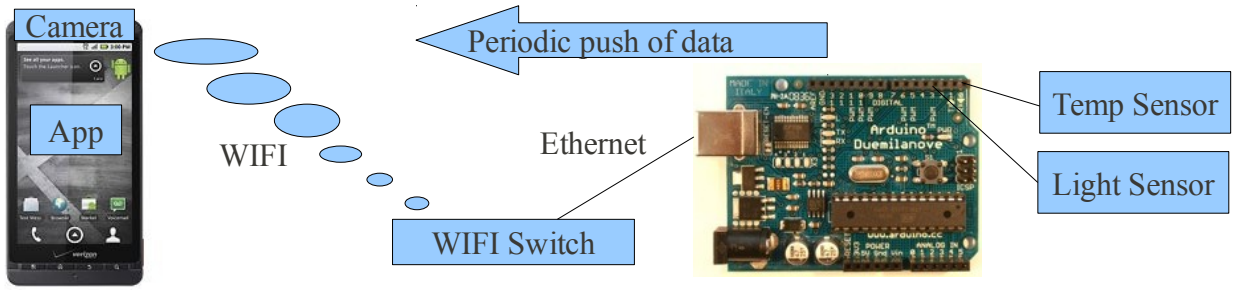
- Develop image and sensor data gathering code
- Develop computer vision image processing code
- Develop the basic Android application
- Develop communication between micro-controller and Android application

The integrate phase will consist of taking each of the code/test tasks and bringing them together on hardware. It will also contain the tasks for testing and collecting the experiment data. Following the integrate phase, the documentation will start to create the necessary artifacts (write up, demo, videos, poster, presentations, website).

Digging into the details of the development, I'll be starting with the sensor package. The micro-controller board configuration would first be setup to enable the pins/interfaces that connect to the sensors and also the Ethernet interface for off board communication. Then the data collection and format would be coded that would result in IP packets for relay to the Android application. Once this basic application platform is in place, I'll begin the process of integrating the computer vision library into Android and building a basic model of image matching either using histograms and or edge/point detection.

Equipment

Since I'm taking this class via distance education, I'll be using simulation combined with some sensing hardware. The main component is an Android phone that will collect sensor data and images. The sensor data comes from an external Arduino sensor package that would normally be attached to a robot's hand. The sensors being used are a one-wire temperature sensor and a resistive photo sensor. For my experiment, I'll be manually placing the sensors close to the object being learned. In the diagram below, the connectivity is shown between components.



Some of the initial work I've done has been focused in validating a working hardware platform.

- I've found that I wasn't going to be able to use WIFI for communication from my micro-controller because of outstanding issues with the provided drivers. Luckily there was still an Ethernet option.
- The micro-controller example software has allowed accelerated development because of provided example code that allows the hardware configuration I need to pretty much work our of the box. (wasn't expecting this)

Data Structures

Micro-controller communication packets are defined as a UDP packet of data size 10bytes to port 20000 with the following format.

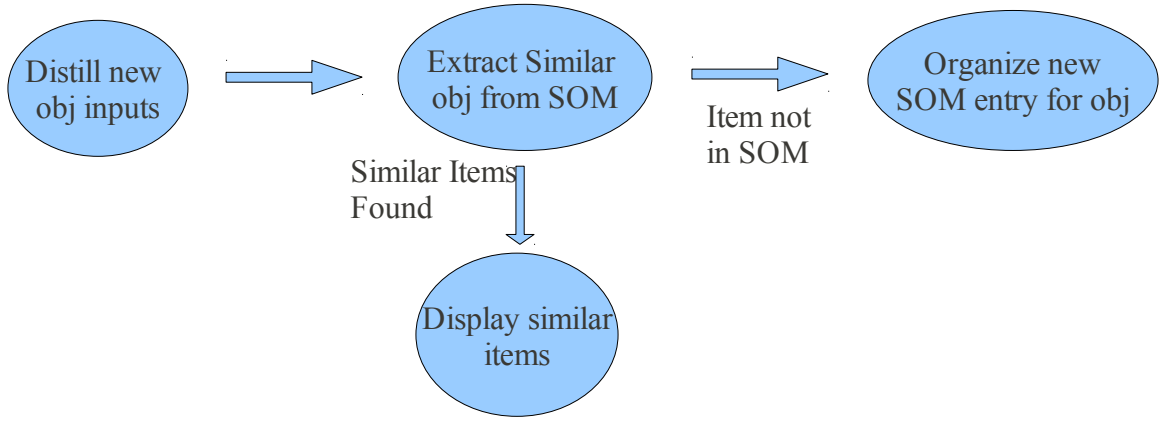
2 byte(s)	1 byte(s)	2 byte(s)	2 byte(s)	3 byte(s)
NA	Temperature	NA	Light level	NA

Object data storage is going to be based on the neural net configuration, but at a high-level the following will be captured.

- Image features (Histogram/Edge Point data) – (Array of normalized data)
- Temperature – (normalized integer)
- Light level – (normalized integer)

Algorithms

For this project there are (3) categories of algorithms required. One to distill the raw data to get information from it, the second to take that distilled information and organize it, and the third to efficiently extract the information that was organized for comparison.



Distillation

The data from the sensors will be taken and normalized between -1 and 1 for use in the neural net. For the image processing, I'm making some assumptions that the image captured will be focused in on the object excluding most of the surrounding noise that could affect the generation of a histogram for comparison purposes. A histogram is a graph of the amplitude of the number of pixels (read left(black) to right (white)). Shown below is the basic graph of count vs intensity and an example image w/ it's respective histogram (http://www.shotaddict.com/tips/article_Reading+A+Histogram+Correctly.html). That histogram data would be stored as part of each objects set of traits.

Beyond using histograms to analyze the images, I've investigated edge/point detection using an algorithm like Speeded Up Robust Features (SURF). A combination of SURF and histograms looks to be the best analysis to do image matching. (http://www.cscjournals.org/csc/manuscript/Journals/IJIP/volume3/Issue4/IJIP-51.pdf) Depending on the histogram analysis approach, it looks to compensate for the illumination and rotation issues that come with the SURF algorithm. Especially since the histogram is just a pixel distribution count not a feature detection algorithm. Some of the first testing during development will be to understand the success rate of both techniques to create image "finger prints" that provide good data for the neural net.

Organization

One of the concepts I'm still working to figure-out, is how to utilize a self-organizing net that allows me to easily change the dimensions on the fly and reorganize. I'm not sure if I'll get that resolved in this project or if the net will be a large fixed size to start with.

I've been researching neural nets and found one in particular that stuck out as a good pick for this project. It's a Self Organizing Map (SOM). I found a couple really good examples that utilize the SOM for doing picture predictions based on previously learned images. Also since the SOM can be told to refine it's relational organization, the learning process doesn't require a fixed library of objects that have already been learned. It's possible to start out with new objects and build up to having a large set of object data that is relationally organized for fast prediction searches. Using the example discussion about the picture prediction as my basis, I'd like to factor in the sensor data to produce predictions on if objects are hot/cold bright/dark in-addition to if a picture of a is similar. Basically allowing multiple sensor sources to refine the prediction.
The way a SOM works is to take multidimensional nodes that each have a relationship to an input node that provides normalized values (-1 to 1) that define what's unique about that node. For example most of the SOM examples are using colors. i.e. like below, there would be 3 input nodes (RGB) and each map node would have a normalized value to represent each of the three colors to create the specific shade that defines that map node.

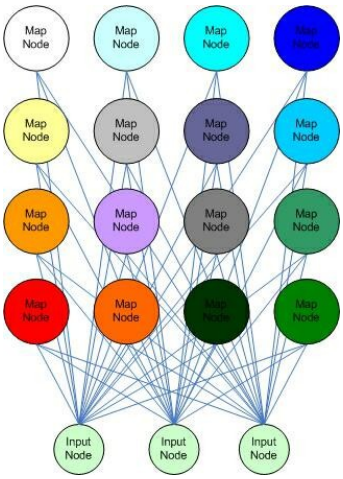


Illustration 2: SOM Input and Map Node Connectivity
(http://www.generation5.org/content/2004/aiSomPic.asp)

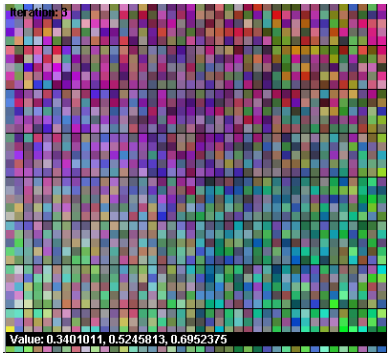


Illustration 4: SOM Color Example - Unsorted
(http://www.ai-junkie.com/ann/som/som1.html)

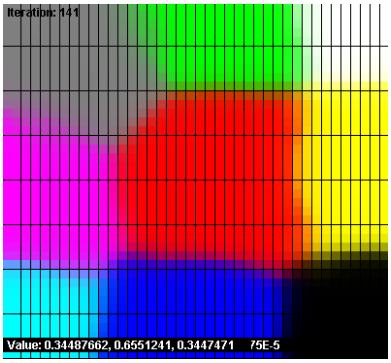


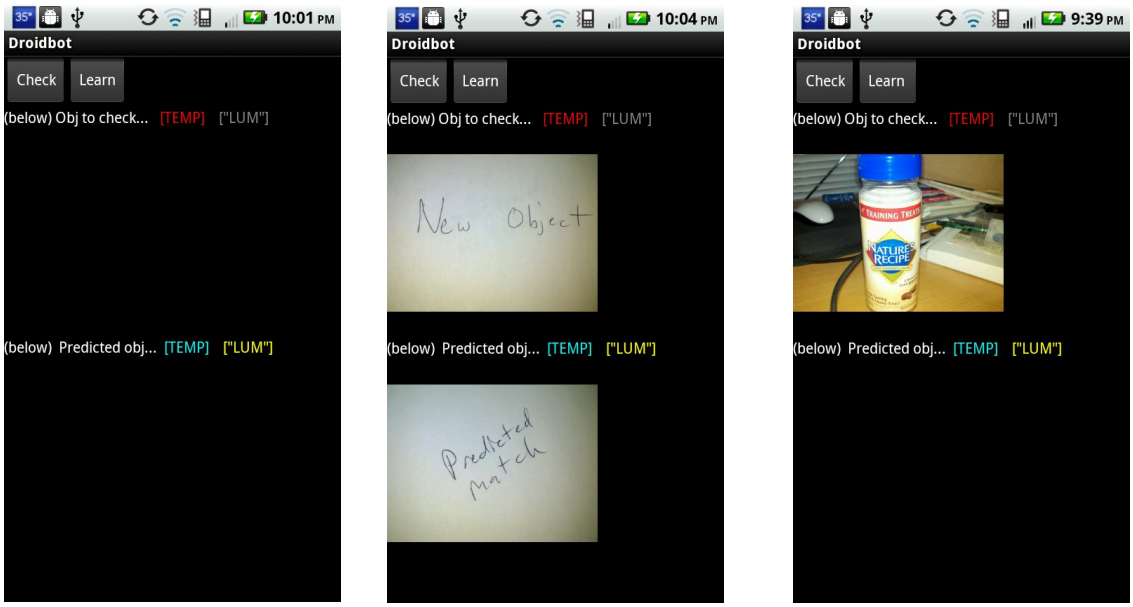
Illustration 3: SOM Color Example - Sorted *(http://www.ai-junkie.com/ann/som/som1.html)*

Extraction

The extraction process would take place every time there is a input of a object for analysis. The object would be characterized and then an attempt would be made at locating the item in the SOM. Optimizations can definitely be made to the bookkeeping of the SOM to allow fast locates of similar objects, but even without that in place, it'd be possible to just use a simple tree following algorithm to pick a node at random to start with and follow to find the best position where that object would fit. The data that would be followed would be the normalized values of the image and sensor data. The end goal of extraction is to characterize the object to determine if it is hot/cold without just reading the temperature sensor. I assume that when a normal person is learning and they come upon a object that appears through previous experience as something that could be hot/cold, they wouldn't just stick out their finger and touch it first without thinking about if it would burn them. So I'm doing the same thing. First predicting if something might be hot/cold and then verifying by touching it.

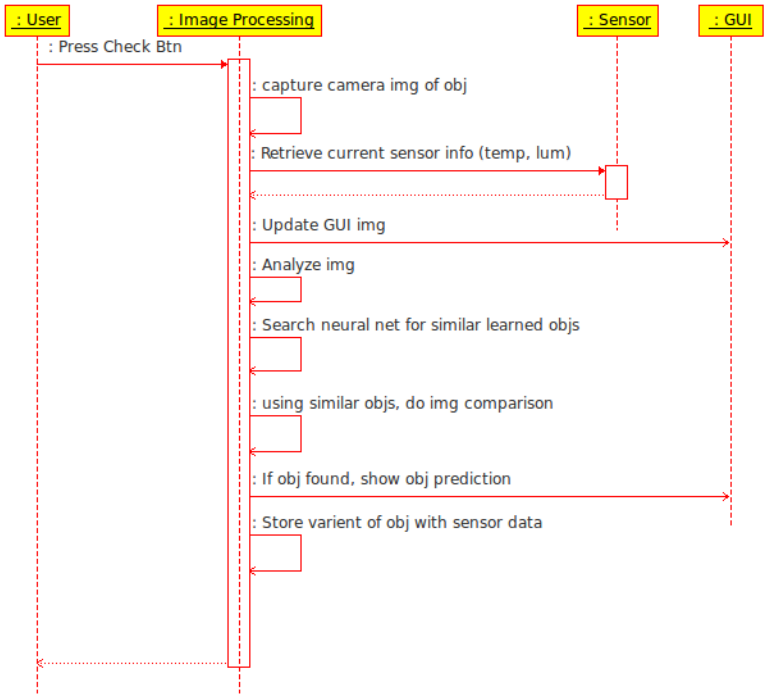
User interface

Here are some initial mock-ups of the user interface. The first image shows the interface broken into capturing data, predicting, and learning. The “Check” button will be used to start the process of gathering an image for analysis. After the image has be analyzed and a prediction is attempted (image two), the bottom of the screen will provide a suggested match along with the temperature and luminescence sensor info for that match. The temperature will change color between blue and red to represent cold and hot respectively. The luminescence/light sensor will change color between gray and yellow representing dark and light respectively. The user can then decide to agree with that match or learn the obj as a new obj. If the decision is made to learn, then the objects attributes along with sensor information are stored. i.e. in these initial mock-ups if this was the case for the last image analyzed, the user could press the learn button to add it to the neural net. Also just to note the behavior of the sensors. The temperature and light (luminescence) sensors provide a semi-realtime feed to the application and update above the top image until a object is checked and then lock in until the item is found or learned.

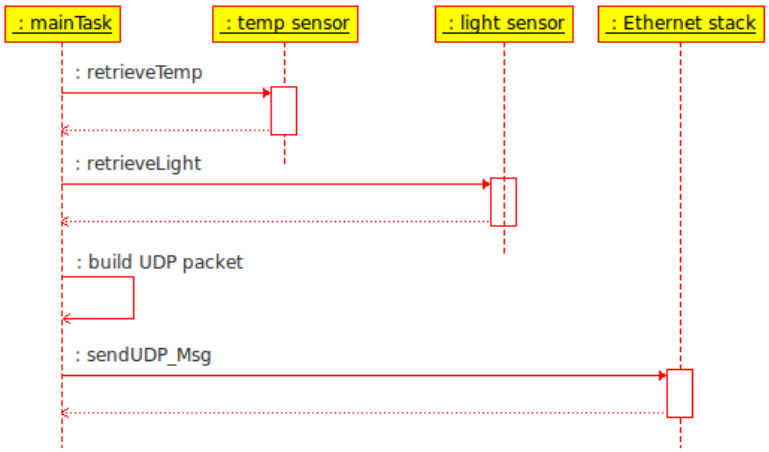


Dataflows

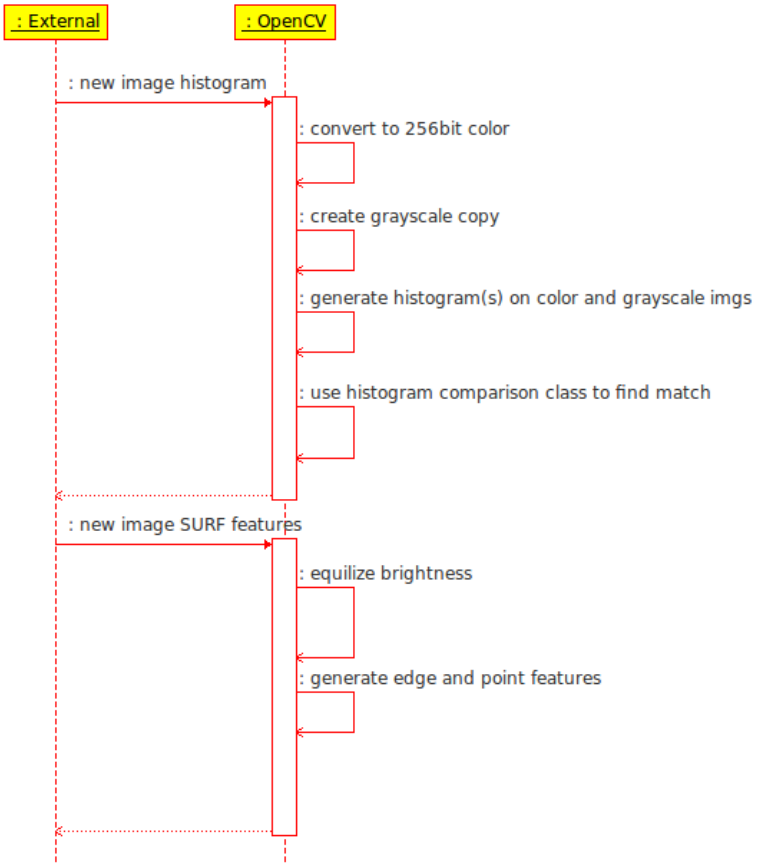
UseCase	User button press on GUI
Description	User finds new object to analyze, to predict if we can find a match.
Preconditions	Sensor data is valid and camera is focused on new object.
Postconditions	New object is either stored in the SOM or a object was displayed that matched what was analyzed.



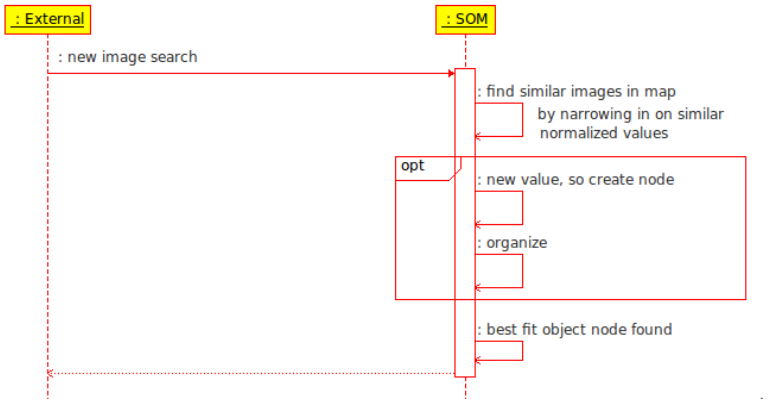
UseCase	Sensor data provider
Description	Micro-controller pushes data out via Ethernet for application use
Preconditions	Ethernet connectivity Sensors initialized
Postconditions	Repeat in endless loop.



UseCase	Image processing
Description	Convert images into a hash or simplified value that can be compared.
Preconditions	Image is png formatted
Postconditions	Image has been converted into a mathematical representation.



UseCase	SOM processing
Description	Organizes data by normalized values unique to each node
Preconditions	Data is normalized for insertion to map New node was added to map and needs to be placed
Postconditions	Map is organized with correct relationships formed.



Software

Libraries

The baseline software for doing the image analysis is the OpenCV port to the Android platform. It's libraries seem to support all the operations I'll need to distilling down a image into comparable characteristics. It currently directly interfaces with the Android application allowing me to use a Android application to directly control the camera and feed the image into OpenCv directly for analysis. Following the analysis the data would be handed off to extraction step to see if the SOM contains similar data or if this is new data that needs to be added. I have not used this library before, but the examples look pretty easy for the type of analysis I need to perform. Specifically they have a image detection example that looks to offer one way of looking at that problem.

I've looked at a few other approaches to the image analysis using histograms, hashes, and hamming distance for comparisons to find close matches. The library pHash might be another good fit if the initial testing with OpenCV doesn't look promising. (http://stackoverflow.com/questions/2146542/opencv-surf-how-to-generate-a-image-hash-fingerprint-signature-out-of-the-d) (http://www.phash.org/)

The micro-controller software is provided with the kit. A few examples for the sensor implementation are provided by the manufacture or Ardunio website. I haven't worked with this before, but my initial investigation has allowed me to already complete the capabilities I need this to perform. It has built in sensor and UDP Ethernet libraries that can be directly reused to relay all the sensor information via WIFI to the Android phone application.

The Android software for the user application is created using the freely available SDK. I have done a little coding with this before and hope to leverage all the examples to do the functionality I'd like to accomplish.

Experiment Setup

The setup consists of a set of images to use in the test cases and also the hardware configuration show above in the Equipment section. The set of images will contain a variety of black and white and also color images. Each set independently tested to see the impacts of color on determining if a object is going to be cold/hot. Here are some of the sample images. (http://www.ngsp.com/Portals/0/Downloads/41018_tg.pdf) These samples would be used to initially train the SOM. Once the algorithms are worked out I would then move on to using pictures I take and tyeing in the sensor data as an extra trait to be analyzed. Success would be to have these samples grouped by common specific shapes and colors.

Timeline (Due on)

March 4 – Narrowed down hardware and software library selection

March 7 – Completed micro-controller integration

March 10 – Turn in project proposal

March 18 – Complete Image analysis with OpenCV/pHash and have information to feed into SOM

March 25 – Complete initial SOM testing with example containing the same inputs required for this project

April 1 – Complete first attempt at using SOM to store and organize OpenCV analyzed image data

April 8 – Make SOM searchable to do comparisons to new object information

April 15 - Testing

April 21 – Writeups Due (poster, writeup, webpage)

April 25 – Dead week presentations

Evaluation Methodology

The plan for testing is to develop a set of know data and expected results (inputs hopefully == outputs).
The first step will be to build a automated OpenCV application for Android that captures camera pictures and converts them using a SURF or histogram algorithm to raw data that I could dump to a file that could be used for post analysis. This data would then be flowed into the SOM processing to test the learning quality of the node organization. Success is measured with the ability to take a converted picture and compare it to another picture of the same object with varying light, angle and focus and have the algorithm pick out that they match. A good example of test data is found in this paper (http://sites.google.com/site/chrisevansdev/files/opensurf.pdf). It shows a sequence of images that would be ran through the SURF algorithm to verify it's functioning correctly.

To visualize the SOM and organization, the first step is to build a SOM test application and test it to understand if the Java version of the SOM algorithm I'm using is going to work. Using this test application I should be able to take the image data collected from the OpenCV test app and build a SOM model that I can verify for organizational correctness. Then using that organized model I can attempt to search the SOM for a similar object to those currently occupying the nodes. The level of success will be measured by it placing the new object somewhere in the region of similar objects. Once this SOM model and searching algorithm are solid then I can convert the application over to a Android application for integration with the camera as my input device for gathering/interrogating new objects.

For the final testing of the Android application, the test conditions would be as follows.

- Identification of a set of objects that would have their pictures taken and temperature/light emittance measured.
- Instructions for the sequence of application event interactions to test the state-machine for valid transitions
- Post analysis of the SOM organization at specific points during the learning process to make sure things are getting organized correctly
- A set of objects to classify and try to identify features based on relationships to the set of learned objects.

The evaluation of the results will look at the sets of inputs and output data to verify that a reasonable match has been made that fits the expected results (Similar in color and sensor hot/cold properties). If the result isn't what is expected, the post analysis of the SOM data and whatever the results did end up as, should allow refinement of the decision filtering process.

Conclusion

On completion, this project will model a child's learning process to associate images and sensory feedback to previous experience. As part of the effort an open source framework would be developed that would allow further experiments using self-organizing maps (SOMs) within Anrdoid. (Possibly with computer vision applications for other self learning through sensory input.) i.e. Using this project as a basis, many other applications come to mind. From simple things like selecting the perfect bottle of wine by pointing your phone at the store shelf and letting it pick-out the correct bottle based on previously learned characteristics. Also possibly based on other information you pass to the application, you could narrow down it's pick based on food pairing and atmosphere/event. Hooks could even be put in to allow other peoples feedback to direct the application's selection decisions.

I believe I've selected something that's within my capabilities to accomplish within the necessary deadline. There are some definite challenges with some of the technologies required, but the amount of resources currently available through software libraries or examples on the Internet seem to lower the risk. One of the first things I'll need to tackle is narrowing in on the exact behavior of items like image processing to make sure that algorithms will be solid. By discovering issues with those high risk items first, I believe this should be successful. Thanks for taking the time to read and review my proposal!