Camera & Sensor Submodule

What is the Camera & Sensor submodule?

The submodule consists of two main parts.

1. Pixy (CMUcam5) vision sensor

Pixy is a fast vision sensor, built by Carnegie Mellon University, that can quickly be "taught" to find objects, and it connects directly to Arduino and other controllers. Pixy is the fifth generation of CMU vision sensor that can perform significantly better than previous generations in lighting and exposure varying conditions. Complex image processing is computed by an on-chip microprocessor to save resources for the main controller of the entire rover system.

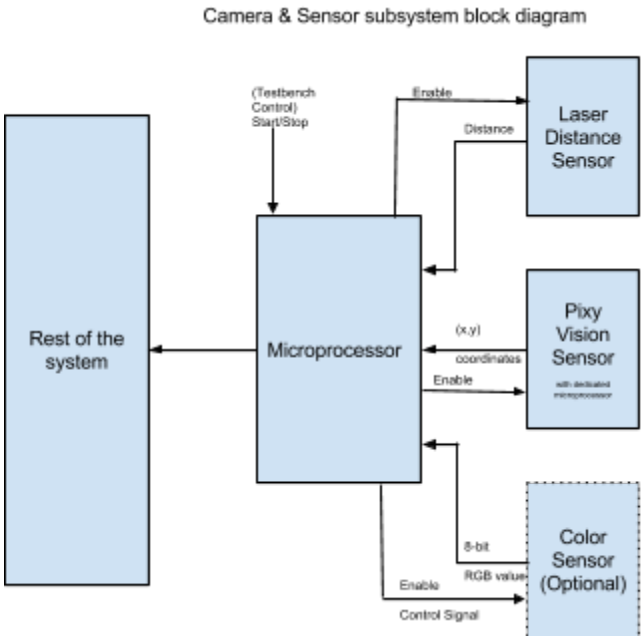
2. Parallax 15-122cm Laser Range Finder

Parallax Laser Range Finder is a distance-measuring device that uses laser to measure the distance to an object. The design uses a Propeller processor (model TQFP-44, eight 32-bit cores), CMOS camera (native resolution 640X480), and laser diode(635nm, < 3mW, 6.2mm D) to create a low-cost laser range finder. Distance to a targeted object is calculated by optical triangulation using the concept of "time of flight" as well as simple trigonometry among the centroid of laser light, camera, and object.

3. (Optional) TCS230 Color Recognition Sensor Module Detector

For testing purposes only, not used in actual design

A block diagram of all the components as well as interfaces of the Camera & Sensor subsystem is shown below.



Where is the Camera & Sensor mounted?

Both the Pixy camera and Laser Rangefinder will be mounted at the end of arm. Therefore, the Pixy camera will always orient the arm to the location where the object is and then Laser Range Finder will measure the distance how far the arm should extend.

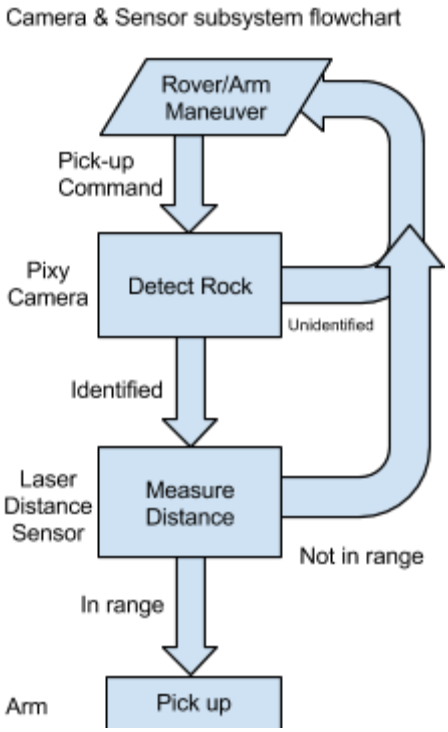
How the does the design work?

Object Detection Algorithm:
There are 6 basic steps for object detection.
Step 1: Capture Image
Step 2: Edge Detection
Step 3: Dilate the Image
Step 4: Fill Interior Gaps
Step 5: Remove Connected Objects on Border
Step 6: Smoothen the Object
A MATLAB example code and a detailed image illustration is included in the appendix.

Color value is preserved for object detection. With the algorithm above, the vision sensor outputs the exact xy coordinates (in respect of the center of the screen) of the object detected.

Once the object is detected, the screen will show exactly where the object is. The user is asked to control the rover so that it is pointing straight at the object. After that, the Laser Range Finder will measure the distance between the object and the rover and send the information to the arm how far to reach. If it is out of reach, it will notify the user to move closer.

A flowchart of how the Camera & Sensor subsystem works in shown below.



Why Camera & Sensor ideal for the project?

Image sensors are cheap and flexible choices for object detection. With the right algorithm, an image sensor can sense or detect practically anything. However there are two drawbacks with image sensors: 1) they output lots of data, dozens of megabytes per second, and 2) processing this amount of data can overwhelm many processors. And even if the processor can keep up with the data, much of its processing power won't be available for other tasks.

Therefore a dedicated processor will be assigned to complete the computationally intensive task of image processing. A product called Pixy addresses these problems perfectly. Pixy processes images from the image sensor and only sends the useful information (e.g. blue ball detected at x=23, y=54) to the microcontroller, at the frame rate of 50 Hz. The on-chip microprocessor performs all the image processing tasks. The information is able to be transmitted through several interfaces: UART serial, SPI, I2C, digital out, or analog out. So microprocessors like Arduino can communicate easily with Pixy without sacrificing lots of computing power.

Pixy outperforms many other vision sensors by utilizing a hue-based color filtering algorithm for object detection. Pixy calculates the hue and saturation of each RGB pixel from the image sensor and uses these as the primary filtering parameters. The hue of an

object remains largely unchanged with changes in lighting and exposure. Changes in lighting and exposure can have a frustrating effect on color filtering algorithms. Pixy's filtering algorithm is robust when it comes to lighting and exposure changes and significantly better than previous versions or other vision sensors.

A distance measuring device is needed for locating the object. LRF (Laser Range Finder) is preferred in this scenario over ultrasonic sensor and IR sensor for accuracy and versatility. An ultrasonic sensor is good at detecting obstacle. It has a relative short range and wide angle for detection. An IR sensor works best in dark environments; when exposed under direct sunlight, it becomes unreliable. Therefore, an LRF is the one stands out. There are two major pros about the particular LRF (Parallax 15-122 cm Laser Range Finder). First, accuracy: when within the optimal range of 15-122 cm (maximum detection range at 8 feet (2.4 meters)), the measurement error is under with 5%, average 3%. Second, additional CMOS camera(automatic white balance (AWB), automatic exposure control (AEC), and automatic gain control (AGC)): it is a compact module with both integrated CMOS camera and laser system. The device is then capable for all necessary object detection and measurements.

How does it talk to the rest of the system?

The information from Pixy vision sensor will be transmitted through several interfaces: UART serial, SPI, I2C, digital out, or analog out. Since only the color and coordinate values are the only data going to be sent out, the main controller has a lot of computing power for other jobs.

Laser Range Finder talks to the controller via serial interface. LRF outputs different data based on different commands. For example, if an ASCII letter "R" is sent to the LRF, it will output a 4-digit decimal value in mm. There are 7 basic commands and 5 advanced commands.

To be improved

Color recognition will be implemented in the object detection algorithm.
Multi-labeling will be added to the object detection algorithm.
At this moment, the Pixy camera has a viewing angle of 75 degrees. In order to achieve better performance and visibility, lens will be changed for a larger viewing angle.

Known Costs

Camera & Sensor:

Parallax 15-122cm Laser Rangefinder
$99.00

Pixy (CMUcam5) vision sensor
$69.00
(Optional)TCS230 Color Recognition Sensor module for MCU/AVR
$6.68

Testbenches

1. In an indoor environment with fluorescent lighting condition, with three rocks of size from ¾ inch^3 to the size of a human fist, colored red, green and blue, placed at the distance of 10 meters, 5 meters and 1 meter, angle with the front center of the rover at 30, 90, 180.
2. The Camera & Sensor system should be able to
    a. live feed the image
    b. output the coordinate (x,y, origin at the center of the image) of the rock on the image
    c. output the color of the ball
    d. output the distance of the ball

Appendix

Example Code for Object Detection (MATLAB)

```
function Object_Detection_MATLAB(image)

%Read Image
I = imread(image);
%disp(I);
%disp(size(I));
figure,subplot(2,3,1), subimage(I), title('Original
Image'),axis off;
% text(size(I,2),size(I,1)+15, ...
%     'EPICS Rock Rover Team', ...
%     'FontSize',7,'HorizontalAlignment','right');
% text(size(I,2),size(I,1)+25, ....
%     'Purdue University', ...
%     'FontSize',7,'HorizontalAlignment','right');

%Detect Image

I = I(:, :, 1);
[junk threshold] = edge(I, 'sobel');
fudgeFactor = .5;
```

```
BWs = edge(I,'sobel', threshold * fudgeFactor);
subplot(2,3,2), subimage(BWs), title('Binary Gradient
Mask'),axis off;

%Dilate Image
se90 = strel('line', 3, 90);
se0 = strel('line', 3, 0);

BWsdil = imdilate(BWs, [se90 se0]);
subplot(2,3,3),subimage(BWsdil), title('Dilated Gradient
Mask'),axis off;

%Fill Interior Gap
BWdfill = imfill(BWsdil, 'holes');
subplot(2,3,4),subimage(BWdfill);
title('Binary Image With Filled Holes'),axis off;

%Remove connected objects on border
BWnobord = imclearborder(BWdfill, 4);
subplot(2,3,5),subimage(BWnobord), title('Cleared Border
Image'),axis off;

%Smoothen the object
seD = strel('diamond',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);
subplot(2,3,6),subimage(BWfinal), title('Segmented
Image'),axis off;
```

Graphic Demonstration