**************************Software Readme*************************

Here we will outline the included software modules and what is necessary for each to run.

Operating System: Ubuntu 11.10 (Oneiric Ocelot) 64-bit Linux kernal 3.0.0

Prerequisite Libraries: OpenCV 2.3.1, Redis, Apache2, libVISCA, Blackmagic Intensity Drivers

Resources For Previous Software Set:
Installing OCV and sample code
http://thebitbangtheory.wordpress.com/2011/10/23/how-to-install-opencv-2-3-1-in-ubuntu-11-10-oneiric-ocelot-with-python-support/
Using OCV: http://opencv.willowgarage.com/wiki/
Installing Redis:http://redis.io/download
Installing Apache: sudo apt-get install apache2
Installing Visca: http://sourceforge.net/projects/libvisca/
Use this site as a guide: http://damien.douxchamps.net/libvisca/

Software Modules:
Header files are self explanatory, each source file has a small description of function below it.
{{{{{Format::
folder:
sources:}}}}}}
src:

capture_blackmagic.cpp / capture_blackmagic.h

This module sets up a local Blackmagic Intensity Pro capture card. It contains four functions: init, start, block, unblock and a structure definition. Calling init will initialize and open the capture card, but not start capturing. Start will start the capture process. Block and unblock are wrappers around a pthreads mutex conditional block calls. Capturing is done via a user settable callback function pointer. The raw bytes are provided and are available until the next callback call.

image_conversion.c / image_conversion.h

This module performs color psace conversion between the YUV color space and RGB. It implements UYVY422 to BGR24 and UYVY422 to BGR24 conversion

Makefile

Makefile for generating executables to run processes that communicate with the Blackmagic capture card, must type “make” while in src directory to build necessary executables.

include:

This folder contains external library header files that we link against. Their corresponding 64-bit ELF libraries are located in the lib/ directory.

legacy:

Included in this folder is our legacy code from the first semester deliverables. All this code in this folder is not necessary for the system to run.

lib:

This folder contains precompiled libraries used to link against, specifically a pixel format conversion library and libVISCA.

tests:

This colder contains rudimentary tests for various software modules. It can be compiled using make, but is not necessary for full operation

visca_handler:

This folder contains a module hooks into the libVISCA library and talks to the camera over serial. It implements a number of commands and talks to other programs using the local Redis server.

vision_code:

fsm_mover.c / fsm_mover.h

Generic Finite State Machine module. This module can build up a generic FSM during run time and dynamically link together states. It is used to allow us to build up a FSM of the room setup and transition between camera states.

linked_list.c / linked_list.h

Generic linked list implemented in C

main.cpp

Contains main vision processing routines; has classes that handle video and frame processing, and functions that generate keypoints and calculate optical flow. The main function that does this is cv::calcOpticalFlowPyrLK(), which takes in parameters outlined here:

http://opencv.willowgarage.com/documentation/cpp/motion_analysis_and_object_tracking.html
This file also contains methods that make it easy to use video files as input, such as .mp4, .mjpg, .avi and so on. Refer to the OpenCV documentation for more information.

Makefile

Generates executables to run with webcam or through blackmagic/sony ptz camera, must type "make" in shell while in this directory to generate "vision_code_blackmagic" so system can be run.

movement_logic.c / movement_logic.h

This extends the FSM module before to implement our movement logic in a controller fashion. It receives the X/Y value from the vision analysis code nad determins based on configuration values (left/right threshold, number of successive frames in picture among others) if it should transition to the next left or right state. The movement controller will populate a list in the Redis server that the website reads to populate a dropdown to manually select state. Although it is not implemented this is almost the ability to define multiple rooms and select room configuration at run time.

t_fsm_mover.c

This module implements a simple test of the FSM module.

website:

Apache2 with PHP is installed. The PHP Predis library should also be installed

css/

Misc external CSS files used by libraries

js/

Misc external JavaScript files such as jQuery

index.php

Main page that implements our web GUI

camera_command.php

Script that sets variables in the Redis server for the VISCA handler to read. E.g. move camera

poll_info.php

Polls misc data out of the Redis server to populate various fields in the index.php page. E.g. Room name, state information, Camera pan/tilt/zoom, etc...

- **status_image.c**
  - C CGI program that will read out the status image from the Redis server and output it as a image/jpeg that is polled from index.php
- **Makefile**
  - This will make the status_image.c program and copy all needed files to /var/www for use by the website
- **apache_conf.conf**
  - This contains the Apache2 configuration for use with the CGI bin and other web services. It assumes /var/www is the document root

controlCode/

Experimental control code, not used in final product.

redis_vars.txt

List of redis variables that are in use by the system.