**Camera Software Outline** (v1)

**Liam Hebert 2018-12-20**

**Overview:**

* This software will function as the interface between the camera modules and the operating system. The software will utilize the OpenCV library to capture photos from two different cameras when run then save them in a specified area. We will also take into account various problems that may arise in a spatial environment by considering multiple countermeasures.

**Objectives:**

* Capture two photos from two different cameras taking into account potential problems in a spatial environment.
* Calibration of cameras in a spatial environment
* Ease of modification by operating system to account for potential changes in physical environment, such as disk space and camera damage.

**Language:**

* C++

**Target Operating System:**

* Ubuntu Server 18.01

**Dependencies:**

* OpenCV 4.0
  + <https://opencv.org/>
  + Library of various Computer Vision interfaces and algorithms.
  + Specifically, *imgproc*, *imgcodecs*, *calib3d* and *VideoCapture* modules will be used
* *Dependencies may increase as development progresses*

**Pseudocode**

On startup: {

//gathering deviceinfo and various tests to ensure proper function

Try {

Read config.txt for camera port info, device info and file destination

}

Catch (read exception in config.txt) {

Log error;

Default camera settings ;

}

Try {

Init camera module 1 with settings;

}

Catch (init error) {

Log error;

//Ignore camera 1;

}

Try {

Init camera module 2 with settings;

}

Catch (init error) {

Log error;

//Ignore camera 2;

}

If (both camera 1 and 2 are ignored)

Throw severe error, quit program;

calibrate(camera 1, camera 2)\*

If (camera1 != null)

photo(camera1ID, “destination”);

If (camera2 != null)

photo(camera2ID, “destination”);

Return 0;

}

calbrate(camera camera1, camera camera2) {

//*note: may not be necessary, more research required*

*Run calibration test;*

}

bool Photo (int cameraID, String destination) {

//init camera for video capture

VideoCapture cap = VideoCapture::VideoCapture (*%cameraID,* CAP\_ANY);

Mat frame;

Cap >> frame;

bool result = false;

Try {

Result = imwrite(“*path/filename”,* frame); //note: no compression

//might use file serialization class in OpenCV

}

Catch (cv::Exception) {

Log error;

}

Return result;

}

**Potential Challenges**

* **Uncalibrated cameras**
  + The software will have to ensure that the camera’s stay calibrated over the long duration of the mission. A loss in camera quality could greatly undermine the project’s success. More research required on the implications of this approach
* **Corrupted/invalid Config.txt**
  + It may be necessary to research alternative methods to gather device information in the event that the Config.txt file is missing, corrupted or incorrect.
* **Failed initialization**
  + It is possible that the camera’s fail to open during the start up process. Further research will have to be done to minimize the occurrence of this happening.
* **USB interface with cameras**
  + More research will have to be done to appropriately detect the correct cameraID via USB when multiple devices will be connected to the OBC via the same interface
* **NIR Camera interface**
  + Further research will have to be done in regards to the NIR camera to ensure the software takes advantage of the full benefits of the camera. This includes how to calibrate a NIR camera
* **File Storage**
  + While highly unlikely, it might be possible that the software runs out of hard drive storage. It may be worth considering countermeasures to this event
* **File Naming**
  + Ensure unique file names for every photo to avoid overwriting files. Potential workaround could be using a file serialization system
* **Speed of photos**
  + It may be worth utilizing multi-threading (ie: running both cameras at the same time) to ensure both photos capture the exact same environment. More research/testing required to gage importance and capability within the OpenCV library.