Project Periscope

Updated Requirements and Verifications

Team:
Aabhas Sharma (asharm36)
Michael Congdon (mcongdo2)
Siddharth Murali (smurali2)

TA:
Ankit Jain

ECE 445 – Senior Design

Fall 2015

University of Illinois at Urbana-Champaign

1. Requirements and Verifications

- Modular Functionality

Requirement	Verification	Points
Oculus Rift Unit		
● The Oculus position data must correspond 1:1 with User's movements with a tolerance of ~10 degrees	● Measure SDK Pitch, Yaw and Roll values with the received Matlab Pitch, Yaw and Roll values.	10
● Achieve a maximum latency of 3s ± 500ms	● When User movement degree and 3D Vector degree are exactly equal, timestamps should not differ by more than 18ms with tolerance of ± 2ms	10
Camera unit		
● The camera to PC image transition latency must be within 1.2s ± 5ms	● Connect the video feed over IP to the VLC Media and time video lag. ● For the timing, use a stopwatch to monitor object appearance	10

● The image processing algorithm must return output within an additional 0.5s ± 5ms	● Run the image processing on the input video feed. Send the output video feed to a server. ● View output video feed over IP using the VLC Media Player and time video lag. ● For the timing, use a stopwatch to monitor object appearance	10
● To meet our latency standards of 18ms(± 2ms), the image processing must return output image within 15ms ± 5ms	● The timestamp at the end of the program must be within 15ms ± 5ms of the program start time	15
Swivel Arm Unit		
● Simulink can read encoder counts and output PWM signals to control the motors in the arm	1. Use “raspi” function in Matlab to ensure connectivity to raspberry pi 2. Scope encoder signals, make sure no encoder counts are being missed a. Set up a Matlab function block to increment/decrement a counter for	10

	each encoder pulse. Should track position accurately 3. Output PWM pulse through Simulink a. Motors should move accordingly	
● Motors are able to produce 1.5 Nm of torque to within 10%	1. Attach simulated load calculated to require 1.5 Nm of torque to lift/rotate 2. Power motors with battery, make sure they are able to lift the load	10
● Motors should be able to actuate camera unit with less than 7% overshoot. (At 8+%, the differences in FOV's may be overcome and the video will oscillate)	1. Power on system, with either the real camera unit or simulated loads attached to each motor 2. Using step input (to simulate worst case scenario of infinitely fast head movement), have motors turn unit 180 degrees 3. Measure encoder outputs and analyze system response. Should not exceed 7% overshoot.	20
Total		100

- Tolerance Analysis

  A major challenge for us to be able to map a 2D camera image feed to the concave Oculus display. The Oculus offers the user a very immersive experience by presenting a virtual image that appears larger than it actually is. Although this is a key positive aspect that enables the complete VR experience, it could result in major health hazards if not done right. If the Oculus' display is screening images that are distorted or have unexpected skips in motion, the user could become vulnerable to seizures, epilepsy attacks, motion sickness etc.

  Given this situation, it is important for us to ensure we are significantly accurate in our image processing. We will be running our camera feed through barrel distortion algorithms[2] that will map our two dimensional feed to a concave screen using the following formula: $0.24r^4 + 0.22r^2 + 1$. Basically how this works is that for any pixel on the Oculus screen that is distance r away from the centre of the concave screen, the appropriate pixel to be copied over from the 2D camera feed is given by $0.24r^4 + 0.22r^2 + 1$. [2]

  Our tolerance to this has to be significantly small. As highlighted earlier, we can't have a poorly distorted image. Therefore, we are only accepting a tight tolerance of ± 2px offset in radius distance while mapping the 2D image to a barrel image.

  The way we will ensure this tolerance is using a fundamental method of verification that is used in most image comparison software - we will use sample Oculus images as the

input for our image processor. We will then compare our output image to the Oculus version of the sample image. While comparing, we will not just check each 2D pixel with its corresponding barrel pixel location, we will check all barrel pixels within a 2 pixel radius of the expected barrel location. Our verification program will run through the entire output image and compare each pixel with nearest neighbors (2px) on the sample output, if there is a match, we would consider this phase a success, else we will have to go back and debug.

Appendix:

1. Angular velocity and acceleration data acquired from this study:

   http://scholar.lib.vt.edu/theses/available/etd-08182005-222028/unrestricted/thesis.pdf

2. How barrel distortion works

   https://www.youtube.com/watch?v=B7qrgrrHry0&feature=youtu.be&t=11m26s

   Barrel Distortion

   http://stackoverflow.com/questions/28130618/what-ist-the-correct-oculus-rift-barrel-distortion-radius-function