FollowThrough Application

Test

Philip Habib
habibp

Djordje Petrovic
petrod

Zayan Imtiaz
imtiaz

George Plukov
plukovga

November 2016

Contents

1 Overview

The purpose of this document is to provide a plan for testing our FollowThrough application. The following outline gives a description of what is covered in this document.

- A proof of concept test is detailed in section 2.
- System testing is described in section 3.
- Software requirements specification testing is described in section 4.
- A time-line of the Test Plan is given in section 5.

2 Proof of Concept Testing

Before being able to do any other testing, there needs to be a proof of concept test. This test will be in place to ensure the project at hand is feasible in the first place.

2.1 Significant Risks

To have this project be successful we would first need to overcome a small number of major risks. Those risks are as follows:

1. The software needs to be able to successfully track the ball no matter where it goes. This is the basis of the entire system and if it doesn't work the project is no longer feasible.
2. The next most important risk to overcome would be making sure that the hardware is well protected in its environment. This could either be inclement weather or physical objects making contact with the gear.
3. Lastly, the software needs to be able to successfully upload/update any information over a server. If the server is to crash or the connection becomes disconnected the software must retain the information in a different way.

2.2 Demonstration Plan

The plan for the demonstration is to present a video of one of the developers playing basketball and have the software track the ball. The cases for this demonstration will be as follows:

- Starting at a 90 degree angle to the basketball net.
- Make a few shots, some are intentionally missed and some are intentionally made.

This demonstration is done this way to show that the ball can be tracked properly, even though it will be losing sight of the ball for a fraction of a second as it passes through the hoop or off screen.

Test 2.2.1	**Proof of Concept**
Description:	This test is meant to identify if the major risks are overcomable
Type:	Proof of Concept (manual)
Tester(s):	Developers
Pass:	This test will pass upon the successful demonstration of the project

3 System Testing

The system testing will be composed of three major sections. There will be the OpenCV/Python program testing that ensures the program processes the video correctly. There will the hardware testing which will ensure the Camera and Rasberry Pi is functioning correctly. Finally there will be website testing. Test cases for each of these sections will be detailed below.

3.1 OpenCV

The following tests will ensure the OpenCV/Python program is functioning correctly.

Test 3.1.1	**Recognize Basketball**
Description:	This test will consist of the OpenCV program detecting a basketball if it is within the webcam's view. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	A basketball is within the camera view.
Pass:	This test passes any standard basketball placed in front of the webcam can be immediately detected by the OpenCV/Python program.

Test 3.1.2	**Basket Ball Tracking Test**
Description:	This test will consist of the OpenCV program tracking the basketball by highlighting it as it moves around court. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	The user is holding the basketball.
Pass:	This test passes if the ball can be tracked regardless of speed and direction when the user shoots it as long as it's within view of the camera.

Test 3.1.3	**Arc Tracking Test**
Description:	This test will consist of the OpenCV program tracking the basketball's arc by forming a line of its trajectory as it moves around court. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	The user is holding the basketball.
Pass:	This test passes if the Open CV program correctly follows the ball's arc and forms a line in the program that indicates the arc of the shot.

Test 3.1.4	**Arc Analyzing Test**
Description:	This test will consist of the Python program processing the user's shot arc made by shooting a basketball and comparing it to the an ideal shot arc. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	The program receives a finished arc of the user's shot in the form of a graph.
Pass:	This test passes if the Python program correctly compares the user's shot arc with the ideal basketball shot arc and displays feedback regarding the differences as a line of best fit.

3.2 Web Application

The following tests ensure that the web application, and the server it's running on, is functioning properly.

Test 3.2.1	**Upload data to server test**
Description:	This test will consist of using GET and POST methods to upload sample data to the web server. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	Sample data is present locally.
Pass:	This test passes if the sample data is successfully uploaded to the server.

Test 3.2.2	**User can register to the web application**
Description:	This test will consist of using GET and POST methods to ensure users are saved in the system.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	User is not registered and cannot log in.
Pass:	This test passes if the the sample user is successfully saved to the server.

Test 3.2.3	**User can log in**
Description:	This test will consist of validating that the login process to the web application works. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	User access the website and is not logged in.
Pass:	This test passes if the user can successfully log in.

Test 3.2.4	**User can retrieve data**
Description:	This test will consist of validating whether or not the user can retrieve the data sent from the OpenCV application. This is a functional requirement.z
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	There is no data present on the web application.
Pass:	This test passes if the user can view the recorded data of their shot on the web application.

3.3 Hoop Hardware

The hardware going underneath the net will be primarily comprised of a Raspberry Pi, a Raspberry Pi (RPi) camera module and a infrared distance measurer. This is a second camera, separate from the main camera.

Test 3.3.1	**Raspberry Pi setup test**
Description:	This test will consist of a user attempting to attach the RPi device to the rim. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	User
Initial State:	The user has not yet set up the device.
Pass:	This test passes if the user can successful set up the RPi and have it upload data to the web server.

Test 3.3.2	**Player Position Tracking Test**
Description:	This test will include a player standing on the court and moving around. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	The tester is holding the basketball on the court, in view of the hoop camera.
Pass:	This pass tests if the tester's location can be tracked within a range of about 1 foot.

Test 3.3.3	**Scored Basket Test**
Description:	This test will include a player shooting the ball through the hoop. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	The tester is standing on the court and shooting a ball.
Pass:	This test passes if the user's shot has scored and was tracked properly by the IR sensor.

Test 3.3.4	**Missed Basket Test**
Description:	This test will include a player shooting the ball at the hoop but not scoring any points. This is a functional requirement.
Type:	Functional (manual)
Tester(s):	Developers
Initial State:	The tester is standing on the court and shooting a ball.
Pass:	This test passes if the user's shot has scored and was tracked properly by the IR sensor.

4 Requirements Testing

After the project is implemented and tested, the functional and non-functional requirements are to be examined. The functional requirements in this case will all be tested during the general system testing whereas the non-functional requirements will have to be tested separately.

4.1 Functional Requirements Testing

All of the functional requirements are tested in the System Testing section.

4.2 Non-Functional Requirements Testing

The test cases for the non-functional requirements are as follows:

Test 4.2.1	**Look and Feel Requirements**
Description:	This test will consist of someone looking at the final product and reviewing the outlook.
Type:	Aesthetic (manual)
Tester(s):	Developers
Pass:	This test passing is contingent on the opinion of various testers. If there is a unanimous agreement by the developers on the layout, the program has succeeded.

Test 4.2.2	**Ease of Use Requirements**
Description:	This test is being done to ensure that the product can be used by a variety of people.
Type:	Functional (manual)
Tester(s):	Developers
Pass:	The condition to pass this test is having multiple users agree that it is easy to use.

Test 4.2.3	**Learning Requirements**
Description:	This test is meant to identify how easy the system is to learn. It will be done by having someone outside the developers try it out for the first time.
Type:	Functional (manual)
Tester(s):	Random Person
Pass:	To pass this test the user at hand must be able to get a quick grasp of how the program works.

Test 4.3.4	**Speed and Latency Requirements**
Description:	This test is in place to ensure that the user is given feedback in a timely manner.
Type:	Functional (manual)
Tester(s):	Developers
Pass:	This test is passed when the software returns feedback to the tester within 0.5-4 seconds.

Test 4.3.5	**Precision or Accuracy Requirements**
Description:	This test is supposed to find out how accurate the feedback the software provides is.
Type:	Functional (manual)
Tester(s):	Developers
Pass:	For the software to pass the feedback must be approved by the testers.

Test 4.3.6	**Capacity Requirements**
Description:	The software will eventually accommodate multiple people.
Type:	Functional (manual)
Tester(s):	Developers
Pass:	This software passes the test if it works properly with more than one person on the court.

Test 4.3.7	**Multi-Platform Requirements**
Description:	This test is meant to find out whether or not the software functions over multiple operating systems.
Type:	Functional (manual)
Tester(s):	Developers
Pass:	This test passes if it works on Windows 10, Linux and OSX.

Test 4.3.8	**Language Requirements**
Description:	This test is meant to ensure proper grammar and spelling is used throughout the application.
Type:	Functional (manual)
Tester(s):	Developers
Pass:	This test passes if all of the grammar and spelling in the application is correct.

5 Timeline

This document is structured to roughly match the anticipated chronology of testing. A proposed testing timeline is given in

Table 1: Testing timeline

Completion Date	**Responsible Party**	**Task**
11/15/2016	George Plukov	Record all video test cases.
11/20/2015	Development Team	Demo 1, test system with basic basketball shot videos and arc tracking
11/25/2016	Development Team	Create python script to automatically test opencv program with all test videos
1/15/2017	Philip Habib	Begin testing of web application post/get methods
1/20/2017	George Plukov	Run initial tests for player position testing (Hoop Camera)
2/20/2015	George Plukov	Begin testing basket IR sensor
03/20/2017	Development team	Completion of all web testing
03/20/2017	Development team	Completion of hoop hardware testing
04/07/2017	Development team	Ensure all test cases pass for the three different components