Text to Motion Database

Test Plan

Brendan Duke
Andrew Kohnen
Udip Patel
David Pitkanen
Jordan Viveiros

March 27, 2017

Contents

Revision History

Date	Version	Notes
October 31, 2016	0.0	File created
November 2, 2016	0.1	Revision 0

1 Overview

The purpose of this document is to provide a detailed test plan for the McMaster Text to Motion Database including each subsection that makes up the combined project. The content of this document is covered below:

- Our proof of concept is described in Section 2.
- The set of tests that will test the system as a whole are in Section 3.
- The tests that ensure proper constraints on the system are in Section 4.
- The set of tests that verify functional requirements have been met are in Section 5.
- The set of tests that verify non-functional requirements have been met are in Section 6.
- A timeline of the test plan is given in Section 7.
- Appendix is found in Section 8.

1.1 Test Case Format

The format for the automated test case and manual test case can be found below with descriptions for each section of a test case:

Test 1.1.1:	**Test Name**
Description:	A description of what is being automatically tested
Type:	The type of test (Automated)
Initial State:	The initial state of the system for the test
Input:	The input that is required to complete the test
Output:	The desired output from the system
Pass:	What conditions must be met in order to be a success
Req. #:	The requirement number

Test 1.1.2:	**Test Name**
Description:	A description of what is being manually tested
Type:	The type of test (Manual)
Testers:	The user who will perform the test
Pass:	What the user will use to determine if the test was passed
Req. #:	The requirement number

1.2 Automated Testing

Automated testing for this project will be utilized in each testing section, and will require some software or custom application in order to perform the testing. This testing is critical to each section of the project and is divided into separate requirement sections. The division is required in order to ensure that testing can occur while still in development, i.e. tests for leaf modules can be executed first.

1.2.1 Testing Tools

1.3 Manual Testing

Manual testing will be used in the project when automated testing would require too much additional development, or expertise that cannot be represented through an application. These tests will be done by the external supervisor and his research team along with user test groups in order to remove any bias that the development team would have.

1.3.1 User Experience Testing

Some manual testing will consist of a user experience test in order to assess the experience of the project. They will be asked about the look and feel of the project website, gathering information, and other functionality that is incorporated. Because of the range of complexity within the project the test groups will have to contain different expertise and knowledge about software development.

2 Proof of Concept Testing

Before the project can enter a serious stage of development, a proof of concept must be accomplished in order to prove that difficulties can be overcome in order to prove the need for and usefulness of the project. Sections of this proof of concept are explained in more detail below.

2.1 Significant Risks

The significant risks of this project can be separated into three sections represented by the website, database, and deep learning network. The significant risks for each section and for the project as a whole are shown below:

- Linking the three major sections together in order to have them working in tandem is the largest risk to the project's success, as one failing will directly impact the others.
- The website must use a form of query to the database in order to correctly return the desired information.
- The database must contain a pairing of video and text descriptions in order to provide the correct information to the users and developers.
- Deep learning and pose estimation are active areas of research and have steep learning curves.

2.2 Demonstration Plan

In order to show that the significant risks can be overcome the demonstration will have a set of deliverables that will provide the base for how successful the proof of concept is. Each key deliverable can be found below:

- A functional website, as an interface for running pose estimation.
- Said website should contain a database.
- The ability to upload images and videos, and to update the database with those uploaded data.
- The ability to run human pose estimation on any uploaded image and video. Human pose estimation can be explained as follows: the skeletons and joints of any humans in the uploaded media will be indicated visually.

- The ability to search for uploaded images and videos through some means, e.g. by tag or name.

2.3 Proof of Concept Test

Each deliverable has a test case that should be passed in order to determine how successful the proof of concept is. The test cases are considered to be simple manual tests as the group members will be performing them in isolation or during the demonstration.

Test 2.3.1:	**Functional website for pose estimation**
Description:	A website that contains the ability to run pose estimation on a video or image
Type:	Manual
Testers:	Development team
Pass:	The user should be able to run some form of pose estimation through the web interface that is presented to them
Req. #:	N/A

Test 2.3.2:	**Database and website pairing**
Description:	The database will be paired to the website in order to provide videos or images through the website
Type:	Manual
Testers:	Development team
Pass:	If the user can access data that is stored within the database through the web interface, then the test can be considered a pass
Req. #:	N/A

Test 2.3.3:	**Updating the database**
Description:	The ability to upload images or videos and update the database through the website
Type:	Manual
Testers:	Development team
Pass:	If the user can upload an image or video through the web interface and the user can later access the uploaded image or video the test can be considered passed
Req. #:	N/A

Test 2.3.4:	**Running pose estimation**
Description:	The ability to run pose estimation on an uploaded image or video, which entails that the skeleton and joints of any human, will be clearly visualized
Type:	Manual
Testers:	Development team
Pass:	In order for this test to be considered passed the user will have to upload an image or video and see that the humans are correctly represented by their respective joint positions
Req. #:	N/A

Test 2.3.5:	**Search by tag or name**
Description:	The ability to search through the database by either an associated tag or name.
Type:	Manual
Testers:	Development team
Pass:	In order for this test the user will have to search for an image or video that was uploaded with a specific tag or name and be directed to the correct image or video
Req. #:	N/A

2.4 Solution Constraints Testing

Test 2.4.1:	**Deep Learning Methods Test**
Description:	Test whether the human pose estimation component of the software uses modern deep learning methods.
Type:	Manual
Testers:	Supervisor (Dr. Taylor)
Pass:	Dr. Taylor should confirm that the deep learning methods used are satisfactory and relevant to current research, with a yes or no response.
Req. #:	1

Test 2.4.2:	**Standard Data Format Test**
Description:	Tests whether the human pose data format used in the project is standard, and compatible with existing software libraries.
Type:	Automated
Initial State:	Initialize database query interface.
Input:	Random ID of a record, containing human pose data, in the database.
Output:	Tuple containing data in HDF5 format.
Pass:	The human pose datum should be parse-able by an existing HDF5 data library.
Req. #:	2

Test 2.4.3:	**Linux Platform Build and Run Test**
Description:	Confirms that all nightly build tests, as well as the automated test suite, are working under Linux.
Type:	Automated
Initial State:	None (build test).
Input:	Commands to begin build and run sequence.
Output:	Compile and run success, or errors.
Pass:	Compile and run success.
Req. #:	3

Test 2.4.4:	**Python API Hook Testing**
Description:	Confirms that major module interfaces, such as the image pose estimation interface, and database query interface, have working Python hooks.
Type:	Automated
Initial State:	Initialization specific to each module interface under test.
Input:	Valid parameters for each module interface, written in Python.
Output:	Expected success-case outputs for each module interface, written in Python.
Pass:	Interface calls completed without error, and returned their expected outputs.
Req. #:	4

3 Functional Requirements Testing

Test 3.1:	**Supported Video Encodings Test**
Description:	Tests whether the ReadFrames API is able to decode MP4, MP2 and AAC video files.
Type:	Automated
Initial State:	Call read frames initialization procedure.
Input:	30 second MP4 video file at 30 FPS.
Output:	A set of 900 (30×30) frames.
Pass:	The 900 frames match a set of 900 expected frames from a reference frame-reading system.
Req. #:	7

Test 3.2:	**Frame Reading Timestamp Accuracy Test**
Description:	Tests whether the timestamps on the frames returned by the ReadFrames API match their temporal position in the original video stream.
Type:	Automated
Initial State:	Call read frames initialization procedure.
Input:	30 second MP4 video file at 30 FPS.
Output:	A set of 900 (30×30) frames, which include timestamps.
Pass:	The timestamps on the 900 frames match a set of timestamps on a test vector of expected timestamps for the 900 frames.
Req. #:	8

Test 3.3:	**Video Human Pose Estimation Data Quality Test**
Description:	Test to ensure the data quality produced by the human pose estimator component. A set of Charades videos will be processed by the human pose estimator, and skeleton animations corresponding to the generated human pose data will be created (this is a scoped part of the software pipeline). A double-blind test will be ran, where testers will be shown random mixed sets of the skeleton animations produced by McMaster Text to Motion, together with skeletons from actual motion capture data coming from CMU's motion capture lab. Testers will indicate whether they think the motion capture data came from actual motion capture, or from the pose estimation software.
Type:	Manual
Testers:	Testing Group
Pass:	Within a 5% confidence interval, the McMaster Text to Motion skeletons will be indicated as being actual motion capture data with the same probability that the CMU motion capture skeletons are indicated as being actual motion capture data.
Req. #:	8

Test 3.4:	**Database Output Full Range Coverage Test**
Description:	Tests whether the range of the text-to-motion database search is equal to the entire set of data stored in the database.
Type:	Automated
Initial State:	Initialize database-query and full text search module interfaces. Populate database with Charades data.
Input:	A random matching keyword from the text description of each video (acquired automatically).
Output:	A set of video-pose data from the database that should include the original datum that the input keyword was taken from.
Pass:	The returned set of data contains the original video record.
Req. #:	9

Test 3.5:	**Database No False Positives Test**
Description:	Tests whether the results retrieved from text searches of the database contain any false positives, i.e. results whose text descriptions do not contain any of the searched keywords.
Type:	Automated
Initial State:	Initialize database-query and full text search module interfaces. Populate database with Charades data.
Input:	For each video, a random set of keywords not in that video's text description.
Output:	A set of video-pose entries.
Pass:	The output set of data should not contain the original video that was chosen to be outside the subset of the output range for this input.
Req. #:	10

Test 3.6:	**Full Text Search Order by Relevance Test**
Description:	A test of whether the full text search interface is returning a set of entries that are ordered by relevance to the search keywords.
Type:	Automated
Initial State:	Initialize database-query and full text search module interfaces. Populate database with Charades data.
Input:	A random set of search keywords, drawn automatically from the set of text descriptions in the database.
Output:	A set of entries in the database, in some order.
Pass:	The output set of entries should be randomly ordered and input to a reference full text search engine, which will produce an expected ordering by relevance. A statistical test of the similarity of the McMaster Text to Motion ordering and the reference ordering should be done, and the McMaster Text to Motion ordering should be expected to be the same within a 5% confidence interval.
Req. #:	11

4 Non-Functional Requirements Testing

4.1 Look and Feel Requirements Testing

Test 4.1.1:	**Colour Scheme Test**
Description:	Test user satisfaction of the web interface colour scheme.
Type:	Manual
Testers:	Testing Group
Pass:	On a one to ten scale, the average user rating is above six.
Req. #:	12

4.2 Style Requirements

Test 4.2.1:	**Minimalistic Web Design**
Description:	The website should use a minimalistic and visually informed design through visual aids.
Type:	Manual
Testers:	Testing Group
Pass:	An average rank of 6 with a one to ten rating scale from the test group.
Req. #:	13

4.3 Ease of Use Requirements

Test 4.3.1:	**Upload/Download media**
Description:	Through the web interface a casual user should be able to upload and download files to the database with ease.
Type:	Manual
Testers:	Testing Group
Pass:	Observe the test group and ask them to upload and download an image in order to record the time required. Any time below 30 seconds for each task will be considered a pass.
Req. #:	14

Test 4.3.2:	**User Interaction Test**
Description:	The user should be able to interact with the website without any previous knowledge
Type:	Manual
Testers:	Testing Group
Pass:	On a scale or questionnaire the users will be asked to determine the ease of use when interacting with the website.
Req. #:	15

Test 4.3.3:	**Text Box Functionality**
Description:	The user should be able to input a descriptive word or phrase into a text-box from within the web interface.
Type:	Manual
Testers:	Testing Group
Pass:	Observing the test group when asked to search for a video and record the time taken to find and input text. Anything below 15 seconds will be considered a pass.
Req. #:	16

4.4 Learning Requirements

Test 4.4.1:	**End-User Prior Training**
Description:	Any end-user should be able to use the website without any previous training or understanding of the components.
Type:	Manual
Testers:	Testing Group
Pass:	Users will use the website and rate how easy it was to use on a one to ten scale. An average of 6 will be required to pass.
Req. #:	17

Test 4.4.2:	**Text To Motion Training**
Description:	Users should be able to instantiate a text to motion event without having any prior training.
Type:	Manual
Testers:	Testing Group
Pass:	Users will be asked to instantiate a text to motion event and record the time it takes. Any time below 30 seconds will be considered a pass.
Req. #:	18

Test 4.4.3:	**Software API**
Description:	Anyone familiar with software APIs should be able to easily understand
Type:	Manual
Testers:	Testing Group (Programmers)
Pass:	Users will be asked to set up the Software Interface and use some base functionalities within 15 minutes to be considered a pass.
Req. #:	19

4.5 Understandability and Politeness Requirements

Test 4.5.1:	**Hiding The Inner Workings**
Description:	Users should not be able to see the deep learning model and its training when using the pose estimation. When prompted the website should display the correct skeletons without any low-level detail.
Type:	Manual
Testers:	Development Team
Pass:	When using the web interface to get a pose estimation from media, the output should only be a video with the skeleton overlay. Beyond this there should not be any additional information or access to any information.
Req. #:	20

4.6 Speed and Latency Requirements

Test 4.6.1:	**External Database Connection Response Time**
Description:	The web interface should be able to connect to an external database and store or query items.
Type:	Automated
Initial State:	Initialize the web interface.
Input:	An image or video with a human in frame.
Output:	The output from the web interface would be a confirmation of the image or video being uploaded, along with an associated database query.
Pass:	In order for this test to be considered a pass the confirmation of the image being uploaded would have to occur within 30 seconds so that additional resources are not wasted by the database.
Req. #:	21

Test 4.6.2:	**Deep Learning Model Response Time**
Description:	The deep learning model should be able to process an image in a relative time of one frame per minute.
Type:	Automated
Initial State:	Initialize the web interface.
Input:	An image or video with a human in frame.
Output:	A pose estimated image with a skeleton overlay of each joint in a frame by frame instances.
Pass:	In order to pass this test the deep learning model would have to adhere to the frame per minute rule set in the description, and a five frame video would take a maximum of five minutes.
Req. #:	22

Test 4.6.3:	**Website Search Responsiveness**
Description:	When given a word or phrase the web interface will be able to respond with an image or video of a pose or action within a reasonable time.
Type:	Automated
Initial State:	Initialize the web interface.
Input:	Text of a word or phrase within the search field.
Output:	An image or video matching the input description.
Pass:	Providing a matching video or image within two minutes will constitute a passed test as the web interface has to parse the input, interpret the pose that best matches, and output the match.
Req. #:	23

5 Precision or Accuracy Requirements

Test 5.1:	**Bone and Joint Position**
Description:	The project must be able to accurately encapsulate and represent the bone and joint positioning of the human that is found within the frame.
Type:	Manual
Testers:	Development Team
Pass:	In order to pass this test the program must accurately represent the positioning of bones and joints with regards to an average human. This will be determined through visual confirmation that the bone or joint positioning overlay is within an uncertainty of 20 pixels.
Req. #:	24

6 Reliability and Availability Requirements

Test 6.1:	**Software Availability**
Description:	The software component of the project should be available 24 hours a day, and 365 days a year.
Type:	Automated
Initial State:	Initial software interface during a scheduling constraint.
Input:	Any command that requires the API.
Output:	The correct response to the input command.
Pass:	If the output matches the input and correctly responds, it can be assumed that the software will work during a scheduling constraint and be considered a pass.
Req. #:	25

Test 6.2:	**Website Availability**
Description:	The web component of the project should be available 24 hours a day, and 365 days a year. The only exception to this is scheduled maintenance or migration
Type:	Automated
Initial State:	Initial web interface.
Input:	A call to the web server.
Output:	A response that signifies the call has been processed and the server is still up and running.
Pass:	In order for this test to be passed the call would have to return a HTTP verified response like 2xx in order to represent a successful connection.
Req. #:	26

7 Robustness or Fault-Tolerance Requirements

Test 7.1:	**Web Interface Error Handling**
Description:	The web interface should respond to unhandled exceptions by throwing the corresponding error messages.
Type:	Automated
Initial State:	Initial web interface.
Input:	A known error or misuse of the web interface.
Output:	The corresponding error message to the input.
Pass:	If the error message correctly responds to the input or contains a catch all, then the test can be considered a pass.
Req. #:	27

Test 7.2:	**Web Interface Text Parsing**
Description:	The web interface will have to parse the text that is input in order to determine if the input is unintelligible.
Type:	Automated
Initial State:	Initial web interface.
Input:	An unintelligible word or phrase.
Output:	An error message containing the unintelligible word.
Pass:	In order for this test to pass the web interface must respond to the faulty input with a corresponding error message.
Req. #:	28

8 Capacity Requirements

Test 8.1:	**Multiple Connections**
Description:	The web interface should be able to serve multiple connections.
Type:	Automated
Initial State:	Initial web interfaces.
Input:	5 web interfaces that are open.
Output:	5 web interfaces that continue to run and function as intended.
Pass:	If each web interface can fully serve the 5 users and respond to each action as intended, then the test can be considered a pass.
Req. #:	29

Test 8.2:	**Database Capacity**
Description:	The database should contain a large amount of information when first created in order to facilitate growth.
Type:	Manual
Testers:	Development Team
Pass:	The development team should have at least 5GB of data stored in the database when complete to be considered a pass.
Req. #:	30

9 Scaling of Extensibility Requirements

Test 9.1:	**Deep Learning Training**
Description:	The deep learning model should be put through a rigorous test set, with expected outputs, in order to accurately represent the pose estimation on humans.
Type:	Manual
Testers:	Testing Group
Pass:	The pass for this test will come from a test set that should contain thousands of pictures to be trained with. In addition to this test set, the success of other tests or core functionality will prove if the deep learning model was properly trained or not.
Req. #:	31

10 Operational and Environmental Requirements

Test 10.1:	**Linux Friendly Tensorflow**
Description:	The web interface should be run on a Linux friendly server that can access the Tensorflow model either directly or indirectly.
Type:	Manual
Testers:	Development Team
Pass:	The development team will design the web interface to run on an Apache or NGINX server. This test case will be considered a pass when said server is functioning correctly.
Req. #:	32

Test 10.2:	**Tensorflow Library and Model**
Description:	The web interface should interact with the Tensorflow library, as the deep learning model cannot be run on the web interface alone.
Type:	Automated
Initial State:	Initial web interface
Input:	An image or video.
Output:	A yes or no response to the question 'Is there a person in this media?'
Pass:	If the Tensorflow model can correctly determine if there is a person within the media provided the test will be passed.
Req. #:	33

Test 10.3:	**Export types**
Description:	The project should be able to export multiple types of media (JPEG, PNG, etc) in order to support all major operating systems.
Type:	Manual
Testers:	Development Team
Pass:	The development team should be able to export an image as multiple types such as JPEs, PNG, DDS, and more in order to pass this test.
Req. #:	34