SOFTWARE TEST REPORT

MCS13

FIT 3162

MEMBERS

Matin Raj Sundara Raj (32124260)
Suah Wen Hung (30721083)
Louis Juliano (31507794)

BadminSmash - Visual Analysis of Badminton Shot Styles

Table of Contents

1.0 Introduction

The aim of this Test Report is to document the various tests that have been carried out by the team, in order to prove the effective and adequate implementation of our software. The tests performed will highlight the requirements that we have met, as set by the supervisor, while proving correctness and accuracy. This report will also provide insight into any limitations that our software currently has through the analysis of the test results. It is important to perform and document these tests as it will help the team to recognise any form of success or shortcomings that require attention.

2.0 Description of Test Approach

The main testing method is to manually key in input values (where applicable) and perform actions based on testing methods. The outcome of each test is then crosschecked with what is desired.

Unit tests are done on the primary components of our software such as the Input GUI which users interact with, and the calculateAngle function that is applied to the skeleton landmarks which is used by our model to predict the shot style.

Blackbox testing is done by inputting videos of non-badminton content (as a naïve user would without prior knowledge regarding the software) and ensuring the outcome does not have undesired effects.

Integration tests are performed on every major stage in our software pipeline, such as input, TrackNet, mediapipe and classifier stage, to ensure that all components and libraries are working as intended.

System testing is done for each requirement and also to gauge the performance, accuracy and scalability of our software.

Finally, we gathered a test group of technical and non-technical users to try out our software. Later they were interviewed to provide a usability score and comment on various aspects of the software.

3.0 Unit Testing

Test focus	Input	Expected outcome(s)	Result
calculateAngle function			
Ensure that all the function is able to calculate edge cases	Inputs are all tested with realtime video input	With all points being 0, the expected outcome of the code is also 0.	PASS
Ensure that there are no angles that are larger than 180.		All angles that are larger than 180, will be subtracted by 360 so it will always give an angle that is smaller than 180.	PASS
Ensure that the angles calculated are correct		-	PASS

Test focus	Input	Test	Expected outcome(s)	Result
Input GUI				
Ensure only 'mp4' videos are accepted as input	A badminton video clip in mp4 format	Appendix 1.1	File that are not mp4 format are greyed out and user will not be able to select when browsing for input file	PASS
Enter invalid input file path	An invalid file path	Manually type in non-existent input file path into the input box	Return Error code 1 on terminal	PASS
Enter invalid output path	An invalid output file path	Manually type in non-existent output file path into the input box	Return Error code 2 on terminal	PASS
Empty Input/Output path	An empty string as input/output path	Click 'Begin Analysis' button directly without browsing for any input/output location Appendix 1.2	Nothing occurs	PASS
Ensure Analysis terminates on keyboard input	Keyboard Input	Press on 'q' key when analysis is running on preview window	The preview window closes, but main program is still running and Input GUI is still functioning	PASS

Ensure Output video exist and is properly stored in specified output path	A valid output path	Browse to a directory by manually clicking on 'Browse button' in Output section	An output video containing all processed annotations in mp4 format exists in the specified output directory	PASS
Ensure program exits safely	-	Click 'Exit' button	Input GUI closes and program is terminated	PASS

4.0 Blackbox Testing

In our unit testing, the enforcement of only allowing on 'mp4' format files as input has already been tested. Hence, for Blackbox testing, we will only cover different video content as input.

Input	Screenshot	Output	Expected outcome(s)
Badminton video	Appendix 2.1	Appendix 2.2	Video outputted correctly with pose detection, ball detection and shot style annotation.
Sport video (Jogging)	Appendix 2.3	Appendix 2.4	Video outputted correctly with only pose detection.
Normal video (Car traffic)	Appendix 2.5	Appendix 2.6	Video outputted correctly with no functionality as no human detected.

5.0 Integration Testing

This test has been separated into two parts. The first is a test on each workflow of the system, while the other is a test on the few important single components needed for the system to work as intended.

Function call	Screenshot	Output	Expected outcome(s)	RESULT
Workflow testing				
Input Stage				
input_path = tk.filedialog.askopenfilename(filetypes=[("Video files", "*.mp4")])	Appendix 2.1	-	Input video inserted correctly as expected	PASS
TrackNet Stage				
Predict_tracknet = model.summary()	Appendix 2.1	Appendix 2.7	Video outputted correctly with accurate ball tracking prediction	PASS
MediaPipe Stage				
mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_pose.POSE_CONNECTIONS)	Appendix 2.7	Appendix 2.8	Video outputted correctly with ball and pose detection	PASS
Classifier Stage				
cv2.putText(image, model_class, [25, 25], cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 255, 255), 2, cv2.LINE_AA)	Appendix 2.7	Appendix 2.2	Video outputted correctly with pose detection, ball detection and shot style annotation.	PASS
Additional libraries testing				
Reading frames				
ret, frame = cap.read()	badminton.mp4	True value and a singular frame	Cv2 library returns frame to be used.	PASS
Outputting video				
output_path = tk.filedialog.askdirectory()	C:/User/Output Path/	badminton_output.mp4	Tkinter library asks for a directory and return a file name	PASS

Calculating angle				
radian = np.arctan2(c[1] - b[1], c[0] - b[0]) - np.arctan2(a[1] - b[1], a[0] - b[0])	a, b, c values = [0.56, 0.62, 0.71]	Radian value = 3.191	Numpy library calculates the tan value to retrieve radian value for calculation	PASS

6.0 System Testing

6.1 Performance

Test focus	Input	Outcome
Performance of the code with TrackNetV2 integration on a short video	52 seconds video	Takes too long to predict together with TrackNetV2. Video is run by the code 3 times which causes the product to run extremely slow. With input of 58 seconds, and a strong computer specification. Time needed to complete prediction is about 15 minutes
Performance of the code with TrackNetV2 Integration on longer video	10 minutes video	TrackNetV2 never finished predicting the 10 minutes video after 30 minutes of waiting.
Performance of the model without TrackNetV2 on short video	52 seconds video	Model is able to run and predict the video in real time without any FPS drops.
Performance of the model without TrackNetV2 on longer video	10 minutes video	Model is able to run and predict the video in real time without any FPS drops.

6.2 Accuracy

The confusion matrix is used to estimate the accuracy of the model. To retrieve the confusion matrix, we use the sklearn library. Then with the help of seaborn and matplotlib, we are able to plot the confusion matrix into a graph. Testing method: Appendix 3.1
Highlighted in green is the True Positive and True Negative of the model. False positives and negatives are highlighted in red.

Logistic Regression
Test Outcome: Appendix 3.2
Accuracy: 69%

	backhand	forehand	netdrop	serve
backhand	5	2	5	0
forehand	1	41	4	5
netdrop	2	13	20	1
serve	0	1	2	16

Ridge Classifier
Test Outcome: Appendix 3.3
Accuracy: 69%

	backhand	forehand	netdrop	serve
backhand	4	2	6	0
forehand	1	40	4	6
netdrop	2	13	23	1
serve	0	2	2	15

Random Forest Classifier
Test Outcome: Appendix 3.4
Accuracy: 70% to 76%
Average Accuracy: 74%

	backhand	forehand	netdrop	serve
backhand	5	4	3	0
forehand	0	45	5	1
netdrop	4	11	21	0
serve	0	2	2	16

Gradient Boosting Classifier
Test Outcome: Appendix 3.5
Accuracy: 69% to 70%

	backhand	forehand	netdrop	serve
backhand	4	5	3	0
forehand	1	44	4	2
netdrop	3	10	21	2
serve	1	1	1	16

The model chosen by default is Random Forest Classifier

6.3 Scalability

Features	**Difficulty (1 to 5)**	**Reason**
Adding a new pose to estimate	**1**	In order to add a new shot style to estimate, new data for the pose are needed to be trained in the model. In simple terms, new datasets are needed to train to add new poses for the model to train.
Adding a new feature or prediction into the code	**4**	Adding new features into the code is not necessarily hard. However, it is more challenging on the performance as the code is already slow with our new model.
Adding new attributes to use for prediction	**3**	Adding a new attribute to predict is a little challenging due to new functions needing to be coded. However, the information of the landmarks are available to the user to utilise.

6.4 Requirement Test

ID	Requirements	Category	Test	Status
01	Product must be able to detect the player's position in real time.	Non-Functional	Appendix 2.8	PASS
02	Product must be able to show the player's skeletal pose in real time.	Non-Functional	Appendix 2.8	PASS
03	Product must be able to correctly predict the player's backhand shot pose.	Functional	Appendix 2.2	PASS
04	Product must be able to correctly predict the player's forehand shot pose.	Functional	Appendix 2.2	PASS
05	Product must be able to correctly predict the player's net drop shot pose.	Functional	Appendix 2.2	PASS
06	Product must be able to correctly predict the player's serve pose.	Functional	Appendix 2.2	PASS
07	Product must use a well-trained model to accurately predict badminton shots to display match statistics.	Non-Functional	Appendix 2.2	PASS
08	Product must be able to utilise TrackNetV2 to track the ball.	Functional	Appendix 2.7	PASS
09	Collect enough data to train the machine learning model	Functional	-	PASS

7.0 Usability

To test the program in a real-life scenario, usage of the program has been carried out with a few of the team's peers, both with technical understanding and those without, by requesting them to run their own intended badminton match video on the application. Then, a short interview was conducted with each user to get their feedback on the various aspects tested. Below highlights two combined responses with average scores, where one test group has a Computer Science background and the other with no technical background.

Combined response of common users with no technical background – Test Group size of 5

Questions Asked	Response/Ratings
Setting up	
How easy was it to set up the application?	2 / 5
How useful were the provided instructions?	4 / 5
Comments	"I didn't understand what dependencies or pip is but following the instructions was easy to get it to work"
User Interface	
How easy was it to use the path browser?	4 / 5
How useful was the labelling of the interface elements?	5 / 5
How visually appealing was the interface?	2 / 5
Comments	"Selecting my input video and output path was straightforward and easy." "Interface looks simple and could look nicer and given more colour."
Preview Screen	
Did the program function as you intended?	Yes
Did the program have any unexpected errors?	No
Choice of colour of annotation?	4 / 5
How useful was the annotation on-screen?	5 / 5

Comments	"Annotations were not too clustering, but text could be larger for easier reading." "I didn't understand the numbers below the shot styles, maybe it can be removed." "The red colour on both the ball and player made it easy to focus."
Overall Feedback	
"It took quite some time for my final output video to be done, maybe I can skip through everything without watching the whole video." "Output video appeared on the exact path I inserted after the video ending, which was nice."	

Combined response of technically capable users – Test group size of 5

Questions Asked	Response/Ratings
Setting up	
How easy was it to set up the application?	5 / 5
How useful were the provided instructions?	5 / 5
Comments	"Easy to set up"
User Interface	
How easy was it to use the path browser?	5 / 5
How useful was the labelling of the interface elements?	4 / 5
How visually appealing was the interface?	3 / 5
Comments	"Simple enough interface but could provide more info."
Preview Screen	
Did the program function as you intended?	Yes
Did the program have any unexpected errors?	No
Choice of colour of annotation?	3 / 5
How useful was the annotation on-screen?	4 / 5
Comments	· "Annotation confidence was useful to see the accuracy of the shot type." · "Frame rate was slightly slower than expected."
Overall Feedback	
"Video took a while to compile, spent some time just waiting." "The provided interface is way easier to use than TrackNet's argument based input." **"**Need more information on what Is happening when video is processing in the background. A loading bar may be better"	

8.0 Discussion of Test Results and Software Limitations

The results of Unit and Integration testing shows that our software features and code are working as intended with all the requirement being successfully and correctly implemented. The test results have shown that the model that we are using is a moderately accurate model. The model's algorithm that we have chosen is the Random Forest Classifier as the model has an accuracy of 70% to 76% with an average of 74%. With an accuracy of 74%, we are able to predict multiple poses successfully and correctly. With more data, we should be able to increase the accuracy of the model.

As for the performance of the code, unfortunately, the performance of the code is bad. This is due to the number of times the video is replayed. TrackNetV2 requires the video to be run twice as our product requires the video to be run once. In total, 3 runs are required in order to fully predict the video. Due to this, we aren't able to obtain a much more efficient code. This introduces longer wait times for the video to be processed in the background before analysis and pose estimation can be done.

The issue of long wait times is also highlighted during usability testing by multiple participants in the test group. While they commended the software's ease of use, they also expressed the need to view more information when video is being processed in the background.

9.0 Recommendations for Improvement

There are things that we are able to improve for this project. The first item that needs improvement is the performance of the project. Currently, we run the project by predicting using TrackNetV2, then annotating the trajectory of the ball with TrackNetV2 and lastly, predicting the shot style of the player. One improvement we can do is while TrackNetV2 is predicting, our model should also predict the shot style of the player. With this, we are able to improve a lot on the performance of the product.

Another improvement we can do is having multiple mediapipe objects to predict multiple players. Currently, we are predicting one player and there is a chance that mediapipe might mistake other objects as players. By having multiple pose predictions, we are able to narrow down which one is the actual player and whose shot style to predict, thus improving accuracy and reliability.

We can also provide more information to users through the terminal or Input GUI when the video is processing in the background. An example would be, detailed descriptions of the current processing pipeline, with a loading bar to indicate progress.

10.0 Discussion of Limitation Of Testing Process

The tests that were carried out in this report are all done manually. Although we believe that we have covered sufficient input cases and scenarios, we were not able to verify the robustness of our software by covering large test sets catering to every permutation. An improvement in the testing process that the team could have done given additional time would be to automate some of the mundane manual input testing methods. This would have allowed the team to perform a greater number of tests in a lesser amount of time. Although it is to be said that the design of our software does not facilitate the use of large automated test sets as there is not much variability that can occur with the software.

A test that is not done is the separate accuracy measurement of each shot style type. This test is not performed as our current model was trained using a collage of datasets containing all the shot styles in one go. Thus, it is not feasible to know the accuracy of each shot style. This test could be performed if the machine-learning model was trained incrementally first by keeping track of its accuracy at each addition of a new shot style. Knowing the accuracy of a particular shot style may have enabled us to finetune it individually by adding more reliable training data.

11.0 Conclusion

The tests that were conducted in this report demonstrate a successful working product that meets all the requirements set by the supervisor. The team has successfully developed an Input GUI that makes it easy for users to interact with the software while reducing the dependency on the terminal for inputting video files. Through Integration testing, we know that the software is able to correctly and reliably annotate the shuttlecock and its trajectory using TrackNetV2, and annotate the various shot styles. The model that we have trained has a relatively high accuracy, thus ensuring the predicted shot style annotations on the output video corresponds to the player's actual pose.

Although our project has passed the various aspects that it has been tested, the performance issue is to be regarded as one of the software's shortcomings. We were not able to develop and successfully integrate TrackNetV2 in an efficient manner. Nevertheless, the test results show that this project has indeed been a success, covering all requirements.

12.0 Appendix

Reference	Data
1.1	`input_path = tk.filedialog.askopenfilename(filetypes=[("Video files", "*.mp4")])`

1.2	`if filename != '' and output_directory != '':`
	`  if Checkbutton1.get() == 1:`
	`  leftHandBool = 1`
	`  analyse(filename, output_directory, leftHandBool)`

2.1 Badminton video Input	
2.2 Badminton video Output	

2.3 Jogging video Input	
2.4 Jogging video Output	
2.5 Car traffic video Input	

2.6 Car traffic video Output	
2.7 Badminton video with TrackNet	
2.8 Badminton video with MediaPipe	

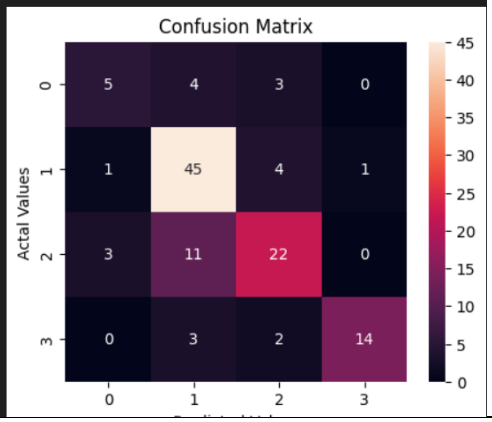
3.1

```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm = pd.DataFrame(cm)
```

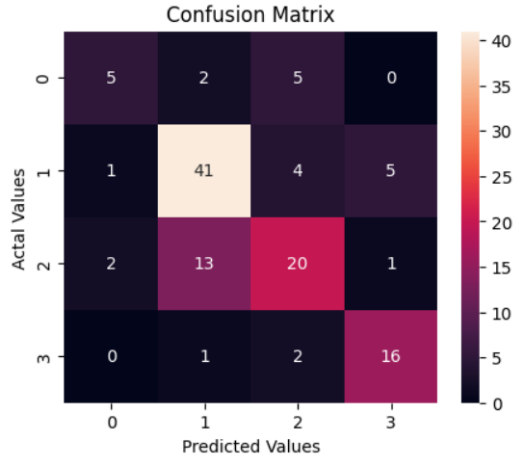
✓ 0.0s

```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True)
plt.title('Confusion Matrix')
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()
```

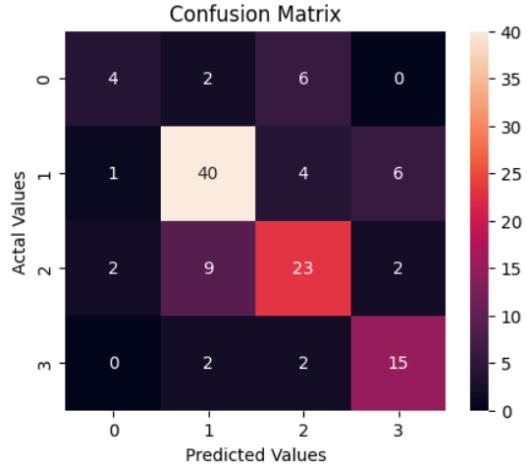
✓ 0.1s

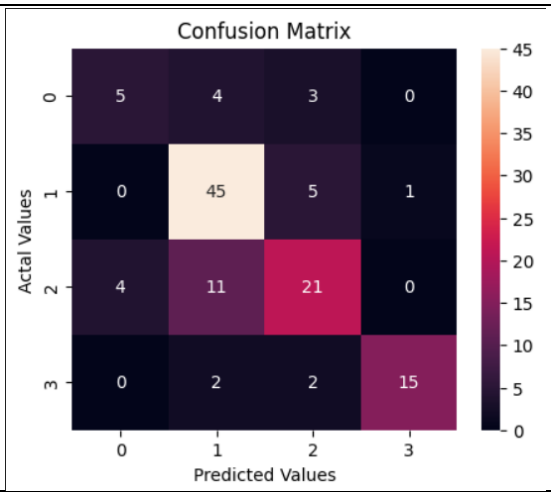
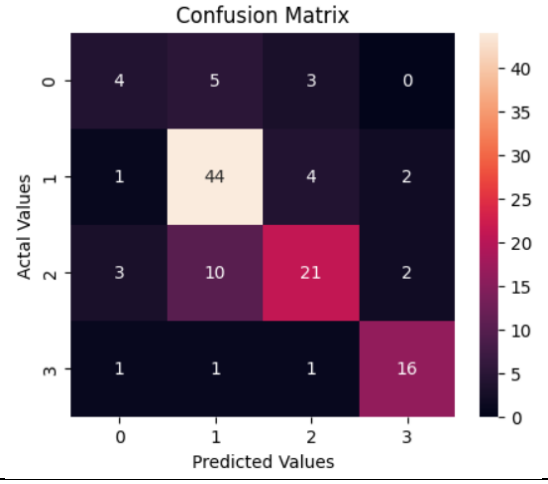


3.2



3.3



3.4	 Confusion Matrix Actal Values Predicted Values 5 4 3 0 0 45 5 1 4 11 21 0 0 2 2 15 0 1 2 3 45 40 35 30 25 20 15 10 5 0
3.5	 Confusion Matrix Actal Values Predicted Values 4 5 3 0 1 44 4 2 3 10 21 2 1 1 1 16 0 1 2 3 40 35 30 25 20 15 10 5 0