SCHOOL OF ELECTRICAL AND INFORMATION ENGINEERING

UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

ELEN4012 Project Plan - Group 19G05

Supervisor: Dr. Hugh Hunt

Classification of Lightning Events using Deep Learning

15 July 2019

Tyson Cross 1239448 Jason Smit 709363

Abstract

The project requires a lightning classification system from video input using deep learning. A procedure is presented detailing the critical sections of manual labelling, tool development, static classification system leading to temporal classification. Edge detection, background removal and noise reduction are indicated for automatic detection of the lightning. A data-centric CNN using the TensorFlow framework and transfer-learning is proposed for lighting detection, but temporal event classification requires experimentation. A suitable network topology is yet to be selected.

1 Introduction

This plan details the specifications, deadlines and the scheduled work-flow for building a system to detect and classify lighting events captured from high speed footage files, using deep-learning techniques. The requirements in § 2 are the foundation for the project and layout the problem domain. The scheduling for the project is presented as a Gantt chart in Appendix A. Data-science for machine learning [ML] involves an amount of exploration of the source data, and creative application of image-processing techniques. To accommodate this, an iterative workflow utilising a sprint-based methodology will be adopted, with weekly reviews on progress.

1.1 Glossary

This section details terms or words that will be used throughout the document. The items listed here may have multiple definitions or conflicting uses in various literature sources [1][2][3], so some definitions are listed here for clarity.

1.1.1 Leader: Leaders are electrically conductive channels, producing illumination from the ionised plasma. Leaders propagate due to the local differences in charge distributions (e.g. negatively charged clouds and positively charged ground.) If a stepped leader connects to a the ground (usually a raised structure) then this can develop into a stable channel between the charge differences resulting in a stroke as defined in § 1.1.2 [2] [3].

1.1.2 Stroke: A Stroke is a lightning event from initial contact by a leader which makes contact with an oppositely charged electric field. Referred to as connected leaders in some literature [2].

1.1.3 Strike: A strike is stroke but terminates when the lighting is no longer visible.

1.1.4 Flash frame: A frame where a large magnitude discharge results in clipping or over-exposure of luminance values in the image. The light produced results in blown out frames.

2 Requirements

The intention of the project is to investigate the feasibility of a deep learning system that is capable of accurately classifying sets of pixels from high-speed (HS) lightning footage as lightning events, and to identify the direction of the lightning stoke (Up, Down or horizontal).

The requirement of this project is to process HS video footage (in the proprietary `CINE` format), where large portions of the footage is a non-event. The raw input data consists of manually annotated, partially labelled video files. The output of the system should be of the form of a table format (xls or csv) which labels the time of a stroke and the duration of a stroke. The accuracy of the classification should be determined by inputting previously unseen data into the system, and comparing the output to the known labelled data for a validation metric. An accuracy of greater than 60 percent is acceptable for this application. A high level block diagram of the proposed system is shown in Figure 1.

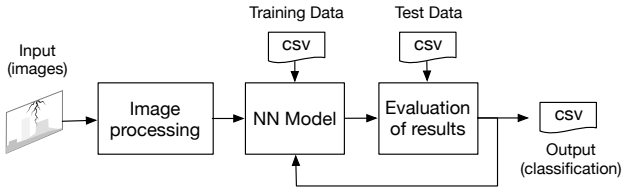


Fig. 1: System diagram

3 Procedure

It is proposed that the work-flow be divided into smaller sections to allow for scheduling and resource allocation. The sections are manual labelling, tool development (including video conversion and image pre-processing), static classification, temporal classification and system evaluation. Each of these are discussed in the sections below. The sections are shown in Figure 2.

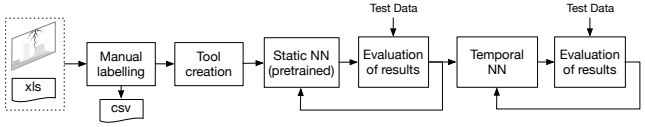


Fig. 2: Development workflow

4 Manual labelling

The training data provided included an excel spreadsheet that has a timestamp of each individual stoke as well as the duration of a strike in milliseconds. Since the frame-rate of the film is above 10 000, this information is not sufficient to determine the exact frame where the strike occurred, furthermore information is not available for the the leaders preceding a stroke. It is required to find the frame number of the initial leader occurrence and the frame of the stoke for each lightning event, which is to be done manually. The final frame can be determined using the duration time and the frame-rate as a buffer is included before and after strikes to provide blank training for the system.

5 Tool development

5.1 Video Conversion

The footage will be converted to a well defined/open format(video or image) which can be used as ground truth for the machine learning system. The converted footage should be of a short clip durations only containing the lightning events with some lightning-free surrounding frames (for the system to learn a baseline.) The proposed process is to use the `pycine` [4], `openEXR` [5] and openCV [6] python libraries to develop a script which converts the `CINE` video to a sequence of `EXR` images, while preserving the metadata as per-frame `csv` files. Since there will be a large number of images that contain useful information, a 16 bit single channel floating point image format (greyscale) can be used to reduce the image size.

The frames containing lightning events will be selected using the corrected frame/timing data produced. This can be calculated by selecting all the frames from the initial leader frame to the final lightning frame, (with a sufficient buffer), excluding the stroke frames. This reduces the dataset while providing more accurate data on the type of information the system should detect.

5.2 Image pre-processing

The `EXR` images produced by the tool in Section 5 would require further manipulation to allow for an accurate model to be developed. Ideally this should be produced using a python script as an extension of the tool in Section 5, but a method to achieved this has not yet been devised. It is therefore planned that the images be initialy manipulated using the non-commercial version of Foundry's `Nuke` compositing software on a per-video basis

6 Image processing

6.1 Edge detection

Edge detection algorithms find edges by comparing luminance discontinuities in surrounding pixels. This technique of image processing can produce an strongly contrasted image that outlines lightning and ground structures, as most storm footage occurs in an environment with dark backgrounds due to the moisture and small particular matter in the air normally present under storm conditions. A small selection of images with various edge detection techniques from some example footage is shown in Figure A.1. A threshold value can be chosen to help eliminate noise with edge detection.

6.2 Background removal

To isolate the lightning from the ground structures in the edge-enhanced frames, the rapid speed of lightning compared to static/slow moving object in the images (clouds, sky, buildings, etc). This is achieved by subtracting the information of an empty frame (that does not contain lightning) from the current frame (that contains lightning). The resulting image ideally only shows the leaders within each frame, and can be used as an image mask to specify the pixels of lightning. A bounding box can be semi-automatically produced which uses a threshold for the maximal luminance, and calculated the minimum region of interest [ROI] that contain the pixels of interest, as shown in Figure 4. This can also be achieved by using the auto-crop function within `Nuke`. This bounding box information should be written to per-frame text files, as four values defining the ROI, with the bottom left corner pixel of the image being [0,0].

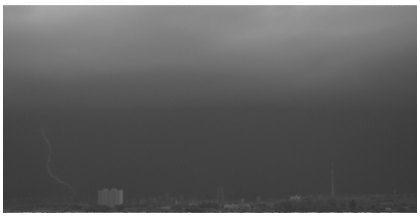


Fig. 3: Original video frame

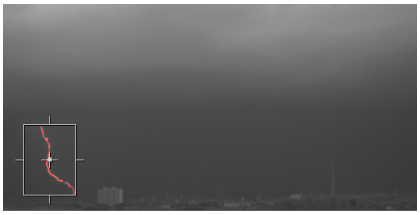


Fig. 4: Automatic ROI detection and image mask

6.3 Noise reduction

A blur or median filter can help reduce the random noise present on the frames, from atmospheric conditions or introduced by the camera sensor under low-light capture environment. The noise present in dark areas of frame can be reduced using an inverse log gamma grading of the image, which enhances the bright pixels of the frame (lightning) and reducing the dark areas luminance. Techniques exploiting the relative speed of motion of the lightning compared to the leader translation such as smoothing across the temporal domain, e.g. (`Nuke's` TimeEcho node implements this process). Additionally, the edges of the images exhibit 'edge artefacts', which can be removed using an image mask which excludes the edge luminance data. Temporal sampling to produce a noise-profile can be performed, which can be used to remove or reduce image noise,, but this could be computationally expensive.

7 Image classification

The topologies of modern deep learning networks can be very specific and requires expert domain knowledge. The use of pre-trained, performant networks can be used along with transfer learning to rapidly develop a network for a new set of classifications, with substantially reduced development and training time. Multiple individual networks may need to be compared or combined, or possibly including additional feedback mechanisms within the network (to compare surrounding and previous frames) to develop the classification of the direction of the leaders.

7.1 Static system

The pre-processed images with the `csv` label and bounding data will be used as an input to train an initial traditional CNN to detect the presence of lightning on a per-frame basis. The system should produce coordinates of a bounding box of the combined maximum ROI around the pixels containing the lightning along with the labelling of lightning frames.

7.2 Temporal classification system

The system will be extended to include temporal sampling techniques to the labels of frames that contain lightning. This could partially be accomplished by using temporal blending techniques to include elements of future and previous frame [7]. Performing an edge detection operation only in the `x` or `y` direction of an image (such as comparing the surrounding line pixels in that specific direction, on a preceding frame) can also be used to evaluate the gradient of motion of image elements [8]. Other techniques include the use of long short-term memory [LSTM] [9][10] or recursive neural networks [RNNs] [11].

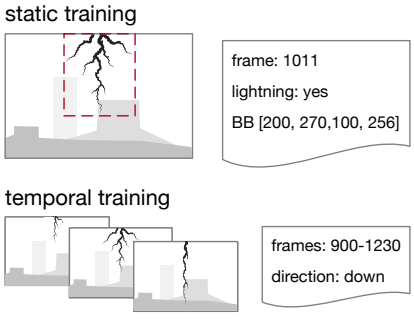


Fig. 5: Static and temporal image/label inputs

7.3 Network selection

The performance of leading networks, such as Inception ResnetV2, ImageNet, VGG, AlexNet and other publically available deep learning models already trained on large data sets, will be used as the basis of the system's network. Performance comparisons on the prepared data and labels should be done to determine an appropriate network topology.

7.4 Tensorflow

Google's TensorFlow framework for AI and machine learning offers a stable, widely used environment using high-level python libraries to develop deep learning applications. The support for GPU parallelism with wide support for local and cloud computing services makes the framework an appropriate choice for the project.

7.5 Training and validation

The input data should be segmented into training and test sets. A portion of the labelled data should be kept aside for evaluating the performance of the network, and never used for training to help ensure dependable metrics of accuracy. Portions of the training data can be used to validate the training process. This data can be interleaved on successive epochs of training to help prevent overfitting, and extend the available training data. A division scheme is shown in Figure 6.

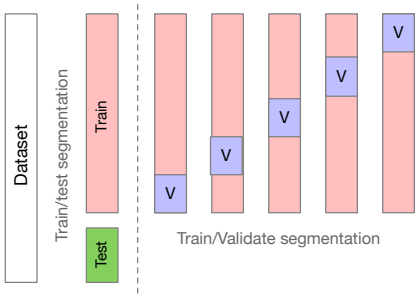


Fig. 6: Proposed 80/20 segmentation of labelled input data.

8 Validation and project success

The system will be provided with previously unseen (labelled) data in the form of the test dataset images, and the output compared to the known ground truth of the footage. The bounding box accuracy, the mask segmentation and the correctness of the classification will be compared to the actual ground truth, and the error established using the mean squared error [MSE]. The system should aim for the lowest achievable MSE within the project time and resource constraints, with an accuracy above 60% considered acceptable. Additionally, unlabelled data may be used as an input, but the validation of this performance may be difficult to establish numerically, without human visual comparison and analysis of the output features, and is therefore not included in the project base success criteria.

9 Conclusion

The project parameters, intentions and success criteria were discussed, along with a description of the planned methodology and an explanation of the planned image processing. The incomplete existing data will require some manual labelling, with tool creation using open source libraries to develop automatic mask and bounding box metadata. The basic network type for object classification from still images was chosen to be CNNs using transfer learning, but the nature of a data-driven process means further experimentation with network topologies and preexisting models will be required to achieve temporal classification of complete events. The task breakdown into critical sections for a workflow procedure were established.

REFERENCES

[1] P. Lalande, A. Bondiou-Clergerie, G. Bacchiega, and I. Gallimberti, "Observations and modeling of lightning leaders," *Comptes Rendus Physique*, vol. 3, no. 10, pp. 1375–1392, 2002.

[2] V. A. Rakov and M. A. Uman, *Lightning: physics and effects*. Cambridge University Press, 2003.

[3] V. Cooray and T. Institution of Engineering and, *Lightning Electromagnetics*, ser. IET Power and Energy Series. The Institution of Engineering and Technology, 2012, no. Vol. 62.

[4] B. Hagen, "pycine," 2018, last accessed 14 July 2019. [Online]. Available: https://github.com/OTTOMATIC-IO/pycine/blob/master

[5] Industrial Light and Magic, "Openexr," 2019, (Last Accessed on 07/15/2019). [Online]. Available: https://www.openexr.com/

[6] OpenCV team, "Opencv," 2019, (Accessed on 07/15/2019). [Online]. Available: https://opencv.org/

[7] K.-T. Lai, D. Liu, M.-S. Chen, and S.-F. Chang, "Recognizing complex events in videos by learning key static-dynamic evidences," in *European Conference on Computer Vision*. Springer, 2014, pp. 675–688.

[8] R. B. Gin, R. A. Bianchi, and B. H. Pilon, "A computer vision system to analyse lightning flashes images," in *Seventh Conference on Artificial Intelligence and its Applications to the Environmental Sciences*, 2009.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] C. Olah, "Understanding lstm networks," 2015, (Accessed on 07/15/2019). [Online]. Available: \url{http://colah.github.io/posts/2015-08-Understanding-LSTMs/}

[11] Y. Bengio, P. Simard, P. Frasconi *et al.*, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[12] J. Canny, "A computational approach to edge detection," in *Readings in computer vision*. Elsevier, 1987, pp. 184–203.

APPENDIX A

A.1 Risk Management

The risk register is tabulated in Table I. The probability and expected impact of the listed risks are presented with and action that is required if the risk materialises. The risks are not fixed but should be evaluated weekly. With a hard reevaluation deadline set as per the Gantt chart.

TABLE I: Tabled risk register for the project as a whole

Risk	Probability	Impact	Response	Actions
Delay in data handover	H	H	Accept/Mitigate	Reuse of transformed existing data, web resources
`pycine` library is unusable due to lack of maintenance and documentation.	L	H	Monitor	Manual exportation using Phantom Cine Control (PCC), additional hard drive space, compute power and crash the work-flow to allow for additional time requirement
Image processing is too computationally heavy	M	H	Monitor/Mitigate	Increase the cloud compute resource and/or reduce resolution and/or reduce the dataset size
Training is more computationally intensive than estimated	M	H	Mitigate	Use of sparse network topology techniques (Fast-CNN, Faster-RNN), Increase the cloud compute resource and/or reduce the dataset size
Personnel Crisis during parallel segments	L	H	Mitigate	Reduction of scope
Data loss	L	H	Avoid	Multiple secondary and primary backups, no write operations to the video data.

A.2 Assumptions

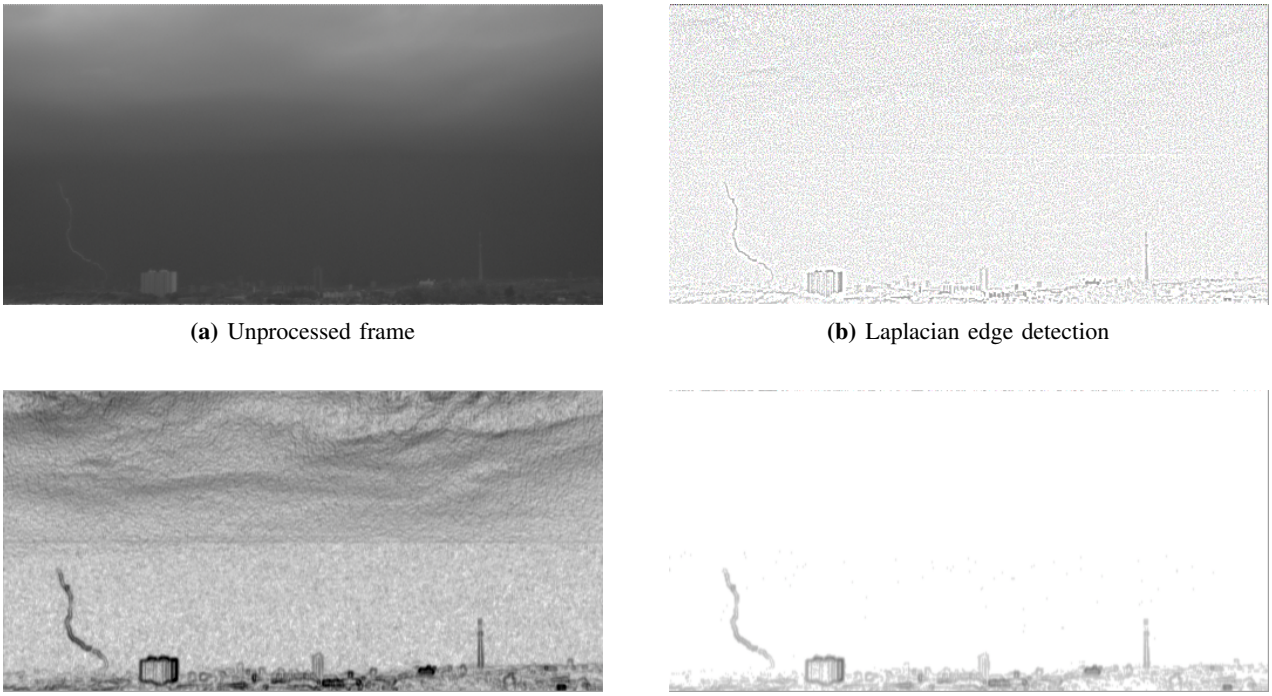
The assumptions for the project are listed in Table I with the reason for the assumption made and the value the assumption has to the project completion. Additional assumptions may be required to complete the project and the the assumptions made here could be modified to conform to changes made to the project.

TABLE II: Table of assumptions with reasons and the effect of the assumption on the design

Assumption	Reason	If True:
Sufficient data to produce a useful output	Large number of frames required for deep learning techniques	Deep learning techniques can be utilised
Sufficient time for pre-processing and training	Basic calculation on small sample data	Noise reduction in dataset, improved accuracy
Sufficient compute power to perform the operations	Basic calculation on small sample data	No requirement for cloud services
Sufficient transfer speed to transfer to cloud computing services	Projected requirement based on small sample	Can make use of cloud computing services
Single channel is sufficient to capture the information (Link to this earlier)	Predominantly monochromatic footage	Reduction in computing requirements
Static/slow moving background	Only high speed lightning data input	Background separation can use automatic techniques
Bright lightning on dark background	Examination of sample footage	Allows for luminance detection and keying techniques
Pre-trained networks with transfer learning are applicable	Consultation and research	Reduced complexity

A.3 Edge detection techniques

Edge detection filters can accentuate the lightning, but can also increase the background information (noise) present in a frame. An edge detect filter works by comparing one or more derivative operations across the delta of neighbouring pixels with a threshold, and often applying a smoothing operation [12]. There are two common techniques used in edge detection. A Laplacian allows for a more refined edge detection but is more susceptible to noise as this is effectively a double derivative [12]. This system preserves finer edge detail around the lightning but has greater background noise, and less contrast. Raising the threshold for edge detection can help improve isolating the noisy dark background components. The visual comparison of the techniques and an unprocessed frame are presented in Figure A.1



(a) Unprocessed frame

(b) Laplacian edge detection

(c) Sorbel edge detection

(d) Sorbel edge detection (raised threshold)

Fig. A.1: Comparison of edge detection

A.4 Task separation

The tasks required to be completed in the project are divided equally between the group members, where the strengths of each member is maximised. Sequential tasks that require discrete amounts of time on the critical path are assigned to both members, with a planed pair programming approach for the deep learning network implementation.

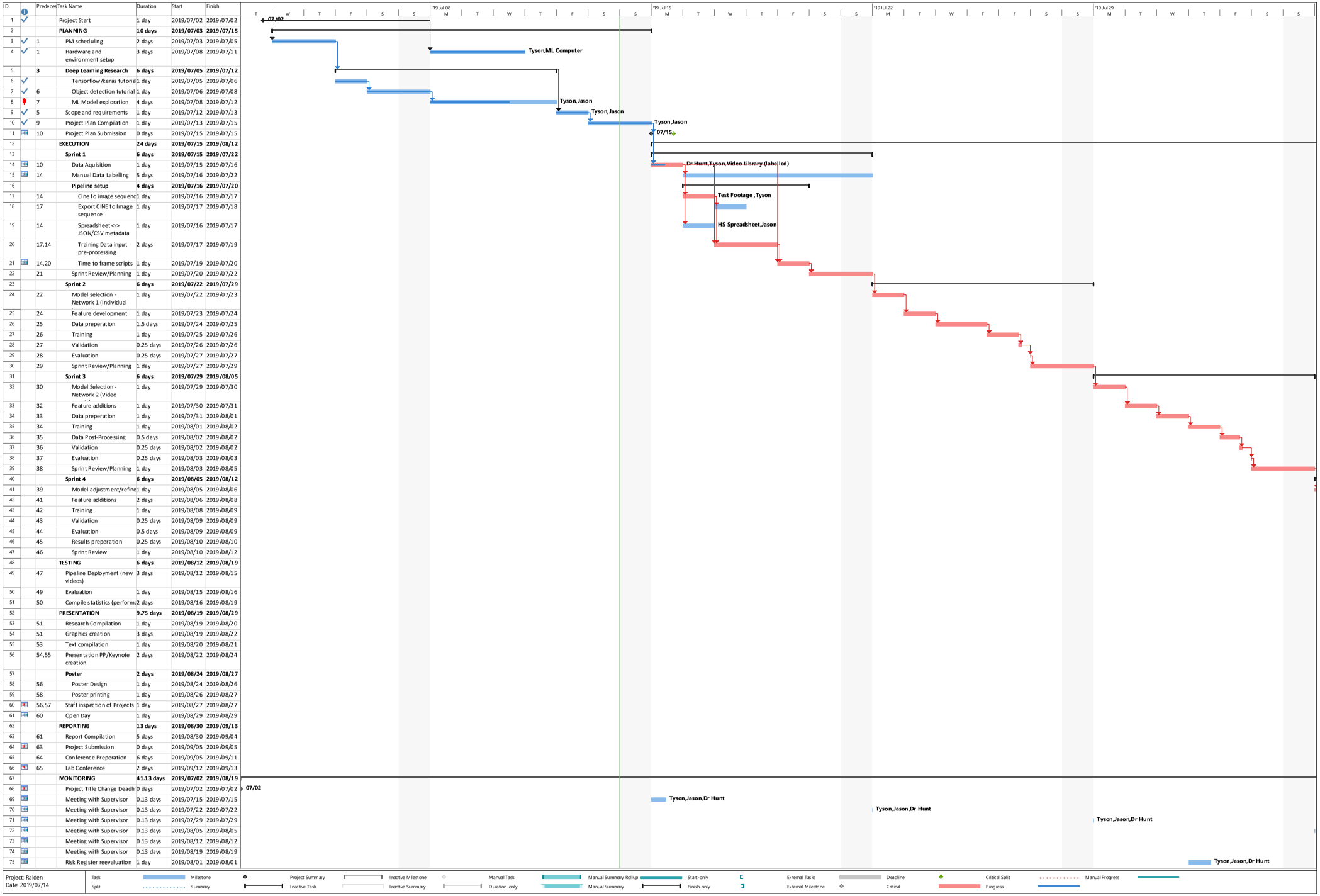
TABLE III: Suggested work assignment

Tyson	**Jason**
Image Pre-Processing (Nuke)	Cine to exr script
Image Pre-Processing (OpenCV)	Frame range to clips
Bounding Box	Spreadsheet to labels
Manual Labels	Manual Labels
Masking	Lightning Detection [ML]
Lightning Detection [ML]	Direction Detection [ML]
Direction Detection [ML]	Architecture decisions records
Meeting Minutes	Presentation content
Graphics for presentation	Accumulate statistics
Graphics for statistics	

APPENDIX B

B.1 Project scheduling

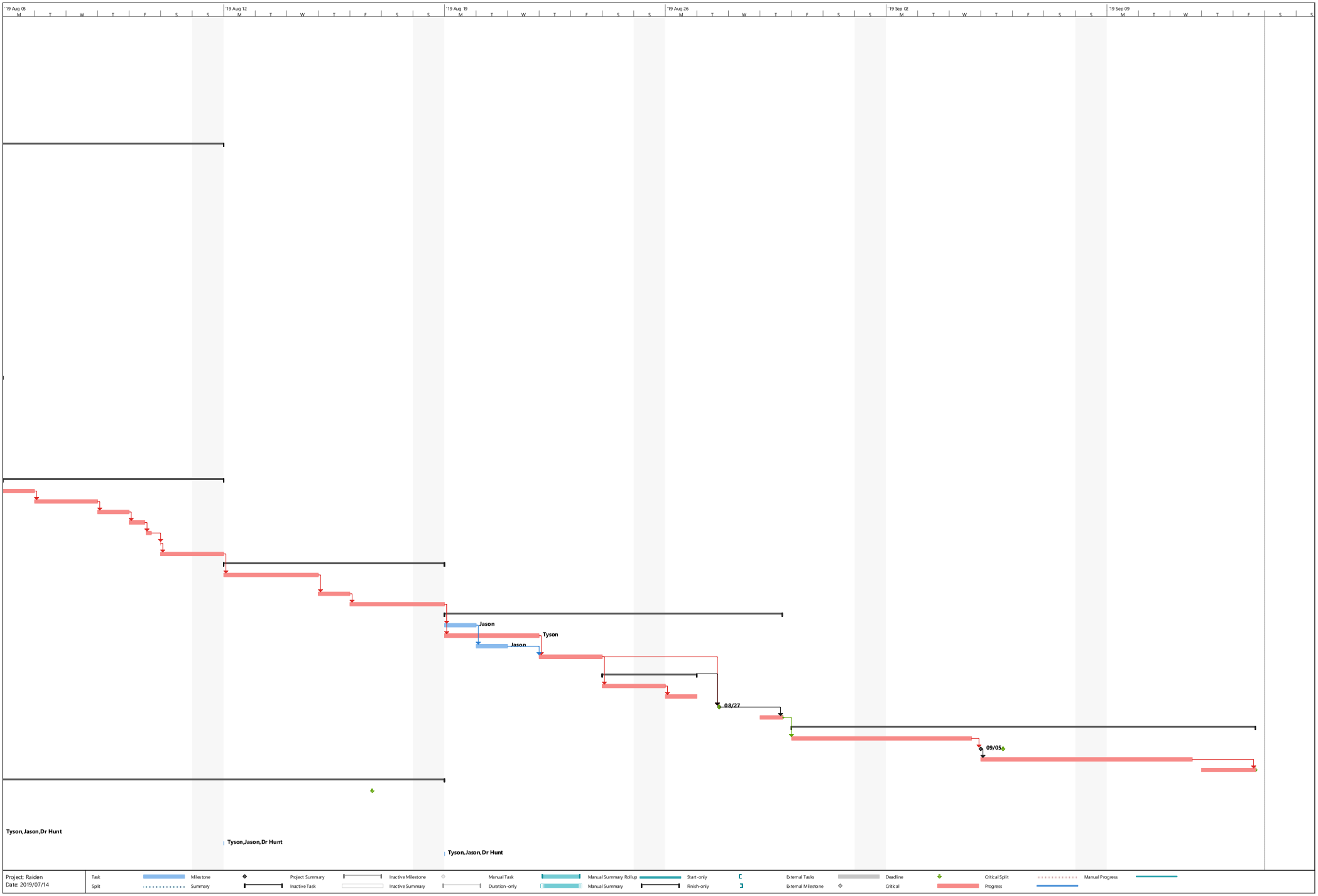
The time and task allocations for the project are shown in the Gantt charts presented below. Tasks on the critical path are shown in red, from project initialisation to the formal project start date of 15/07/2019, through the project presentation and report deadlines to the final conference.



ID	Predeces	Task Name	Duration	Start	Finish
1		Project Start	1 day	2019/07/02	2019/07/02
2		**PLANNING**	**10 days**	**2019/07/03**	**2019/07/15**
3	1	PM scheduling	2 days	2019/07/03	2019/07/05
4	1	Hardware and environment setup	3 days	2019/07/08	2019/07/11
5	**3**	**Deep Learning Research**	**6 days**	**2019/07/05**	**2019/07/12**
6		Tensorflow/keras tutorial	1 day	2019/07/05	2019/07/06
7	6	Object detection tutorial	1 day	2019/07/06	2019/07/08
8	7	ML Model exploration	4 days	2019/07/08	2019/07/12
9	5	Scope and requirements	1 day	2019/07/12	2019/07/13
10	9	Project Plan Compilation	1 day	2019/07/13	2019/07/15
11	10	Project Plan Submission	0 days	2019/07/15	2019/07/15
12		**EXECUTION**	**24 days**	**2019/07/15**	**2019/08/12**
13		**Sprint 1**	**6 days**	**2019/07/15**	**2019/07/22**
14	10	Data Aquisition	1 day	2019/07/15	2019/07/16
15	14	Manual Data Labelling	5 days	2019/07/16	2019/07/22
16		**Pipeline setup**	**4 days**	**2019/07/16**	**2019/07/20**
17	14	Cine to image sequenc	1 day	2019/07/16	2019/07/17
18	17	Export CINE to Image sequence	1 day	2019/07/17	2019/07/18
19	14	Spreadsheet <-> JSON/CSV metadata	1 day	2019/07/16	2019/07/17
20	17,14	Training Data input pre-processing	2 days	2019/07/17	2019/07/19
21	14,20	Time to frame scripts	1 day	2019/07/19	2019/07/20
22	21	Sprint Review/Planning	1 day	2019/07/20	2019/07/22
23		**Sprint 2**	**6 days**	**2019/07/22**	**2019/07/29**
24	22	Model selection - Network 1 (Individual	1 day	2019/07/22	2019/07/23
25	24	Feature development	1 day	2019/07/23	2019/07/24
26	25	Data preperation	1.5 days	2019/07/24	2019/07/25
27	26	Training	1 day	2019/07/25	2019/07/26
28	27	Validation	0.25 days	2019/07/26	2019/07/26
29	28	Evaluation	0.25 days	2019/07/27	2019/07/27
30	29	Sprint Review/Planning	1 day	2019/07/27	2019/07/29
31		**Sprint 3**	**6 days**	**2019/07/29**	**2019/08/05**
32	30	Model Selection - Network 2 (Video	1 day	2019/07/29	2019/07/30
33	32	Feature additions	1 day	2019/07/30	2019/07/31
34	33	Data preperation	1 day	2019/07/31	2019/08/01
35	34	Training	1 day	2019/08/01	2019/08/02
36	35	Data Post-Processing	0.5 days	2019/08/02	2019/08/02
37	36	Validation	0.25 days	2019/08/02	2019/08/02
38	37	Evaluation	0.25 days	2019/08/03	2019/08/03
39	38	Sprint Review/Planning	1 day	2019/08/03	2019/08/05
40		**Sprint 4**	**6 days**	**2019/08/05**	**2019/08/12**
41	39	Model adjustment/refine	1 day	2019/08/05	2019/08/06
42	41	Feature additions	2 days	2019/08/06	2019/08/08
43	42	Training	1 day	2019/08/08	2019/08/09
44	43	Validation	0.25 days	2019/08/09	2019/08/09
45	44	Evaluation	0.5 days	2019/08/09	2019/08/09
46	45	Results preperation	0.25 days	2019/08/10	2019/08/10
47	46	Sprint Review	1 day	2019/08/10	2019/08/12
48		**TESTING**	**6 days**	**2019/08/12**	**2019/08/19**
49	47	Pipeline Deployment (new videos)	3 days	2019/08/12	2019/08/15
50	49	Evaluation	1 day	2019/08/15	2019/08/16
51	50	Compile statistics (perform	2 days	2019/08/16	2019/08/19
52		**PRESENTATION**	**9.75 days**	**2019/08/19**	**2019/08/29**
53	51	Research Compilation	1 day	2019/08/19	2019/08/20
54	51	Graphics creation	3 days	2019/08/19	2019/08/22
55	53	Text compilation	1 day	2019/08/20	2019/08/21
56	54,55	Presentation PP/Keynote creation	2 days	2019/08/22	2019/08/24
57		**Poster**	**2 days**	**2019/08/24**	**2019/08/27**
58	56	Poster Design	1 day	2019/08/24	2019/08/26
59	58	Poster printing	1 day	2019/08/26	2019/08/27
60	56,57	Staff inspection of Projects	1 day	2019/08/27	2019/08/27
61	60	Open Day	1 day	2019/08/29	2019/08/29
62		**REPORTING**	**13 days**	**2019/08/30**	**2019/09/13**
63	61	Report Compilation	5 days	2019/08/30	2019/09/04
64	63	Project Submission	0 days	2019/09/05	2019/09/05
65	64	Conference Preperation	6 days	2019/09/05	2019/09/11
66	65	Lab Conference	2 days	2019/09/12	2019/09/13
67		**MONITORING**	**41.13 days**	**2019/07/02**	**2019/08/19**
68		Project Title Change Deadlir	0 days	2019/07/02	2019/07/02
69		Meeting with Supervisor	0.13 days	2019/07/15	2019/07/15
70		Meeting with Supervisor	0.13 days	2019/07/22	2019/07/22
71		Meeting with Supervisor	0.13 days	2019/07/29	2019/07/29
72		Meeting with Supervisor	0.13 days	2019/08/05	2019/08/05
73		Meeting with Supervisor	0.13 days	2019/08/12	2019/08/12
74		Meeting with Supervisor	0.13 days	2019/08/19	2019/08/19
75		Risk Register reevaluation	1 day	2019/08/01	2019/08/01

Project: Raiden
Date: 2019/07/14

Task | Milestone | Project Summary | Inactive Milestone | Manual Task | Manual Summary Rollup | Start-only | External Tasks | Deadline | Critical Split | Manual Progress
Split | Summary | Inactive Task | Inactive Summary | Duration-only | Manual Summary | Finish-only | External Milestone | Critical | Progress


'19 Aug 05
'19 Aug 12
'19 Aug 19
'19 Aug 26
'19 Sep 02
'19 Sep 09
Jason
Tyson
Jason
08/27
09/05
Tyson,Jason,Dr Hunt
Tyson,Jason,Dr Hunt
Tyson,Jason,Dr Hunt
Project: Raiden
Date: 2019/07/14
Task
Split
Milestone
Summary
Project Summary
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
External Tasks
External Milestone
Deadline
Critical
Critical Split
Progress
Manual Progress