Using Artificial Intelligence Techniques to Analyse and Evaluate Snooker Broadcast Videos

James Coulson: 111247132

2016-01-15

Computing and Information Systems

Self-Study

Preliminary Project Report

Acronyms & Abreviations

ANN: Artificial Neural Network
CNN: Convolutional Neural Network
CV: Computer Vision
GUI: Graphical User Interface
PoC: Proof of Concept
TDD: Test Driven Development
SRS: Software Requirements Specification

Introduction

The main objective of this is to explore various computer vision techniques for real-time analysis of snooker matches. The idea came from work – I work in an online bookmaking company and it was mentioned that “in-running” betting models could not currently be created for snooker as there was no live data available. While the idea stemmed from work there are no commitments regarding output for work, and no deadlines other than those set by the university, all work is done entirely in my own time and without external resources.

In-running betting is defined as taking bets on a sporting event while it is currently live, for example offering odds on who's going to score next in a football match, or who will win the next game in tennis. In order to offer these odds to customers, quantitative analysts build models which try to offer the most compelling initial prices while still making a profit for the company, which can then be tweaked by any traders in real time. In order to create these models a large amount of data is gathered and used to predict potential outputs and their likelihood. Once the models are created, a live stream of data is required to put the model to use. The aims of this project hope to cater to that need.

Relevant courses

Artificial Intelligence; Neural Networks; Software Engineering, Algorithm Design and Analysis; Databases;

Aims and Objectives

The main aim of the project is as follows:

- To use artificial intelligence techniques to analyse and evaluate snooker broadcast videos, and to extract usable data from them.

Objectives

- - Literary Survey: A review of the image recognition techniques that have been used successfully (or otherwise) and if and how they may be applied to real-time processing and tracking.
- Data Collection: Collect enough video data of live matches to analyse and use in the methods specified.
- Compare Methods: Build some proof of concept algorithms using the data collected and analyse the results, ultimately choosing the most suitable method to implement in the final program.
- Data Context: Develop an engine which configures the rules for snooker so as the data can be contextualized within the program.
- Front-end Development: Create a messaging API to ultimately allow models to subscribe to, as well as a GUI for anyone who needs to view the output data.
- Evaluation and Analysis: Write up an evaluation outlining the solution built.
- Conclusion and future work.

Methods

From early beginnings where a professor in MIT postulated that it would take a few PhD students no longer than a summer to crack resulting in the Summer Vision Project (Papert 1966), to today where hundreds of levels deep neural networks are currently being employed using clusters of GPU's for massive parallelism (He et al. 2015), processing power is finally reaching a stage whereby computer vision (CV) is becoming mainstream (S. Tarkov & V. Dubynin 2013), (Krizhevsky et al. 2012). This project will attempt to implement one of these techniques into a functional piece of software to allow information to be parsed from snooker matches.

The first stage necessary will be to collect data to train on. This will be in the form of good quality videos of snooker matches that have been broadcast and uploaded online. Ideally all videos will be of an HD 720p quality, although if time permits then using lower quality data will also be considered. A large amount of data will be necessary in order to train the models on, whether it be neural networks which take a large amount of training data (cite), or another feature detector such as SURF (Bay et al. 2008) to test against.

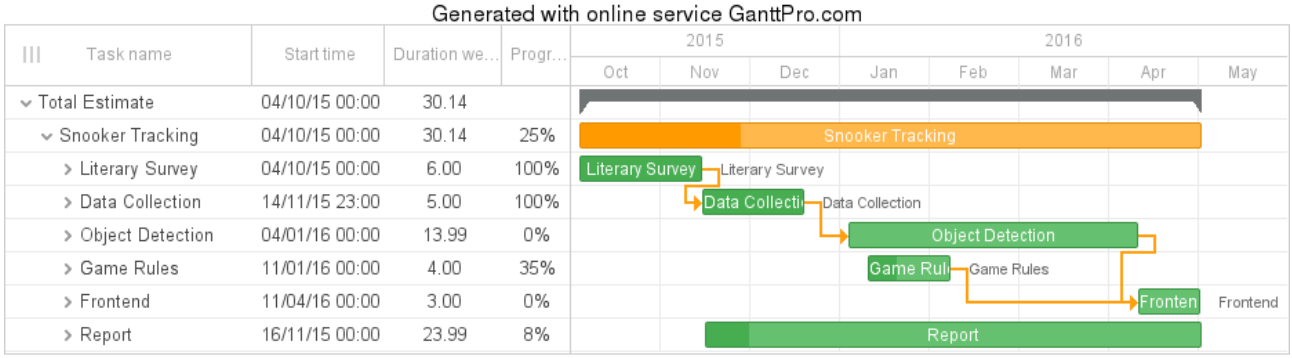
Once the data has been collected, then basic CV methods will be designed as a proof of concept and tested for suitability. The accuracy of the PoC's will need to be established using analytical techniques, and ultimately the best performing algorithm will be implemented into a working piece of software. In order to draw context and semantic information out of matches, a rules engine will also need to be developed allowing the detection software to establish information about the current state of the match. This will involve creating a database containing the rules, as well as an API to allow other software to extract information from it. A Software Requirements System (SRS) will be written prior to development in order to outline the required functionality of the algorithms, and then unit and integration tests will be written during development using a Test Driven Development (TDD) methodology (Beck & Andres 2004). If the method chosen is one which involves training, then an AWS cluster may be spun up, depending on the requirements of the training methods. This can be scripted using such tools as Anaconda Cluster (Anaconda n.d.).

The ultimate use of the application will come in the form of a messaging API which will allow other software to parse the information that the CV algorithm collates. It will be possible to plug into the information being gathered by the CV algorithm and evaluated within the rules engine, allowing the user to parse whatever data necessary to perform real-time analysis of the match. If time permits, then a GUI will be developed displaying the position of each snooker ball on the table. This is a secondary objective however, as the necessary functionality can be performed without any front-end.

Project Plan

A project plan was put together within GanttPro . A condensed version of the plan is shown here, with the full version has been attached in a separate PDF.

Generated with online service GanttPro.com



Progress To Date

Literary Survey

The first stage of the project was to perform a literary survey of CV and ANN's. To start with, a history of computer vision was reviewed, starting from 1963 where Roberts attempted to using CV techniques to detect 3D objects (Roberts 1963). Another attempt at tackling CV was made by the Summer Vision Project, which grossly underestimated the amount of work that was necessary (Papert 1966).

The initial use of Neural Networks for CV begun with Rosenblatt's seminal paper on Perceptrons (Rosenblatt 1958). This proved to be a false dawn for the advent of ANN's as field tests of the algorithm failed, mainly down to their lack of ability around learning non-linear problems AKA the XOR problem (Floyd 2013). Work on ANN's was pretty much halted after Paper and Minsky's book debunking the use of Perceptrons (Minsky & Papert 1969), with researchers spending more time concentrating on pixel-level detection such as edge detection (Canny 1986) and feature detection algorithms such as those using Hough Transforms (Ballard 1981).

ANN's started to pick up traction again during the 80's, with one of the first explorations of the deep learning architecture (Fukushima 1980). This was largely down to the re-discovery of the back-propagation algorithm (Rumelhart et al. 1988), which allowed ANN's to fit to non-linear output and converge to an optimal error state quickly. A paper by LeCun (LeCun et al. 1998) re-introduced a type of ANN called the convolutional neural network (CNN), which implemented the back-propagation algorithm and applied it to character recognition. The use of these CNN's when applied to deep learning techniques have revolutionised the image recognition to an almost human level (cite).

Findings in other CV techniques have been also been accelerating since the 90's such as SIFT (Lowe 1999) which uses feature detection to describe features, and SUSAN (Smith & Brady 1997) which uses processing techniques around detecting corners and images. As well as classifying images, tracking objects within video streams has also been researched, applying many of the image detection algorithms in real time. Methods such as particle and Kalman filters (Chen 2003) have been used, as well as balls in sporting events (Yan et al. 2014), (Yan et al. 2005). Some of these algorithms have even managed to be applied to extracting semantic information from sporting events (Rea 2005), similar to the end goal of this project.

Data Collection

In order to train and validate the CV techniques a lot of data needs to be collected. This was achieved by scouring YouTube for a number of video's and downloading them. The initial search was a very manual technique, with all the URL's needing to be manually checked and validated. Once the URL's were collected, they were entered into a PostgreSQL database, with the script added to version control for future usage

(assuming they're not taken down in the future). Once all the URL's were collected, the video's were downloaded using youtube-dl (Ricardo Garcia Gonzalez n.d.) and stored on disk for future use. Only video's longer than 20 minutes in HD were considered, which allowed plenty of full length matches to be collected. There are currently 20 videos containing over 60 hours of footage, which works out at over 150GB so far. This is still work in progress, and will be increased if necessary.

I've also created a script to sample random images from the videos in case they're necessary to train on. Currently this simply pulls images at random, although in future I hope to only pull images that are necessary, for example only those that are using the top-down view at the end of the table, should this be required. One of the risks with having such a large amount of data is actually being able to process it all. In case the computing power I can access at work or at home is not sufficient, some Amazon AWS clusters will be set up to try and take advantage of their parallel capabilities.

Plan Timelines

Currently I'm behind schedule with regards to my project plan, mainly due to work commitments before Christmas. I've only managed to perform the literary survey and data collection up until now, however I have not yet started development of options used to detect objects, despite taking into account considerations with regards to Christmas holidays. I'm currently about 2 weeks behind, however with more free time available soon it should be possible to catch up.

Planned Work

The next step of the project is to start researching a developing a few select options to compare. This will involve a Neural Network options as well as at least one other method for detecting images. Once the options have been finalised, a PoC with minimal functionality will be created for each option in order to test the results at a basic level. These results then need to be analysed and compared in order to find out which performs the best. There are a number of libraries which can help in this instance, for instance Theano (Bergstra et al. 2010) and OpenCV (Morozov n.d.).

Upon picking a solution to work with, a fully fledged application will be developed for use. This will start with a Software Requirements Specification (SRS) outlining the functionality of the system. This will be proceeded by development using the TDD approach (Beck & Andres 2004). One advantage to this approach is that test coverage is present throughout the application, meaning any changes made will be covered in case they break the codebase. The resulting software will then need to be trained on the data collected, and the results analysed. This whole development section will be more of a cyclical development cycle which isn't present in the Gantt chart plan. Tests such as processing time and accuracy are key too, as an inaccurate or slow program is of little use. This will be done after the functionality, as "Premature optimization is the root of all evil" (Knuth 1974).

The final step is to create the user interface, of which the main deliverable is the API parsing the data. This will be in the form of a messaging service, as the data will be parsed in real-time meaning that any software using the data will be getting snapshots of the current state of the match being parsed. A nice to have would be a GUI front-end outlining the table and the current placement of the balls on it.

Throughout this process the final report will be ongoing, and during each stage it will be updated with any additions or amendments, rather than saving it for when the development and analysis has been complete.

References

Anaconda, Anaconda Cluster. Available at: http://docs.continuum.io/anaconda-cluster/index.

Ballard, D.H., 1981. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2), pp.111–122. Available at: http://linkinghub.elsevier.com/retrieve/pii/0031320381900091.

Bay, H. et al., 2008. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3), pp.346–359. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1077314207001555.

Beck, K. & Andres, C., 2004. *Extreme Programming Explained: Embrace Change* 2nd ed., Addison-Wesley. Available at: http://www.amazon.com/Extreme-Programming-Explained-Embrace-Edition/dp/0321278658.

Bergstra, J. et al., 2010. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, (Scipy), pp.1–7. Available at: http://www-etud.iro.umontreal.ca/~wardefar/publications/theano_scipy2010.pdf.

Canny, J., 1986. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), pp.679–698. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4767851 [Accessed November 16, 2014].

Chen, Z., 2003. Bayesian Filtering: From Kalman Filters to Particle Filters, and Beyond. *Statistics*, 182(1), pp.1–69. Available at: https://scholar.google.com/citations?view_op=view_citation&hl=en&user=IKOGmeAAAAAJ&citation_for_view=IKOGmeAAAAAJ:roLk4NBRz8UC.

Floyd, T.L., 2013. Boolean Algebra and Logic Simplification. In *Digital Fundamentals*. Pearson, p. 241.

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), pp.193–202. Available at: http://link.springer.com/10.1007/BF00344251 [Accessed December 3, 2015].

He, K. et al., 2015. Deep Residual Learning for Image Recognition. Available at: http://arxiv.org/abs/1512.03385 [Accessed January 15, 2016].

Knuth, D.E., 1974. Structured Programming with go to Statements. *ACM Computing Surveys*, 6(4), pp.261–301. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.103.6084 [Accessed January 15, 2016].

Krizhevsky, A., Sutskever, I. & Hinton, G.E., 2012. ImageNet Classification with Deep

Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pp.1–9. Available at: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

LeCun, Y. et al., 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), pp.2278–2323.

Lowe, D.G., 1999. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, pp. 1150–1157 vol.2. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410 [Accessed July 9, 2014].

Minsky, M. & Papert, S., 1969. *Perceptrons: An Introduction to Computational Geometry*, Available at: http://www.amazon.co.uk/Perceptrons-Introduction-Computational-Marvin-Minsky/dp/0262130432/ref=sr_1_fkmr0_1?ie=UTF8&qid=1452081530&sr=8-1-fkmr0&keywords=Perceptrons+%28Minsky+and+Papert%2C+1969%29 [Accessed January 6, 2016].

Morozov, F., OpenCV. Available at: http://opencv.org/.

Papert, S., 1966. *The Summer Vision Project*. Massachusetts Institute of Technology. Available at: http://dspace.mit.edu/handle/1721.1/6125 [Accessed January 11, 2016].

Rea, N., 2005. *High-level Event Detection in Broadcast Sports Video*. Trinity College Dublin. Available at: http://www.mee.tcd.ie/~sigmedia/pmwiki/uploads/Main.Publications/rea04_phdthesis.pdf.

Ricardo Garcia Gonzalez, Youtube-dl. Available at: https://rg3.github.io/youtube-dl/.

Roberts, L.G. 1937-, 1963. *Machine perception of three-dimensional solids*. Massachusetts Institute of Technology. Available at: http://dspace.mit.edu/handle/1721.1/11589 [Accessed January 11, 2016].

Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in …. *Psychological Review*, 65(6), pp.386–408. Available at: http://psycnet.apa.org/journals/rev/65/6/386.pdf\npapers://c53d1644-cd41-40df-912d-ee195b4a4c2b/Paper/p15420 [Accessed January 6, 2016].

Rumelhart, D.E., Hinton, G.E. & Williams, R.J., 1988. Learning representations by back-propagating errors. In *Neurocomputing: foundations of research*. MIT Press, pp. 696–699. Available at: http://dl.acm.org/citation.cfm?id=65669.104451 [Accessed January 15, 2016].

S. Tarkov, M. & V. Dubynin, S., 2013. Real-Time Object Tracking by CUDA-accelerated Neural Network. *Journal of Computer Sciences and Applications*, 1(1), pp.1–4. Available at: http://pubs.sciepub.com/jcsa/1/1/1/index.html [Accessed January 9,

2016].

Smith, S.M. & Brady, J.M., 1997. SUSAN—A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1), pp.45–78. Available at: http://link.springer.com/article/10.1023/A%3A1007963824710 [Accessed January 12, 2016].

Yan, F., Christmas, W. & Kittler, J., 2005. A Tennis Ball Tracking Algorithm for Automatic Annotation of Tennis Match. *Procedings of the British Machine Vision Conference 2005*, pp.67.1–67.10. Available at: http://www.bmva.org/bmvc/2005/papers/paper-150.html.

Yan, F., Christmas, W. & Kittler, J., 2014. Ball Tracking for Tennis Video Annotation. In T. B. Moeslund, G. Thomas, & A. Hilton, eds. *Computer Vision in Sports*. Springer International, pp. 25–45. Available at: http://link.springer.com/10.1007/978-3-319-09396-3_2.