Australasian Transport Research Forum 2017 Proceedings
27 – 29 November 2017, Auckland, New Zealand
Publication website: http://www.atrf.info

Pedestrian Tracking Framework Utilising Computer Vision for Rapid Analysis of Public Spaces

Samuel R Hislop-Lynch[1], SangHyung Ahn[1] and Jiwon Kim[1]

[1]School of Civil Engineering, The University of Queensland, Queensland 4072 Australia

Email for correspondence: samuel.hisloplynch@uqconnect.edu.au

Abstract

The ability to record the trajectories and interactions of pedestrians in public places is necessary to understand, model and analyse the performance of built environments. However, few options are available to researchers to gather this information. Traditionally, simple point-counting techniques or video analysis performed with human sight have been relied upon to collect the required data, but these methods have limitations. Tracking the movements of pedestrians in public areas with point-counters can only reveal abstract flow patterns and lacks the potential to capture fine-grained detail. Human observation is useful for capturing the fine-grained detail of individual trajectories, but is rarely a tractable solution.

Recent advances in computer vision have allowed for automatic pedestrian tracking and interaction capture in open public spaces. Here, we present a framework, based on existing technology that can be used to build a pedestrian tracking and trajectory analysis application which solves the tractability issues associated with human visual analysis. Our framework is especially useful for capturing the movements of pedestrians in open public spaces such as public transport platforms, which will provide researchers with a finer level of detail than previously possible. It should be noted that this framework is aimed at researchers who wish to perform post-processing analysis of recorded video, rather than those who wish to capture the data in real time.

1. Introduction

There is an increasing demand for public transport as a growing percentage of the population chooses to live in urban centres. This presents a challenge for researchers and practitioners who are tasked with designing, building and upgrading the next generation of transport infrastructure. Faced with dwindling room for physical expansion, the favoured solution is more often to improve the utilisation and efficiency of existing spaces, rather than expand. In general, the efficiency of a space, and the merits of potential improvements can be investigated through computer modelling. However, building and calibrating a simulation model with sufficient granularity can require a great deal of information about the current state of the space.

In many cases, collecting enough data to calibrate a model with sufficient granularity, will be more time consuming than building the model itself. A statistically significant number of passengers must be profiled and many variables must be considered (walking speed, time spent lingering in spaces, time spent interacting with services, types of service interactions, etc.). Presently, collecting this type of data requires the intense involvement of practitioners and researchers, potentially dedicating weeks or months of their time. Until recently,

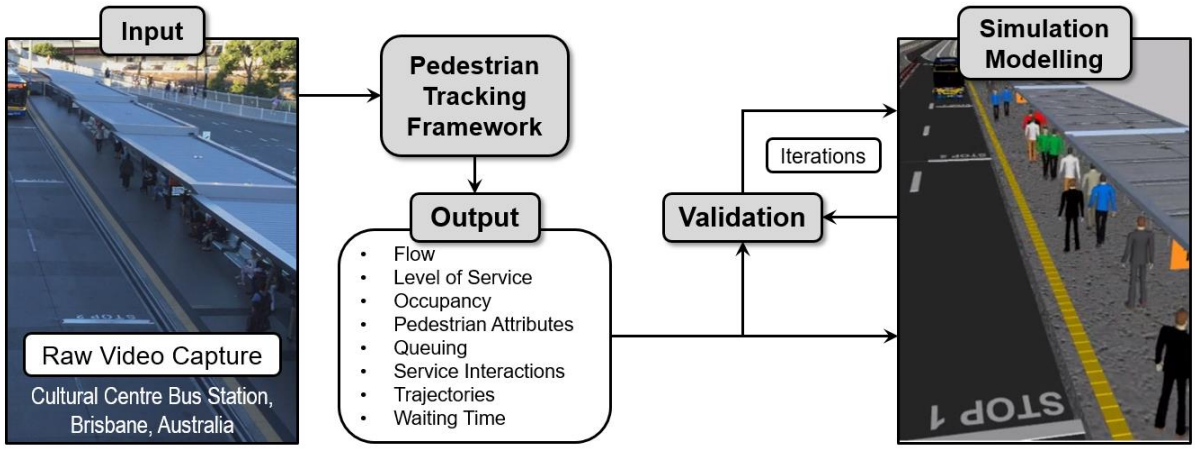
automating this task would have been prohibitively expensive and complex, so a hands-on approach to data collection would have been the only available option.

The automation of pedestrian data collection must be achieved to allow researchers and practitioners to investigate a problem and produce a solution in a realistic timeframe. With recent developments in the field of computer vision coupled with increasing computing power, this can be a reality. However, the fields of pedestrian modelling and computer vision have not yet completely merged. Many robust algorithms for detecting and tracking pedestrians exist, but they are yet to be unified under a single framework that transport researchers and practitioners can integrate into their data collection methods.

Here, we present a framework that seeks to bridge the gap between computer vision research and the field of pedestrian modelling (Figure 1). By taking stock of the current state of computer vision research and probing the suitability of available tools and algorithms, a workflow for detecting, tracking and observing pedestrian interactions has been developed. This has the potential to largely remove the requirement for human intervention in the data collection process and decrease the time required to obtain the necessary information. In essence, this framework is a solution to the tractability problems associated with pedestrian data collection.

In this paper, a case study is presented to showcase the proposed framework. However, this case study is limited to use a single camera to track the movements of pedestrians within an indoor study area. This framework is currently being tested within an airport environment focusing on check-in areas.

Figure 1 : An overview of the pedestrian tracking framework and how it can fit into the simulation pipeline.



2. Literature Review

The ability to observe and extract information at the individual pedestrian level, is central to modelling public spaces. Hoogendoorn and Daamen (2006) acknowledged the importance of individual pedestrian behaviours (specifically walking speeds and reaction times) and asserted that these characteristics should be included in pedestrian models to achieve realistic predictions. They also acknowledged that these parameters were difficult to observe directly. To collect the data necessary for their study, they recruited participants to wear coloured hats, which were associated with their pre-determined walking patterns. Data was then obtained by analysing recorded video based on colour segmentation. Although this type of setup is suitable for controlled experiments, for obvious reasons, this technology would not be suitable as a means of validating their conclusions in a real-world environment.

Predominantly human-based data collection is still a reality for many researchers, as evident in a number of recent pedestrian behaviour-modelling studies. Jaros et al. (2016), for example, presented a model for analysing buildings with high pedestrian flow. Their calibration data was obtained from a busy rail station, by an on-site person who collected a list of activities, followed pedestrians to obtain trajectories and counted flows through certain key areas. In an environment, such as a busy railway, where thousands of pedestrians can flow through in a few brief minutes, the task of gathering a substantial dataset is prolonged and laborious. Based on the comprehensive scope of Jaros's collection effort, it can be inferred that this was the case.

Computer vision as a tool for pedestrian data collection is not a new concept, previous research efforts have focussed on the automation of pedestrian data extraction from video. Tekonomo (2002), for instance, introduced a pedestrian detection and tracking system based on background subtraction. This novel system could be used to recover trajectory information from pre-recorded video, however, this system also lacked a means of strongly confirming human detections. The method of false detection rejection was based largely on background-subtraction-contour-size and trajectory-analysis, which can fail in certain circumstances. Objects of similar size to humans, which also happen to move in similar patterns, are frequently present alongside humans (trollies, large luggage, etc.) and without strong confirmation, could pass as human. The reliability of such a system could be improved by introducing a classifier (potentially based on convolutional neural networks) that would be able to directly qualify the detection as human or not.

3. Overview

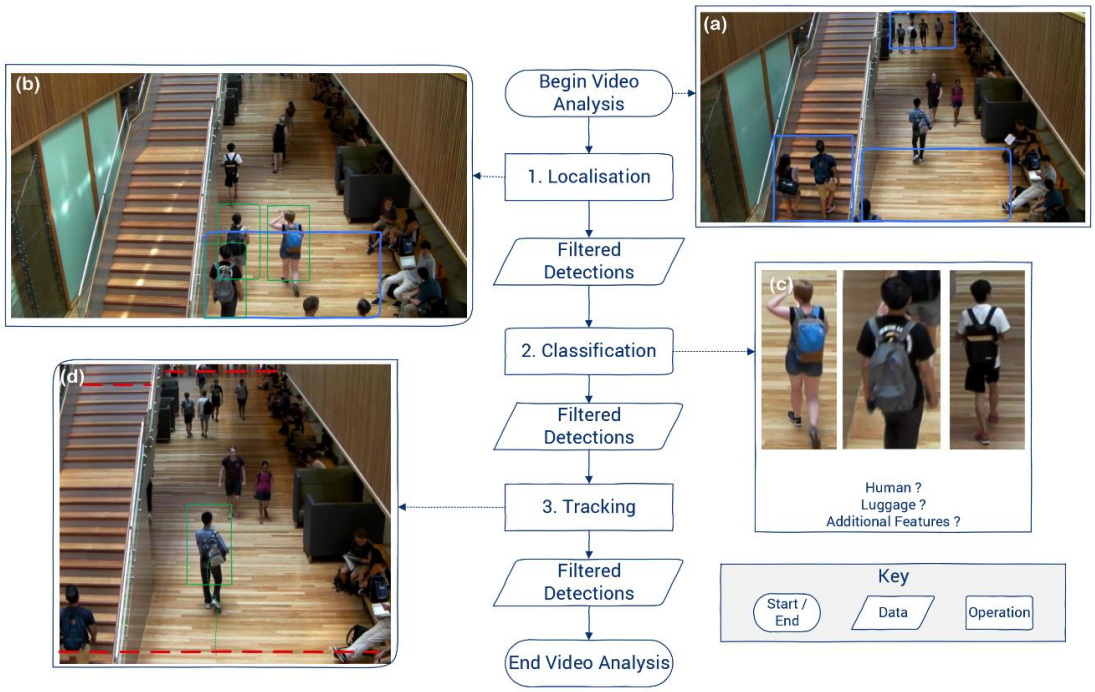
In response to a growing need for automated pedestrian trajectory analysis, a framework incorporating recent advances in computer vision and machine learning has been developed. This framework seeks to provide practitioners and researchers with the tools required to identify and extract trajectories, behaviours, interactions and basic pedestrian characteristics thereby improving the laborious task of manual data collection. In contrast to smart camera technology, a post-processing approach has been adopted. The researcher or practitioner would be required to collect a video data set, which would be later be analysed. This framework is intended for those who wish to investigate complex pedestrian environments, examine microscopic parameters and perform collection activities over short to medium periods of time.

The framework is structured around the hypothesis that all pedestrian events can be broken down into three abstract event categories: source events, sink events and interactions. *Source* and *sink* events are used to represent all possible entry and exit actions in the frame, whereas *interactions* broadly represent the engagements or activities undertaken by pedestrians. To this end, the goal of analysing video is to monitor source regions for pedestrians entering the frame, observe their interactions and terminate tracking upon exiting through a sink region. A broad overview of the proposed framework is illustrated in Figure 2. These three abstract event categories are identified and handled by the three main phases of the framework: localisation, classification and tracking.

The three phases: *localisation*, *classification* and *tracking* are abstract categories that represent a collection of image processing steps. In order to track and subsequently flag interactions, a pedestrian must first be identified. The localisation step is responsible for parsing through the entire video and capturing the first instance of pedestrians as they enter the video frame through source regions. Once the localisation phase has been completed, a list of initial pedestrian locations are passed onto the classification phase, which will ensure

that the detections are human and not false positives. Furthermore, accessories such as luggage, backpacks and even basic demographic information can also be obtained during classification. At the conclusion of the classification phase, a refined list of pedestrian locations will be passed on to the tracking phase. The tracking phase is responsible for estimating the movement of pedestrian between frames, identifying interactions with services and terminating tracking efforts as pedestrians exit the frame through sink regions. After all three phases have finished, a list of pedestrians complete with trajectory information, interactions with services and basic demographic inferences will be available.

Figure 2: Pedestrian Tracking Framework which consists of three phases: (b) localisation, (c) classification and (d) detection. Possible pedestrian entry points are defined prior to analysis (a).



4. Phase 1: Localisation

Trajectory analysis begins with examining the entry points of pedestrians into the video frame. Source regions are placed at all possible entry points and a direction of ingress is specified to distinguish between incoming and outgoing pedestrians. An example of this can be seen in Figure 3, a bounding box has been placed at an entry point into the frame. The direction of entry is indicated by the arrow within the box. Ideally, the source region bounding box should be wide enough to cover the entire path of ingress and tall enough to accommodate the largest pedestrian. Once all source regions have been defined, each will be analysed individually to obtain information about the points-of-origin of incoming pedestrians. This phase will be referred to as localisation from hereon and is visualised in Figure 4.

This framework distinguishes pedestrians from other elements in a source region by means of background subtraction. An empty frame, containing only the background elements, is subtracted from a potentially non-empty frame then greyscaled and thresholded. The result of this operation is a foreground mask, a binary image which differentiates between the foreground and background features. Although relatively simple, this approach has been successfully applied by a number of pedestrian tracking systems; some notable examples include Oliver, Rosario & Pentland (2000); McKenna et al. (2000); Cheung & Kamath (2004); and Kim (2008).

Figure 3: The blue bounding box represents a pedestrian source region, which registers incoming pedestrians travelling in the direction of the arrow and sorts them from outgoing pedestrians travelling in the opposite direction.

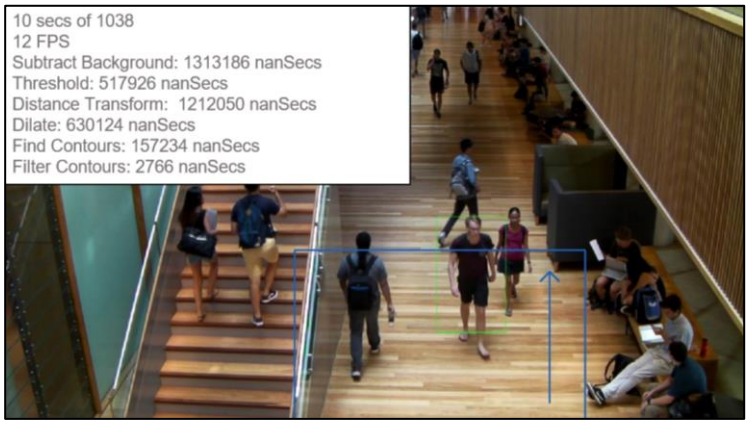
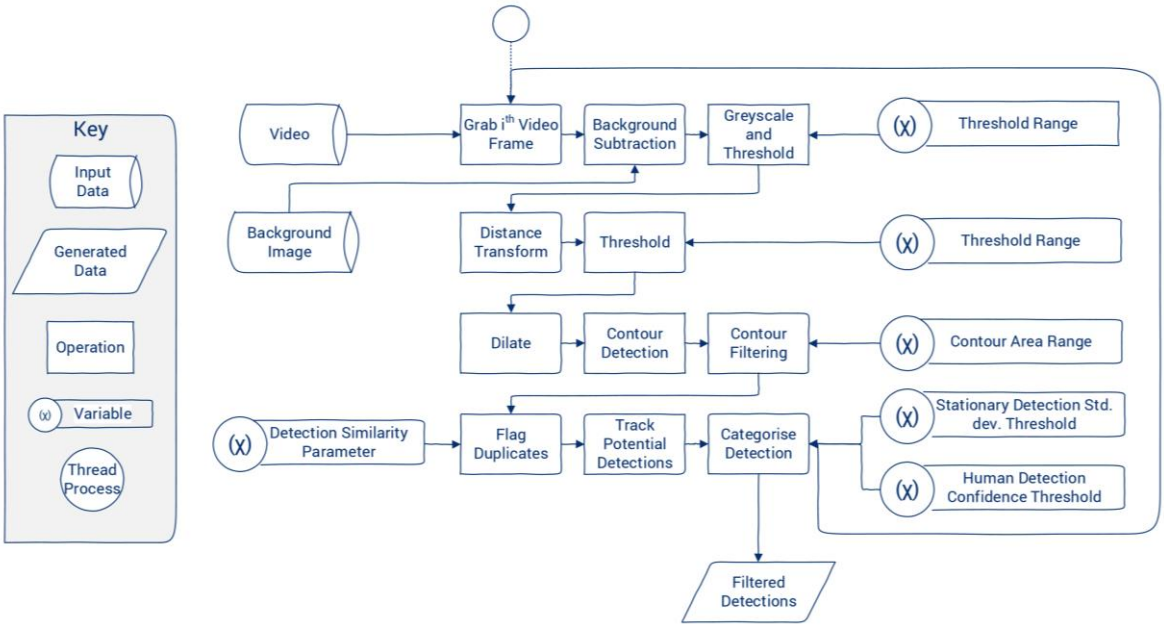


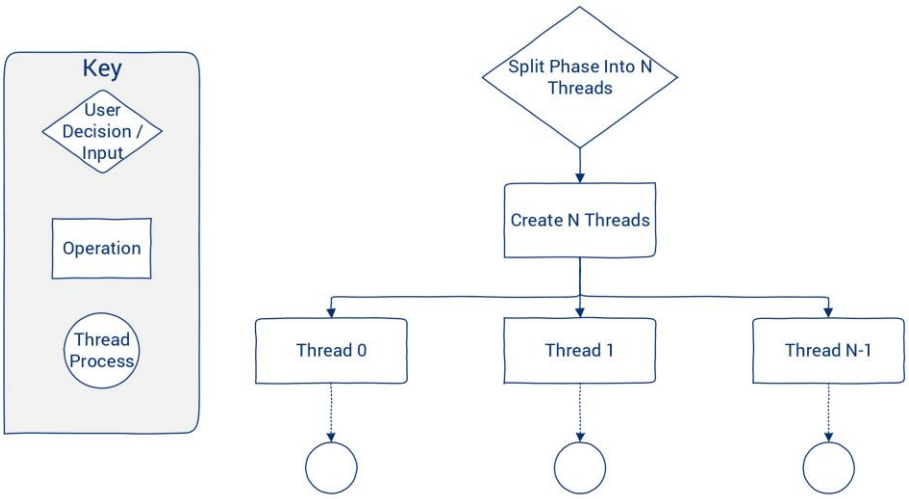
Figure 4: Localisation – Operation: a unique operation thread is created for each unique source region.



A thread of analysis is created for each source region, as depicted in Figure 5, which can be run concurrently or sequentially. Each thread will examine a single source region and attempt to estimate the locations of pedestrians as they enter the frame by performing background subtraction. Further rudimentary inferences about the nature of the detections, and if necessary, basic classification of the detections will also be carried out. Although there is no requirement to run these threads concurrently, doing so can potentially reduce the total analysis time.

Pedestrians will likely be present in the source region for many frames, so once a detection has been registered it must be tracked until it has left the source region. Doing so will reduce the likelihood of duplicating detections. At this point, it should be noted that the classification and tracking steps undertaken in the localisation phase are different and less comprehensive than those undertaken during the classification and tracking phases. The localisation phase is largely only concerned with obtaining the initial entry points of pedestrians and ensuring that the number of duplicate detections is kept to a minimum. The operations required to perform localisation are detailed in the following sections.

Figure 5 : Phase Parallelism - each of the three phases can be split into a number of threads to improve analysis time.



4.1. Background Subtraction

Salient features, or what are referred to here as “potential pedestrians”, are distinguished from the surrounding environment by means of background subtraction. In most cases this process is relatively straightforward. First, an empty background image with similar lighting conditions must be obtained. Then a pixel by pixel, absolute intensity-subtraction is carried out as described by Equation 1. If there are any potential pedestrians in the frame, then their positions will be markedly brighter than the surrounding background as shown in Figure 6(a).

$$pixel_i^{result} = \left| pixel_i^{background} - pixel_i^{current\ frame} \right| \quad for \quad 0 < i < N\ pixels$$

Equation 1: Background subtraction – result is found by performing a pixel-by-pixel absolute intensity subtraction.

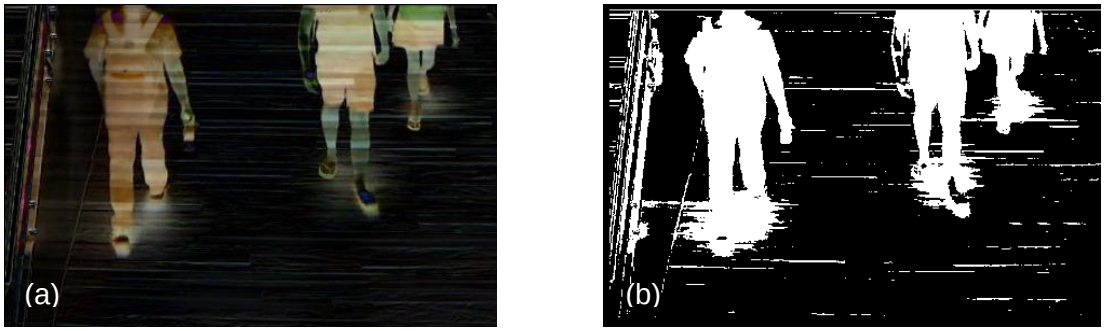
If the region is periodically unpopulated then the background image can be acquired relatively easily from the video data set. However, if some part of the region is always populated, then more sophisticated methods of acquiring the background image are available. A popular model for active background subtraction is presented by (Elgammal, Harwood & Davis 2000). This algorithm also compensates for small background perturbations, like rustling leaves and has shadow filtering capabilities to handle directional lighting artefacts. Additionally, a comprehensive review of background subtraction techniques has been carried out by (Piccardi 2004). Generally, however, most popular computer vision libraries will have optimised, pre-implemented versions of these algorithms.

4.2. Greyscale/Threshold

Once background subtraction has been performed, the frame will be greyscaled and thresholded. The goal of these operations is to produce a foreground mask, a binary map of regions that are different from the background. To produce this foreground mask, the image must first be greyscaled, which involves converting the red, green and blue intensity channels of the image into a single intensity channel. This will produce a noisy image, with most of the background filtered out, but which must be further refined. The noise can be reduced by thresholding which involves examining the intensity of each pixel. If the intensity lies outside a certain, pre-determined range, then the value is set to zero. Likewise, if the value lies within the given range, then it is increased to the highest intensity value.

Choosing the threshold lower and upper bound values requires human input. Raising the threshold lower bound value will generally decrease the noise captured along with any salient features. However, raising it too high will eventually lead to the exclusion of salient features. The same is true for the upper bound values. Therefore, the upper and lower bound values should be adjusted for each video and each source region to achieve the best possible results. An example of a greyscale/threshold operation is shown in Figure 6(b).

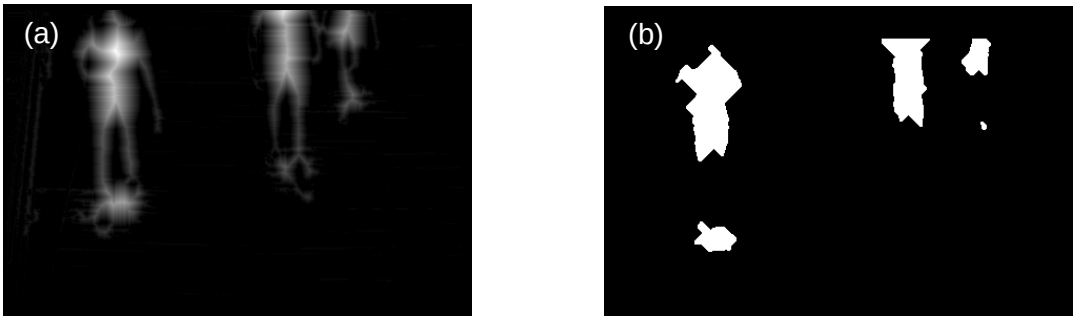
Figure 6: (a) Background subtraction – Localisation relies on distinguishing regions that deviate from an "empty" background image; (b) The result of the greyscale/threshold operation.



4.3. Distance Transform and Dilation

Following the greyscale/threshold step, a distance transform is performed on the region to further refine the location estimates. As shown in Figure 6(b), the two silhouettes on the right aren't completely separated, but are joined at the arm. This is a common occurrence and can present a problem when attempting to distinguish pedestrians in close proximity to one another. By performing a distance transformation, the silhouettes can be refined to a greyscale skeleton, with higher intensity values representing a closer proximity to the binary white centre of mass, as in Figure 7. Removing the lower grey values by thresholding and subsequently dilating the region will result in blobs which represent the location of pedestrians. A commonly cited algorithm for performing a distance transformation in linear time can be found in (Meijster, Roerdink & Hesselink 2002).

Figure 7: (a) Distance transformations help to separate objects in densely populated regions by reducing white salient areas to a skeleton, calculated based on the intensity "centre of mass". (b) Dilating the frame after performing a distance transform further helps to highlight potentially salient regions.

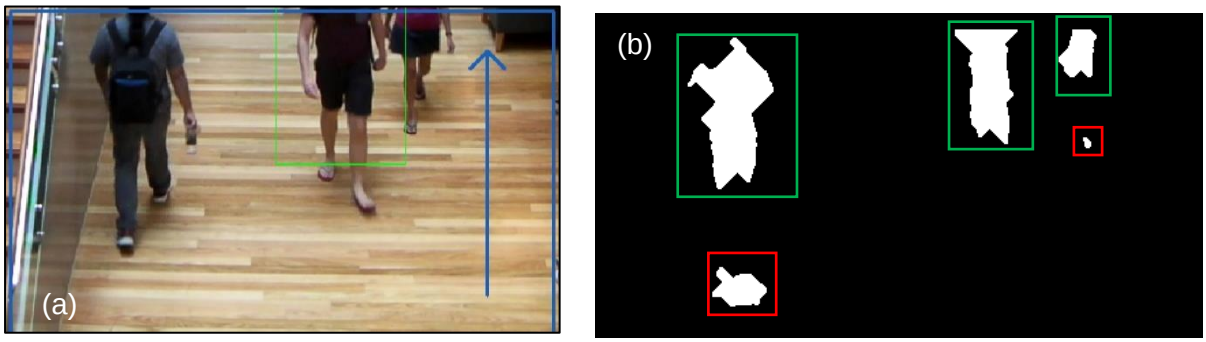


4.4. Contour Detection and Filtering

The next task, which will produce bounding-box estimates of salient feature locations, involves applying a contour detector to the dilated frame. Contour detection searches the frame for closed polygons after which, the approximate location and area of the salient features can be found as in Figure 8. Filtering the contours by area, retrieving only those which are similar in

size to the people in the video frame, will return a set of locations which can be used to initiate tracking. In the case that other objects, that are not part of the background and that fall in the same size range as humans, are also present in the frame, further classification can be carried out. A popular algorithm for performing contour detection, which is also implemented in the OpenCV library, is presented in Suzuki & Abe (1985).

Figure 8: (a) Contour detection aims to search the frame for closed polygons. (b) Once a list of closed polygons have been determined, they are filtered by area to reveal the approximate locations of potentially salient features.



Following contour detection and filtering, the centroid of each filtered polygon is calculated and a bounding box, which is used for tracking purposes, is centred on the polygon. Pedestrians will generally be of uniform size within the same source region, so a single bounding box size will be appropriate for all detections. Again, the size of the bounding box should be chosen at runtime by a trial and error process. At this stage, the origin of the detection will be known, but further work must be carried out to ensure that the detection is unique and not a duplicate.

4.5. Flagging Duplicates

Pedestrians will likely be present in source regions for many frames, therefore, a method of filtering pedestrians which have already been detected, is required. First, the location of the last ten to fifty detections, (depending on the size of the source region and the number of pedestrians flowing through the region), in the current frame is queried. The centroid of those previous detections is compared with the centroid of the current detection. If the centroids fall within a given tolerance, the current detection is flagged as a duplicate and no further trajectory analysis is performed. The tolerance for centroid similarity, labelled as *xDelta* and *yDelta* in the pseudo code in Figure 9, should be establish based on a trial and error process.

4.6. Partial Tracking and Categorisation

As previously explained, pedestrians will likely be present in source regions for many frames, which, without a means of handling, will result in multiple detections of the same pedestrian. Once a detection has been verified as unique, it is necessary to track that detection until it has left the source region to prevent future duplicates. Tracking is explained in detail in Section 6, however, a method for discriminating between detections within and outside the source region is required. The convention adopted here, is to consider detections with bounding boxes whose centroid lies within the source region, to be within the source region.

The ability to test the inclusion of a point within a polygon is critical to determining whether or not an object has left an area. It is also a reasonably straightforward process. A horizontal ray is drawn from any point outside the polygon and the number of times the ray intersects with the edges of the polygon is recorded. If the ray intersects with the polygon edges an odd number of times, then the point lies within the polygon. An even number of crossings would

mean that the point does not lie within the polygon. Although the methodology behind the test is intuitive,

Figure 9: Pseudo code for comparing the similarity of the current detection location to the location of a previously tracked detection.

```
Data: Current Detection X  Y Centroid Locations, Previous
      Detection X  Y Centroid Locations
Initialise Variables: xSimilarity = false, ySimilarity =
 false, duplicateFlag = false ;
1 if Current Detection Y Centroid > Previous Detection Y Centroid -
  yDelta AND Current Detection Y Centroid < Previous Detection
  Y Centroid + yDelta then
2 |  ySimilarity = true ;
  end
3 if Current Detection X Centroid > Previous Detection X Centroid -
  xDelta AND Current Detection X Centroid < Previous Detection
  X Centroid + xDelta then
4 |  xSimilarity = true ;
  end
5 if xSimilarity == true AND ySimilarity == true then
6 |  duplicateFlag = true;
  end
```

programming this test is a non-trivial problem. W. Randolph Franklin (1970) presented an algorithm for testing the inclusion of a point in a polygon, which is among the simplest solutions for this problem.

In addition to preventing duplicate detections within the same source region, partial tracking can also be used to infer whether a detection is entering or leaving the frame. Source regions should only register incoming detections as outgoing detections will likely have been registered at another source region; registering outgoing detections will inevitably lead to duplicates. The procedure detailed in Algorithm 1 is used to discriminate between incoming and outgoing detections.

Algorithm 1: A procedure for flagging outgoing detections

Define an arrow A which points toward the incoming direction

$$A$$

Determine the angle θ between the arrow A and existing Y axis

$$\theta = \cos^{-1}\left(\frac{A \cdot Y}{\|A\|\|Y\|}\right)$$

Remap the y components of the detection centroids to the rotated co-ordinate system

$$y'_i = x_i \sin(\theta) + y_i \cos(\theta) \quad \textbf{for} \quad 0 \leq i < \text{N tracked frames}$$

Sum the transformed y coordinate values for N tracked frames

$$\text{transformation sum} = \sum_{i=0}^{N-1} y'_i$$

If the sign of the transformation sum is negative, the detection is leaving the frame

$$\text{Detection is Leaving Frame} \quad \textbf{if} \quad \text{transformation sum} < 0$$

The possibility that false-positive detections will be registered within the source region must also be considered. False positives will remain stationary over a large number of frames and have the potential to result in the exclusion of true-positives as the area will be flagged as occupied. Therefore, it is necessary to flag and reject these as soon as possible. This is performed by checking the standard deviation of the X and Y coordinates for the detection centroids of the last 10 – 20 tracked frames and filtering out those which fall below a given threshold (Equation 2). The best value for the threshold will depend on the resolution of the video and how jittery the tracking results are. In practise, this value would best be found by a trial and error process.

$$\sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(x_i-\bar{x})^2} > \text{threshold} \wedge \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(y_i-\bar{y})^2} > threshold$$

Given

$$\text{N} = \text{Last N tracked positions}$$

$$\bar{x} = \text{the mean of centroid x positions}$$

$$x_i = \text{the } i^{th} \text{ detection x centroid}$$

$$\bar{y} = \text{the mean of centroid y positions}$$

$$x_y = \text{the } i^{th} \text{ detection y centroid}$$

Equation 2: Checking the standard deviation of detection centroids to determine whether a detection is stationary

If a detection has been flagged as stationary, then a basic classification step can be performed. This requires the use of a classifier, based on a convolutional neural network or an oriented histogram support vector machine, which is covered in further detail in Section 5. The contents of the detection bounding box is fed to the classifier which will return a probability between 0 and 1 that the bounding box contains a human. Depending on the result of the classifier, the detection can be accepted, in which case tracking would continue until the detection leaves the source region. Otherwise, the detection is flagged as a false positive and no further trajectory analysis would be carried out. By performing this analysis, many false positives can be caught before propagating further through the analysis.

5. Phase 2: Classification

It is often necessary to confirm that a detection is human, especially in environments where other human-sized moving objects are present. Additionally, other information such as whether pedestrians are carrying luggage or backpacks may be useful. The second phase of the framework involves classifying the detections obtained in the first phase. Two popular methods of visually classifying detections, Histograms of Oriented Gradients (HOG) and Convolutional Neural Networks (CNN), are discussed. However, as argued, CNNs offer more robust classifications and can be run reasonably quickly on mid-level, consumer-grade computers. In this context CNN classification is therefore, favoured over HOG based classification.

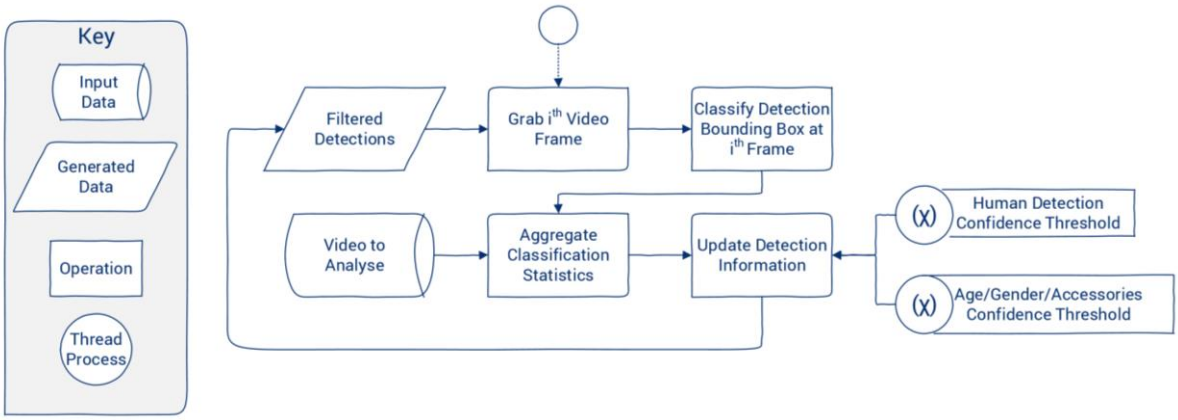
Immediately after the first phase, each potential detection will have a number of tracked positions, located within the source region. The aim of the classification phase is firstly, to ensure that the potential detection is human by examining each of these tracked positions. A CNN trained to detect humans will return a probability between 0 and 1 that the contents of each tracking bounding box contains a human. As occlusion and frame-sharing with other pedestrians may potentially occur, these probability values must be accumulated and

averaged for all frames that the pedestrian has been tracked in. Depending on the requirements of the analysis, further detection classification can also be carried out.

The secondary aim of classification is to infer information about the demographic that the passenger might belong to as well as to determine whether they are carrying luggage or other items. Depending on the quality and view angle of the video source, pedestrian age and gender may be deduced automatically with CNNs. Levi and Hassner (2015), were able to demonstrate that both age and gender could be inferred from face-on images of pedestrians. Such information is especially relevant when observing pedestrian interactions with certain service points, for example, computer-based automatic kiosks. Luggage and other items, which may also influence how pedestrians interact with service points, may also be detected with CNNs.

The classification phase, as depicted in Figure 10, begins by creating N classification threads for N detections obtained during the localisation phase. This is similar to the localisation phase, where N localisation threads are created for N source regions (Figure 5). For each frame that the n^{th} detection appears in, a cropped window based on the tracking bounding box will be extracted. Each cropped window will then be classified by a human-detecting CNN and any other classifying CNNs that the user wishers to run. Once every cropped window has been classified, the probabilities can be aggregated and inferences about the detection can be made. The opportunity to accept or reject the detection as well as any demographic or accessory inferences, based on the aggregated probabilities, arises. As every CNN classifier is different, the rejection threshold should be chosen based on a trial and error approach.

Figure 10: Classification – Operation: detection classification is performed on each detection for as many frames as the detection has been registered in.



5.1. Histograms of Oriented Gradients as Detection Classifiers

Histograms of Oriented Gradients (HOG) coupled with support vector machines were first introduced by Dalal and Triggs (2005) as a means of classifying humans in images. This method relies on examining the orientation of edges within an image and comparing the aggregated orientation values with a support vector machine. Unfortunately however, this method is extremely prone to the Pareidolia phenomenon (similar to humans perceiving objects in clouds) – largely due to the fact that it only employs a single method of classification. Vondrick et al. (2013) explored the tendency of HOG to produce false classifications with a tool that could reverse engineer the image once it had been broken down into a series of histograms. “HOGgles”, as the tool was dubbed, demonstrated that HOG classification could not solely be relied upon to classify a detection. More sophisticated methods of classification, such as convolutional neural networks, should instead be implemented. This method is

mentioned here to acknowledge the contributions of Dalal and Triggs, but also to steer researchers toward more robust, modern classifiers.

5.2. Convolutional Neural Networks as Detection Classifiers

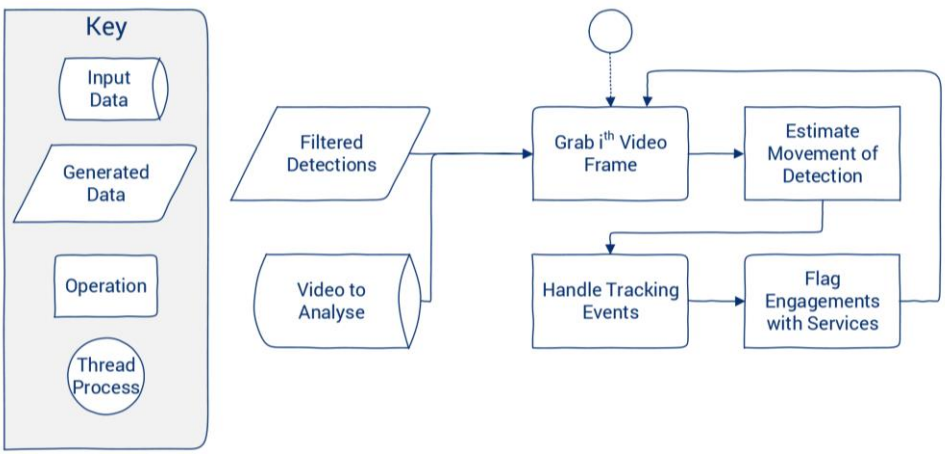
In recent years, Convolutional Neural Networks (CNN) have replaced many other image classifiers. This is likely due to the success that CNNs have demonstrated against the ImageNet Challenge (Russakovsky et al. 2015). Instead of examining and attempting to classify the contents of an image based on a narrow feature set, as is the case with HOG, CNNs examine a broad range of features. At the lowest level, CNNs recognise gradients and textures, later levels begin to recognise complex components such as arms, legs and faces and higher levels begin to recognise the connection between these complex components. Although implementing a CNN from the ground up can allow researchers and practitioners to develop tools specific to their own areas, this is a costly exercise in terms of personal learning and training time.

In some cases, it is sufficient enough to retrain only the final layers of an existing CNN. Generally, the lower levels of most CNNs will consist of similar gradient and texture patterns, only the later layers contain information which can discern complex objects like people. This approach has the benefit of reducing the training time and removing the need for high-performance computers equipped with general-purpose graphics processing units. Additionally, many aspects of CNN operation may be overlooked as the developer should only deal with the abstract concept. An example of a re-trainable CNN, the Inception V3 network, can be readily repurposed to classify pedestrians and other objects in video (Szegedy et al. 2016). Instructions for retraining the Inception V3 network can be found on the TensorFlow website (‘Image Recognition’ 2017).

6. Phase 3: Tracking

The final phase involves continuing the tracking effort started in the first phase. Moreover, the tracking phase is also responsible for flagging interactions between pedestrians and services as well as handling tracking events that could lead to termination. Similar to the first and second phases, the tracking phase requires that N threads be created for the N detections that are received after classification, as is shown in Figure 5. Each of N detections are then tracked until the pedestrian leaves the area via a pedestrian sink or the tracking effort is lost. Figure 11 details the operations carried out on each detection.

Figure 11: Tracking – Operation: the movement of each detection is estimated between frames, any tracking events are handled and engagement with services are flagged.



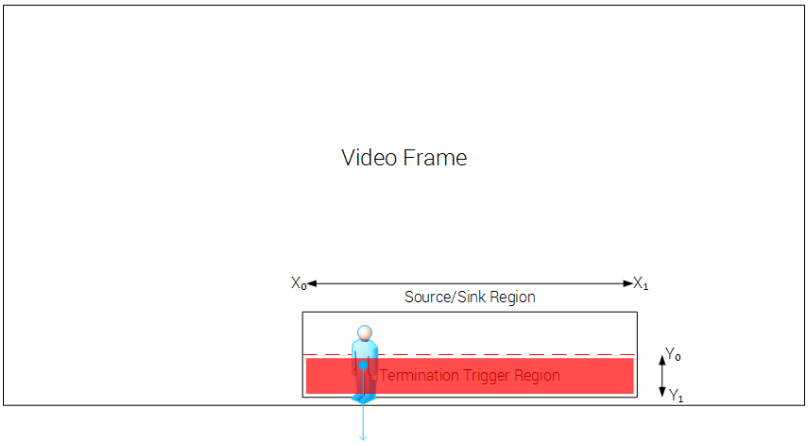
6.1. Movement Estimation

At the core, videos are simply a collection of frames played at high speed, resulting in the illusion of smooth movement. To this end, tracking algorithms attempt to isolate objects from the image background and estimate the motion between frames. A tracking algorithm known as Kernelized Correlation Filters (KCF), introduced by Henriques et al. (2015), is regarded as one of most robust and rapid solutions currently available. KCF has been demonstrated by the authors to track features at a mean frame rate of 154 frames per second (FPS) on raw pixels. This compares favourably with the previous generation of trackers, Tracking Learning Detection (TLD) introduced by Kalal et al. (2012) and Multiple Instance Learning (MIL) introduced by Babenko et al. (2009) which perform at 28 and 38 means FPS respectively.

6.2. Tracking Termination Events

In a video of a crowded public space, pedestrians will be entering and leaving the frame frequently. Where the localisation phase was responsible for initiating the tracking instance, a means of terminating the tracking effort must be introduced in this phase. Generally, tracking algorithms will estimate motion between frames but will not be able to flag an object has been lost outright. To handle this, the source region is halved along the closest, out-of-frame axis and the sub-region is continually monitored for tracking window centroids. In the case that the centroid of a tracking window enters this region, the tracking effort is terminated. Additionally, it is possible for a tracking target to lose the tracking window whilst fully within a trackable region. Generally, when this occurs, the tracking window will remain stationary for a long period of time. Many lost detections can be caught and handled by flagging the detection as idle with the method described in Equation 2, and then using a classifier to determine if a human is still within the tracking window.

Figure 12: The outer half of a source/sink region is monitored for leaving pedestrians. If the centroid of a tracking window crosses into this region, the tracking effort will cease.



The final type of termination event is less straightforward to handle. Occasionally, other objects may steal the tracking window away from the original object of focus, resulting in a sharp change of trajectory. There are a number of reasons that this might occur, for example, if the pitch angle of the camera isn't sufficiently downward facing, pedestrians will be occluded as they weave in and out of each other. When this happens, it is very easy for another pedestrian to acquire the tracking window of another. However, there may also be cases where a tracked pedestrian will genuinely need to change direction sharply. To this end, as there is no means of determining if the tracked object after a sharp trajectory change is still the same object when tracking began, the effort is not terminated. Instead, the tracking effort is flagged for the presence of a sharp change, which can later be examined by a human.

6.3. Flagging Service Engagements

In addition to determining the trajectories of pedestrians, researchers and practitioners are also interested in flagging the engagements of pedestrians with various services. Here, two types of engagements are considered, those that are region based and those that are proximity based. Region based engagements are flagged immediately when the centroid of a pedestrian detection window enters a polygon delineated area. The PNPOLY algorithm, discussed in Section 4.6, can be used to detect region based events. On the other hand, proximity based engagements are better suited to flagging interactions with a service point, as opposed to an area. Here, both a radial distance and a temporal threshold are used to flag proximity based engagements. Both of these threshold parameters are generally required as pedestrians may pass closely to a service point, but may not spend enough time near that point for the event to be considered an engagement. Region and proximity based engagements are primarily used to extract information about the duration of time that pedestrians spend at a service point.

7. Sources of Error and Verification

It is important to be aware of the current limitations of computer vision and the potential sources of error. Although this framework can be used to significantly reduce the time required to obtain trajectory and interaction data, some errors will likely be present in the final data despite the attempts to flag and remove during analysis. Some false detections may pass through the localisation and classification stages, while tracking efforts may fail or be subverted by other objects which can lead to erroneously flagged engagements. To ensure that a data set is largely free of errors, each detection and tracking effort should be reviewed by a human. Although this may seem to undermine the point of automation, tracking efforts can be viewed at high speed and require only minimal human intervention, allowing a large number of tracking efforts to be reviewed quickly.

8. Conclusions and Future Work

This paper introduces a framework for building a pedestrian tracking application that can be used by researchers and practitioners to greatly reduce the time and effort spent on manual data collection. The framework incorporates recent, robust computer vision technology that has the ability detect, categorise, track and observe the interactions of pedestrians in pre-recorded video. Due to the limitations of the technology, some false positive detections and erroneous tracking efforts may still be present in the final data. However, this framework has a number of incorporated strategies to catch errors early and filter these out before concluding the analysis. Our framework has been developed for practitioners and researchers who wish to perform post-processing of video data and would be unsuitable for real-time “smart camera” purposes.

Following on from this iteration, the next generation of pedestrian video analysis frameworks should incorporate real-time processing technology and fit into the smart camera landscape. Such a framework would go beyond just the implementation of CV algorithms. Suitable and cost-effective processing options as well as a way of continuing tracking efforts across multiple cameras should also be incorporated into this technology. A real-time pedestrian tracking solution may even be able to feed directly into real-time simulation which has the ability to forecast demand and help staff mitigate delays and drops in service quality.

9. References

Babenko, B, Yang, MH & Belongie, S 2009, 'Visual tracking with online Multiple Instance Learning', in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990.

Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz & Antonio Torralba 2013, 'HOGgles: Visualizing Object Detection Features', in IEEE, Sydney, Australia, pp. 1–8.

Cheung, SS & Kamath, C 2004, 'Robust techniques for background subtraction in urban traffic video', in pp. 881–892, accessed from <http://dx.doi.org/10.1117/12.526886>.

Dalal, N & Triggs, B 2005, 'Histograms of oriented gradients for human detection', in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, pp. 886–893 vol. 1.

Elgammal, A, Harwood, D & Davis, L 2000, 'Non-parametric Model for Background Subtraction', in *Computer Vision — ECCV 2000*, Springer, Berlin, Heidelberg, pp. 751–767, accessed June 9, 2017, from <https://link.springer.com/chapter/10.1007/3-540-45053-X_48>.

Henriques, JF, Caseiro, R, Martins, P & Batista, J 2015, 'High-Speed Tracking with Kernelized Correlation Filters', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596.

Hoogendoorn, S & Daamen, W 2006, 'Microscopic Parameter Identification of Pedestrian Models and Implications for Pedestrian Flow Modeling', *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1982, pp. 57–64.

'Image Recognition' 2017, *TensorFlow*, accessed June 14, 2017, from <https://www.tensorflow.org/tutorials/image_recognition>.

Jaros, M, Angelo, MD & Ferschin, P 2016, 'Modeling and simulation of pedestrian behaviour: As planning support for building design', in *2016 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*, pp. 1–8.

Kalal, Z, Mikolajczyk, K & Matas, J 2012, 'Tracking-Learning-Detection', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1409–1422.

Kim, Z 2008, 'Real time object tracking based on dynamic feature grouping with background subtraction', in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Levi, G & Hassner, T 2015, 'Age and Gender Classification Using Convolutional Neural Networks', in pp. 34–42, accessed June 15, 2017, from <http://www.cv-foundation.org/openaccess/content_cvpr_workshops_2015/W08/html/Levi_Age_and_Gender_2015_CVPR_paper.html>.

McKenna, SJ, Jabri, S, Duric, Z, Rosenfeld, A & Wechsler, H 2000, 'Tracking Groups of People', *Computer Vision and Image Understanding*, vol. 80, no. 1, pp. 42–56.

Meijster, A, Roerdink, JBTM & Hesselink, WH 2002, 'A General Algorithm for Computing Distance Transforms in Linear Time', in J Goutsias, L Vincent, & DS Bloomberg (eds), *Mathematical Morphology and its Applications to Image and Signal Processing*,

Computational Imaging and Vision, Springer US, pp. 331–340, accessed June 12, 2017, from <http://link.springer.com/chapter/10.1007/0-306-47025-X_36>.

Oliver, NM, Rosario, B & Pentland, AP 2000, 'A Bayesian computer vision system for modeling human interactions', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843.

Piccardi, M 2004, 'Background subtraction techniques: a review', in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, pp. 3099–3104 vol.4.

Russakovsky, O, Deng, J, Su, H, Krause, J, Satheesh, S, Ma, S, Huang, Z, Karpathy, A, Khosla, A, Bernstein, M, Berg, AC & Fei-Fei, L 2015, 'ImageNet Large Scale Visual Recognition Challenge', *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252.

Suzuki, S & Abe, K 1985, 'Topological structural analysis of digitized binary images by border following', *Computer Vision, Graphics, and Image Processing*, vol. 30, no. 1, pp. 32–46.

Szegedy, C, Vanhoucke, V, Ioffe, S, Shlens, J & Wojna, Z 2016, 'Rethinking the Inception Architecture for Computer Vision', in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, accessed from <http://arxiv.org/abs/1512.00567>.

Teknomo, K 2002, 'Microscopic Pedestrian Flow Characteristics: Development of an Image Processing Data Collection and Simulation Model', accessed from <http://arxiv.org/abs/1610.00029>.

W. Randolph Franklin 1970, 'PNPOLY - Point Inclusion in Polygon Test', accessed June 14, 2017, from <https://wrf.ecse.rpi.edu//Research/Short_Notes/pnpoly.html>.