A Multi-Class Tracker using a Scalable Condensation Filter

Simon Denman, Vinod Chandran, Sridha Sridharan, Clinton Fookes
Image and Video Research Laboratory, Queensland University of Technology
GPO Box 2434, Brisbane 4001, Australia
{s.denman, v.chandran, s.sridharan, c.fookes}@qut.edu.au

Abstract

Tracking systems are typically targeted towards tracking a single class of object. In many real world situations, and in the ETISEO evaluation, it is advantageous to be able to track multiple classes of objects. In this paper we describe the adaptation of a single class tracking system to a multi-class tracking system, and describe a modified version of the condensation filter that can be used to track all objects, of all classes. We show that by using simple targeted detectors, we can achieve accurate tracking and can accurately distinguish between classes.

1 Introduction

Tracking systems[3, 13, 5] are typically targeted towards tracking a single class of object (i.e. people, cars, or a generic object/blob), limiting their functionality. The majority of systems that track multiple classes[1, 4, 9] do so by using a generic detector to detect every object, and then use rules or models to determine object classes.

In the ETISEO [1] evaluation (an evaluation of tracking and action recognition systems), many of the datasets require a system that can distinguish between multiple classes of objects. We adapt an existing single class tracker (person tracker) to be able to handle multiple classes (people and vehicles), by using multiple detectors targeted at the different object classes. We use a single condensation filter to track all objects in the system, and use a master-slave configuration to allow each tracker to have its own particle set, and allow the filter's particle count to be scaled as the system changes.

We test the resulting system on part of the preliminary ETISEO dataset (the RD datasets, captured on an outdoor roadway), and show that by using a detector for each class, we achieve accurate tracking results and are able to accurately distinguish between classes.

[1]Information on the ETISEO evaluation can be found at www.etiseo.net

2 Existing Work

A wide variety of tracking systems have been developed for various purposes, most focus on tracking only a single class of object (typically people), however a small number of multi-class trackers have been developed[1, 4, 9]. Bose et al.[1] and Ellis et al.[4] rely on a generic detection process to locate all objects of interest. Siebel et al.[9] used independent processes to locate both people and vehicles.

Single class person detectors are far more common and typically use some form of background segmentation[13, 5, 3], or optical flow[11, 8] as a basis for tracking; and use Kalman filters, or motion models (first or second order) to track and predict object positions. Haritaoglu et al.[5] developed a system for tracking objects using the expected motion of the object to restrict the search space, and relied on the matching of silhouettes to verify the object. Rather than tracking blobs Zhao et al.[13] proposed a system that used an ellipsoid shape model to locate and segment people from the motion image. Yamane et al.[11] proposed a method using optical flow and uniform brightness regions (a section where the optical flow cannot be detected) to track people, while Okada et al.[8] combined optical flow and depth information to track.

Other tracking systems such as those proposed by Kang et al.[7] and Zeng et al.[12] use particle filtering techniques. Kang et a. [7] modified the condensation algorithm to track multiple people in a crowded environment. A competition rule was introduced, such that each tracker suppresses the weights of samples around features belonging to another tracker, helping to avoid multi-modal distributions that can occur when multiple objects are close to one another. Zeng et al.[12] used active particle filtering (combining traditional particle filters with curve fitting) to track heads. Each particle is fitted to its local maxima prior to weighting, allowing the systems to use less particles to track objects.

3 Tracking System

We modify an existing tracking system[3] to allow it to track multiple classes of objects. We modify the system such that two object detection processes are run, each targeting a different class of object (people and vehicles). A third, generic, detection process is then run to locate any remaining regions, and match these to unmatched tracks (see figure 1). Objects are tracked using a modified condensation filter. We propose altering the filter so that the number of particles is scalable, and the filter is capable of assigning particles to one or more specific tracks. A simple colour model is used to deal with any ambiguities that arise when matching, and we propose a method to determine if a colour check is required during matching. Inheritance and polymorphism are used to provide a separate class for each object type, allowing all tracks to be stored in a common list, while still maintaining separation between the various object types. The road datasets (RD14 - 17) pose an additional challenge in that they contain significant amounts of strong shadowing. To overcome this, we perform raw fusion of the colour and IR images, creating a single image. The colour image is converted to Y'CbCr, and the luminance is substituted with the IR image. As all shadow information for the moving cars and people (the trees leave a thermal shadow) is contained in the luminance component the shadowing caused by the traffic is removed while still preserving the colour information (see figure 2). As the area being monitored is an outdoor road, there is no discernible thermal reflection from the moving objects. It should be noted that this is not necessarily the case for all environments, and it is likely that the same approach would not work as well in an indoor scene with reflective surfaces (i.e. tiled/polished floors).

3.1 Object Detection

A simple detection method is targeted at each of the classes we are tracking. The detectors share a hybrid motion detector-optical flow technique[2] as a basis, and scan for appropriate regions of motion. In the case of vehicle detection, this is a large region with a high amount of occupancy (we define occupancy as the percentage of moving pixels within the bounding box).

Person detection is performed by splitting the image into sub-regions which contain concentrated areas of motion, and then locating heads and fitting ellipses within each region[5, 13]. Working within subregions overcomes problems caused by people occupying a common column of the image. Heads are detected by combining the vertical projection and pixel height of the top contour (to aid in overcomming problems caused by holes in the motion image), and finding local maxima; which are then filtered by analysing the surrounding region. Ellipses are fitted to the valid heads at an aspect dependent on the candidate head, and if there is a suitable occupancy the candidate is accepted.

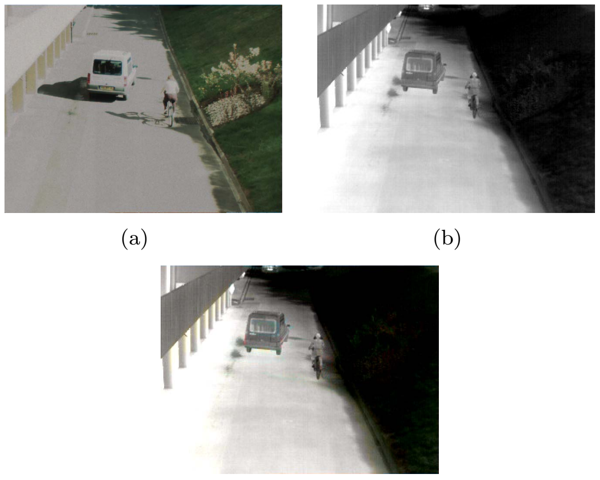
(a) (b) (c)

Figure 2. Shadow Removal - The Luminance in the colour image (a) is replaced with the IR image (b) to generate (c)

Detection procedures make use of a simple perspective transform to scale various detection thresholds (such as sizes, pixel counts) for objects at different depths in the scene.

A third detection method is used to locate any remaining objects that could not be located using the targeted detection procedures. This process locates any remaining large blobs of motion and matches them to unmatched tracks. No perspective transform is applied and no objects are added as a result of this detection stage. Rather it is used as a backup; and as a chance to re-evaluate object class classifications, as objects may be detected by this process because when they first entered the system, they were detected and classified incorrectly. When objects are updated by this process, simple tests (at this stage based on geometric rules) are carried out to determine if their classification is correct. If, over the course of several detections, they are consistently classified as a class different to the one they were created as, they are re-created as the correct

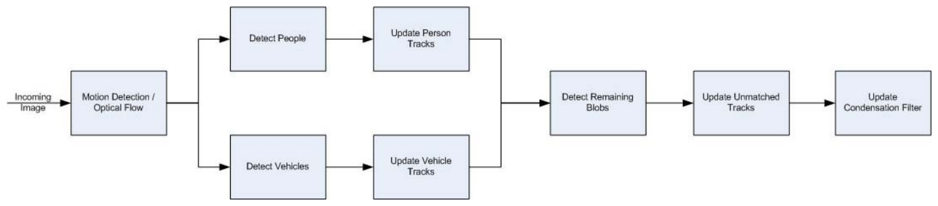

Figure 1. System Flow Chart

class.

3.2 Condensation Filtering

We use a condensation filter[6] to track objects in the system. We use simple object statistics and some basic rules governing how they relate to one another to manage the particles. We use the object position (x and y pixel coordinates), height, width, and the number of moving pixels to define the objects state (x, y, h, w, a). Each variable is free to move within the dimension limits, d_{min}, d_{max}, which are defined by the system (i.e. the limits of x and y are governed by the image size). The distribution of each dimension is Gaussian, with the mean at the the last observed position, and the variance equal to the maximum expected movement of a dimension from one frame to the next, e_{max}. Our system uses a master-slave implemen-

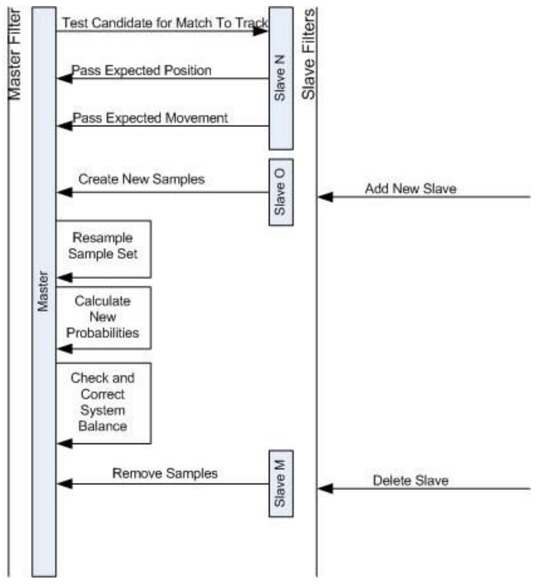

Figure 3. Condensation Execution Structure

tation (see figure 3). The tracker contains the master filter and each tracked object contains a slave. The master contains the samples and their probabilities, and is responsible for updating samples, re-sampling, and adding/removing tracks. The slaves allow each tracked object to communicate with the master, allowing its expected position and movement to be passed; and the slave to view its sample set, and make predictions about its location. When a slave is added, n samples are added to the system. At any given time, a slave s_i has n_i samples associated with it.

One problem that can occur with this configuration is that the particle assignments can become unbalanced (in a system with two tracks, one track may have 250 particles, while the other only has 50). We define the system as being out of balance when one or more tracks has a particle count of less than half the intended number of particles, $n_i < n/2$. When this occurs, we remove samples from the tracks that have 'too many' (where $n_i > n$) and create new samples for the tracks that have 'too few' (where $n_i < n$).

3.2.1 Updating

When calculating weights for samples, samples are 'assigned' to one of more objects, in a manner similar to the competition rule described by Kang[7]. Each sample has its probability calculated according to each slave, and the sample is added to the distributions of the top m slaves. A given track may contain samples which are first, second and third best match to it. This assignment prevents the weight of samples that belong to one object effecting other objects and predictions, and allows objects of different classes to have distinct sample sets. Each track has its probabilities normalised separately, ensuring that each track has a total probability of 1, without having every track associated with every sample. Updating the filter is done by re-sampling the sample set, as shown in figure 4. We look to emphasis particles that have proved to be a good match, rather than just randomly selecting particles to copy into the new set. For particles that have high probabilities we take multiple copies, while we ignore particles that have poor probabilities. Particles are selected from the set and copied at a rate of P_{sample}/P_{ave}, where $P_{ave} = 1/n$, and P_{sample} is the probability of the sample as calculated after the previous frame. We round the fraction down to the nearest whole number, unless it is less than one in which case it

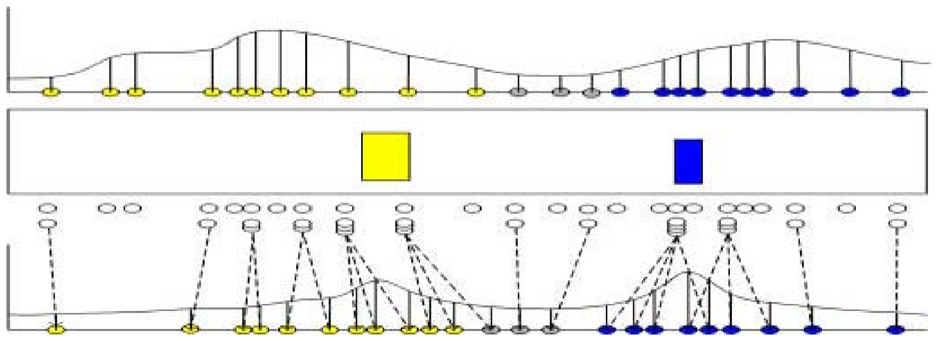

Figure 4. Condensation Process - At time t, the system has the state shown in the top graph. At time $t+1$, the image in the middle is processed, and the system is updated as shown in the lower graph. Yellow and blue samples are those that are distinct to the slaves, and gray samples indicate samples shared by both slaves.

is rounded to 1, ensuring any sample can be retained.

We use the ideal number of samples per track when calculating the average, so that for tracks with more than the ideal number of samples their average probability will be less than P_{ave}, and so fewer samples will be copied; while for tracks with too few samples, the average probability will be higher than P_{ave}, resulting in more samples being copied. This aids in keeping all slaves with the close to the ideal number of particles.

Part of the re-sampling procedure is adding a random value to each new sample. Rather than just add a random vector to each new sample, we propose adding the expected movement as well, allowing the sample set to move with the track.

$$S_{n+1} = S_n + R + M_i \tag{1}$$

where S_{n+1} is the new sample; S_n is the old sample; R is the random sample, which is within the range of $-e_{max}$ to $+e_{max}$, and M_i is the expected movement for the track, i. By adding the expected movement, we aim to have samples that are associated with a given track, remain associated with that track. This will help to improve tracking performance, and reduce the number of times we need to rebalance the system.

As part of all particle updating and creation, we apply a set of rules to each particle, to check that it is describes a valid object. These rules fall into two categories, limits placed on the individual dimensions (i.e. position cannot exceed the range d_{min} to d_{max}), and rules which describe a relationship between multiple dimensions. For our system we use one rule, and place limits on each dimension.

$$S_{area} <= S_{height} \times S_{width} \tag{2}$$

3.2.2 Object Model

Typically, condensation filters are used to track objects that have well defined models with clear state transitions. We are not fitting any rigid models, and so need a flexible model to evaluate candidates. We use two components; the probability of the next position according the distribution; and a simple position fit; which we combine to obtain a probability for a match. The first component is calculated directly from the distribution for the track.

The position fit provides a simple estimation of the match of a track to a sample. It is used primarily as a backup check, in case of anomalies in the distribution, such as those that may occur when the system becomes unbalanced.

$$E_{pos} = |t_x - s_x| + |t_y - s_y| \tag{3}$$

$$E_{bb} = |t_l - s_l| + |t_r - s_r| + |t_t - s_t| + |t_b - s_b| \tag{4}$$

$$p_{fit} = 1 - \frac{E_{pos} + E_{bb} - Err_{min}}{Err_{max}} \tag{5}$$

where E_{pos} and E_{bb} are the position and bounding box errors respectively; t and s are the observed track position and sample position; Err_{min} is the tolerance for the error; and Err_{max} s the maximum error. If the actual error is greater than the maximum allowed error, the result of this equation will be negative, in this case we simply assign a value of 0.

The two measures are combined as shown, to give the probability of a sample arising from a track.

$$p_{sample} = \left({}^{i<numDims}_{i=0} \prod p_i \right) p_{fit} \tag{6}$$

where i is the dimensions and $numDims$ is the number of dimensions, p_i is the probability of the dimension, and p_{sample} is the sample probability. This is used during the update of the condensation filter when each sample is evaluated; and during the candidate matching process, when objects are located in the incoming frame and matched to those in the system.

3.2.3 Dynamic Sizing

Rather than have a fixed number of samples for the filter, we propose dynamically altering the sample count as objects enter and leave the scene. For each track, we have an arbitrary number of samples, n, that are initially created about the objects initial position, and associated with that object.

$$s_{new} = o_{new} + 2 \times r \tag{7}$$

where s_{new} is the new sample, o_{new} is the new objects state, and r is a random value, in the range $-e_{max}$ to $+e_{max}$. This initialisation gives each tracked object a set of samples to model it immediately, rather than needing to allow a period of frames for the system to adapt to its presence. When an object leaves, n samples are removed from the system. Samples are removed according to how well they match candidates within the system. Samples that match no candidates are removed first, followed by samples that match poorly until enough samples are removed. Resizing the system, ensures that no unnecessary updates are done, and improves CPU utilisation.

3.3 Colour Model

A simple histogram model [10] is used to store colour information for each tracked object. We use the Cb and Cr channels to form the histogram. By ignoring the luminance channel, we remove any variation that is brought about by changed light conditions within the scene. The histogram uses bins that have a small overlap (20%), so slight variations from frame to frame do not result in a matching error as there is tollerance built into the histogram. The histogram is updated at a specified rate (about every 10 frames), and is used to match objects when the match from the condensation filter is ambiguous.

We define a fit quality metric to determine the level of ambiguity. We divide the fit of the second best match by the fit of the best match, to get a quality measure. If the quality measure is too low, we use the colour model to determine the correct match.

$$q = \frac{f_{j,k}}{f_{i,k}} \tag{8}$$

where $f_{i,k}$ is the fit for track i to the detected object k, and $f_{j,k}$ is the fit for track j to object k.

4 Results

The system was tested using the RD datasets contained within the first stage of the ETISEO evaluation database. As there is no ground truth data available yet, testing was conducted via visual inspection. The system runs without any user interaction (a parameter file is loaded at start up containing all required settings). The system performed well in testing, producing no major errors (tracks being swapped, prolonged incorrect classification). Small errors were observed when objects first entered, or as they exited the scene. Tracking became less reliable at these times (see figure 6(f)), and in some instances this resulted in a newly created object being lost and re-created. Occlusions were handled well and the different detectors were able to correctly segment their intended targets. Figures 5 and 6 shows a portion of the tracking results from the datasets, RD15 and RD16 respectively. The coloured rectangles indicate the tracks ID and type. The outer rectangle indicates the tracks ID; the inner rectangle indicates the type, red for person, yellow for vehicle.

5 Conclusions and Future Work

We have described the process of converting a single class person tracker into a system capable of tracking and distinguishing between people and vehicles, using simple targeted detectors. We have described how a condensation filter can be used to track all objects within the one filter, yet still have distinct groups of particles for each object; and how the number of particles processed by the filter can be scaled as the number of objects in the system changes. We have also shown how a simple colour model can be selectively used to provide an additional check when matching. Future work will focus on expanding the detection options, providing multiple detectors for each class, and detecting additional classes. Further work will also be done to expand the condensation filters so that they can apply separate rules and limits to different classes, allowing further flexibility in the tracking; and to investigate alternate means of image fusion for the incoming images.

References

[1] B. Bose, X. Wang, and E. Grimson. Detecting and tracking multiple interacting objects without class-specific models. Technical Report MIT-CSAIL-TR-2006-027, Computer Science and Artificial Intelligence Laboratory, MIT, April 25 2006.

[2] S. Denman, V. Chandran, and S. Sridharan. Adaptive optical flow for person tracking. In *Digital Im-*

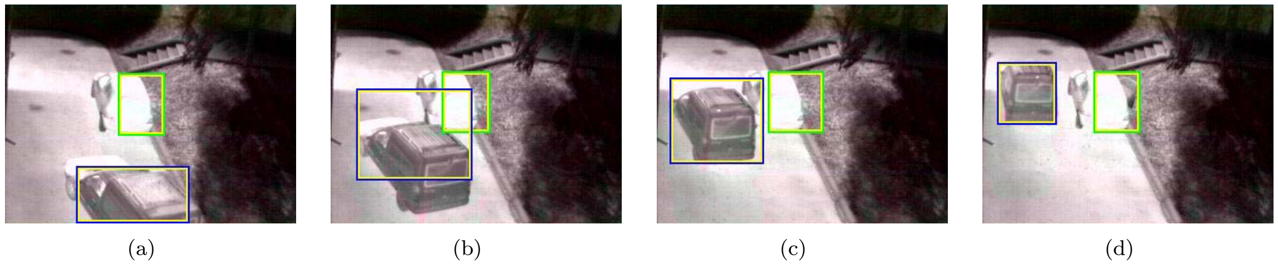

Figure 5. Tracking Results - Segment of Output from RD15

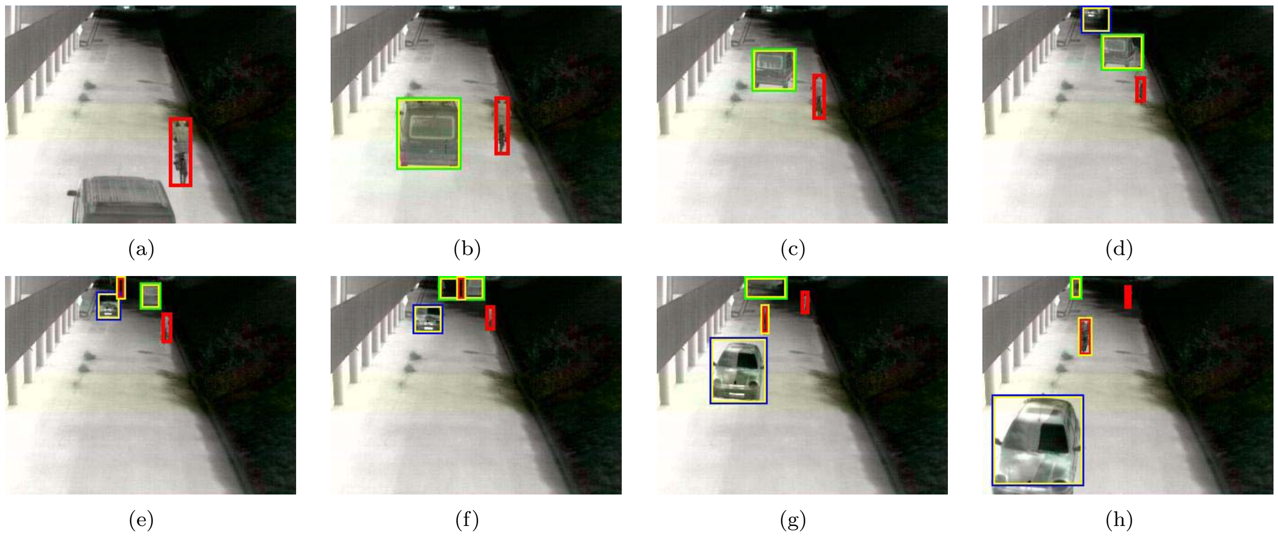

Figure 6. Tracking Results - Segment of Output from RD16

age Computing: Techniques and Applications, Cairns, Australia, 2005.

[3] S. Denman, V. Chandran, and S. Sridharan. Person tracking using motion detection and optical flow. In *The 4rd Workshop on the Internet, Telecommunications and Signal Processing*, Noosa, Australia, 2005.

[4] T. Ellis and M. Xu. Object detection and tracking in an open and dynamic world. In *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, Kauai, Hawaii, 2001.

[5] I. Haritaoglu, D. Harwood, and L. Davis. W4: real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809 – 830, 2000.

[6] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *IJCV*, 1998.

[7] H. Kang, D. Kim, and S. Y. Bang. Real-time multiple people tracking using competitive condensation. In *International Conference on Image Processing Proceedings*, volume 3, pages III–325–III–328 vol.3, 2002.

[8] R. Okada, Y. Shirai, and J. Miura. Tracking a person with 3-d motion by integrating optical flow and depth. In *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 336–341, 2000.

[9] N. T. Siebel and S. J. Maybank. Real-time tracking of pedestrians and vehicles. In *2nd IEEE Workshop on Performance Evaluation of Tracking and Surveillance*, Kauai, Hawaii, 2001.

[10] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, and R. W. Schafer. Methods for volumetric reconstruction of visual scenes. *International Journal of Computer Vision*, 57(3):179–199, 2004.

[11] T. Yamane, Y. Shirai, and J. Miura. Person tracking by integrating optical flow and uniform brightness regions. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 3267–3272 vol.4, 1998.

[12] Z. Zeng and S. Ma. Head tracking by active particle filtering. In *Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 82–87, 2002.

[13] T. Zhao and R. Nevatia. Tracking multiple humans in complex situations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1208–1221, 2004.