An Adaptive Fusion Architecture for Target Tracking

Gareth Loy, Luke Fletcher, Nicholas Apostoloff and Alexander Zelinsky
Department of Systems Engineering
Research School of Information Sciences and Engineering
Australian National University, Canberra 0200
{gareth, luke, nema, alex}@syseng.anu.edu.au

Abstract

A vision system is demonstrated that adaptively allocates computational resources over multiple cues to robustly track a target in 3D. The system uses a particle filter to maintain multiple hypotheses of the target location. Bayesian probability theory provides the framework for sensor fusion, and resource scheduling is used to intelligently allocate the limited computational resources available across the suite of cues. The system is shown to track a person in 3D space moving in a cluttered environment.

1 Introduction

Visually acquiring and tracking targets is a key problem in computer vision, and new and innovative techniques are constantly being developed. However, despite the impressive results obtained, it is clear that no single cue can perform reliably in all situations. The key to an efficient and robust vision system for tracking is to intelligently combine information from a number of different cues, whilst effectively managing the available computational resources.

The development of such a system must address several issues: which cue(s) should be used and when, how should the cues be combined, and how much computational resource should be expended on each cue?

This paper presents a framework for a vision system that addresses these issues by fulfilling the following criteria:

- efficiently allocate finite computational resources when calculating cues, accounting for the cue's expected utility and resource requirement.
- facilitate cues running at different frequencies.
- locate a target in multi-dimensional state space, eg. determining the target's 3D location and orientation.
- allow tracking of multiple hypotheses.

Section 2 of this paper reviews related work on sensor fusion for person tracking and Section 3 reviews the Bayesian approach to target localisation. Section 4 describes the architecture of the system, Section 5 details the sensing methods used, Section 6 reports on the system's performance, and Section 7 presents the conclusions.

2 Related Work

A number of researchers have utilised multiple cues to detect and track people in scenes, however, there have been few attempts to develop a system that considers the allocation of finite computational resources amongst the available cues, the notable exception being Crowley and Berard [2].

Crowley and Berard [2] use multiple visual cues to track a face in video for compression and transmission purposes. This system shows the advantage of selectively activating and deactivating cues based on confidence values, and achieves real-time performance with a Kalman filter fusing the active cues. However, each cue is limited to returning only a single hypothesis.

Recent work by Soto and Khosla [8] presents a system based on *intelligent agents* that adaptively combines multi-dimensional information sources (*agents*) to estimate the state of a target. A particle filter is used to track the target's state, and metrics are used to quantify the performance of the agents. Initial results for person tracking in 2D show a good deal of promise for this particle filter based approach.

Triesh and von der Malsburg [10] presented a system suitable for combining a number of cues to track a target's 2D location in an image. The output of each sensor was compared to a prototype describing the target (a face) with respect to that sensor. An adaptive weighting was given to each cue based on the cue's performance over recent frames, and the final result for each frame was determined as the weighted sum of the probability images. The system adapts to targets with changing appearance by dynamically updating the prototypes based on the sensor outputs at the target location over recent frames.

3 Bayesian Approach to Target Localisation

Given a state space of possible target poses, the problem of target localisation can be expressed probabilistically as the estimation of the posterior probability density function over the space of possible poses, based on the available data. That is, at time t estimate the posterior probability $P(s_t|e_{0...t})$ of a state s_t given all available evidence $e_{0...t}$ from time 0 to t.

Using Bayesian probability theory and applying the *Markov assumption*[1] the desired probability $P(s_t|e_{0...t})$ can be expressed recursively in terms of the current evidence and knowledge of the previous states. This is referred to as *Markov Localisastion* and is represented mathematically by the following equation,

$$P(s_t|e_{0...t}) = \eta_t P(e_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}) P(s_{t-1}|e_{0...t-1}) \quad (1)$$

where η_t is a constant normaliser that ensures the probabilities sum to one, $\eta_t = 1/P(e_{0...t-1})$.

The derivation, as detailed by Thrun [9], sequentially applies Bayes rule, the Markov assumption, the theorem of total probability and the Markov assumption again, and is as follows,

$$\begin{aligned} P(s_t|e_{0...t}) &= \eta_t P(e_t|e_{0..t-1}, s_t) P(s_t|e_{0...t-1}) \\ &= \eta_t P(e_t|s_t) P(s_t|e_{0...t-1}) \\ &= \eta_t (e_t|s_t) \sum_{s_{t-1}} P(s_t|e_{0...t-1}, s_{t-1}) P(s_{t-1}|e_{0...t-1}) \\ &= \eta_t P(e_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}) P(s_{t-1}|e_{0...t-1}) \end{aligned}$$

4 System Architecture

The system detailed in this paper uses a particle filter to track a population of target hypotheses in state space. A number of cues are calculated from image and state information and combined to provide evidence strengthening or attenuating the belief in each hypothesis.

Figure 1 shows the structure of the system. It consists of two subsystems: a particle filter and a cue processor, each of which cycle through their loops once per frame. These subsystems interact as shown by the thick arrows in the figure. The particle filter passes the current particle locations to the cue processor. The cue processor determines probabilities for the particles and passes these back to the particle filter. The inner workings of these subsystems are discussed below.

4.1 Particle Filter

The particle filter approach to target localisation, also known as the condensation algorithm [3] and Monte Carlo localisation [9], uses a large number of particles to 'explore' the state space. Each particle represents a hypothesised target location in state space. Initially the particles are uniformly randomly distributed across the state space, and each subsequent frame the algorithm cycles through the steps illustrated in Figure 1:

1. Deterministic drift: particles are moved according to a deterministic motion model (a damped constant velocity motion model was used).
2. Update probability density function (PDF): Determine the probability for every new particle location.
3. Resample particles: 90% of the particles are resampled with replacement, such that the probability of choosing a particular sample is equal to the PDF at that point; the remaining 10% of particles are distributed randomly throughout the state space.
4. Diffuse particles: particles are moved a small distance in state space under Brownian motion.

This results in particles congregating in regions of high probability and dispersing from other regions, thus the particle density indicates the most likely target states. See [3] for a comprehensive discussion of this method.

The key strengths of the particle filter approach to localisation and tracking are its scalability (computational requirement varies linearly with the number of particles), and its ability to deal with multiple hypotheses (and thus more readily recover from tracking errors). However, the particle filter was applied here for several additional reasons:

- it provides an efficient means of searching for a target in a multi-dimensional state space.
- reduces the search problem to a verification problem, ie. is a given hypothesis face-like according to the sensor information?
- allows fusion of cues running at different frequencies.

The last point is especially important for a system operating multiple cues with limited computational resources, as

[1] The *Markov assumption* states that the past is independent on the future given the current state.

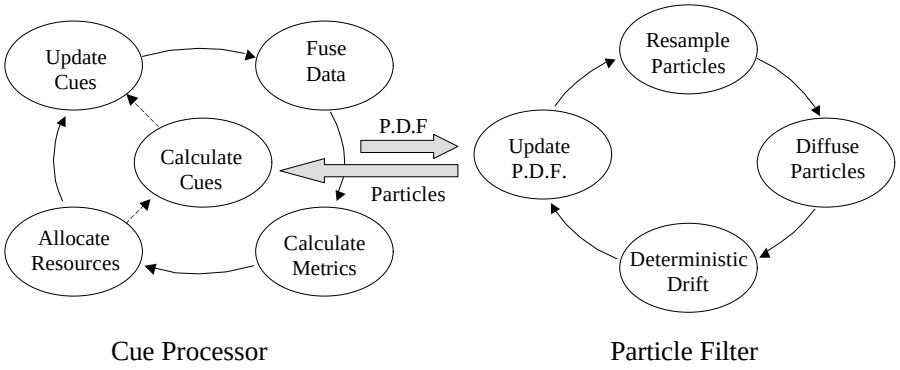

Figure 1. System architecture.

it facilitates running some cues slower than frame rate (with minimal computational expense) and incorporating the result from these cues when they become available.

If a cue takes n frames to return a result, by the time the cue is ready, the particles will have moved from where they were n frames ago. To facilitate such cues the system keeps a record of every particle's history over a specified number of frames k. The cue value determined for a particle $n \leq k$ frames ago can then be assigned to the children of that particle in the current frame, thus propagating forward the cue's response to the current frame. Conversely, probabilities associated with particles that were not propagated are discarded.

4.2 Cue Processor

The cue processor deals with the calculation and fusion of cues. It also determines metrics measuring the performance of each cue, and the allocation of computational resources to individual cues.

Each frame the cue processor cycles through the steps illustrated in Figure 1:

1. Update cues: accesses recently calculated cues.
2. Fuse data: fuses the results of different cues to estimate the overall probability for each hypothesised target state.
3. Calculate metrics: Determine the metrics quantifying the performance of each cue on the last image frame.
4. Allocate resources: based on the anticipated performance of the individual cues, allocate computational resources to maximise the quality of information obtained.

The *calculate cues* component of the system accepts requests for cue measurements and handles the requests using only the quantity of computational resource allocated to it by the *allocate resources* component. Some calculations may take longer than a single frame, but as discussed in the previous section, the update PDF component is able to accommodate these slow cues and propagate their effect through to the current probability values.

4.2.1 Quantifying Cue Performance

The performance, or *utility*, of each active cue is estimated every frame, and used to decide the distribution of computational resources across the cues (see Section 4.2.2).

Fusing the results of all available cues is assumed to give the best estimate of the true PDF $P(e_t|s_t)$ across the state space . So the performance of the j^{th} cue can be quantified by measuring how closely the cue's PDF $P(e_{j,t}|s_t)$ matches $P(e_t|s_t)$. This can be done using the relative entropy, or the Kullback-Leibler distance [6], an information theoretic measure of how accurate an approximation one PDF is to another, given by

$$\delta_t \left(P(e_t|s_t), P(e_{j,t}|s_t) \right) = \sum_{s_t} P(e_t|s_t) \log \frac{P(e_t|s_t)}{P(e_{j,t}|s_t)}$$

where s_t are the particle states at time t. Soto and Khosla [8] used this metric to rate the performance of their cues, and Triesh and von der Malsburg [10] considered it, but opted for a simpler *ad hoc* measure.

The utility of the j^{th} cue at time t is defined as

$$u_t(j) = -\delta_t(P(e_t|s_t), P(e_{j,t}|s_t))$$

4.2.2 Resource Allocation

The computational resources of the system are dynamically allocated based on the performance metrics that predict the future performance of each cue. This configuration not only optimizes the performance of the cues for the current situation, as it dynamically chooses the most suited cues to the

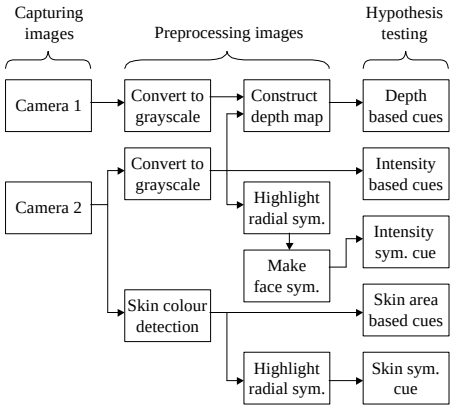

Figure 2. Sensing process

current conditions, but it also makes the system flexible to future changes in hardware and software.

The operation of the resource allocator is a simple process of searching through the complete space of possible cue combinations for the one that has the best overall utility. The overall utility of a combination of cues is the sum of the performance metrics of each cue.

A certain fraction of the time between each frame is devoted to cues running at frame rate, while the rest of the time is devoted to those cues that run at speeds less than frame rate. The performance metric of a cue running at a rate slower than frame rate is reduced exponentially by a discount factor for each frame it is late. The discount factor was introduced on the premise that a result obtained over 8 frames is worth less than one that is obtained over 2 frames.

The resource allocator starts by generating all cue combinations that can run in the time allocated for cues running at frame rate. It then chooses the combination with the best overall utility. A list of all combinations of the remaining cues over all possible slower frame rates is generated such that no combination exceeds the time allocated for the slower cues. Initially the slower rates were set to once every 2, 4, and 8 frames. Taking into account the discount factor for slow cues, the combination that has the best overall utility is chosen.

5 Sensing

The system was implemented with two weakly calibrated colour stereo video cameras as sensors. The images from these cameras undergo some preprocessing and are then passed to the cues where each target location hypothesis is tested by computing all active cues. Figure 2 shows the sensing process when all cues are active. Both the preprocessing and hypothesis testing are discussed below.

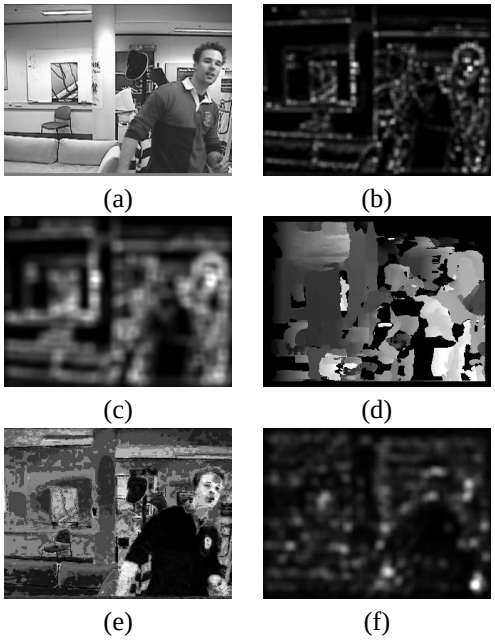

Figure 3. (a) Intensity image, (b) Radial symmetry image, (c) Facial symmetry image, (d) Depth map, (e) Skin colour likelihood image, (f) Radial symmetry of skin colour likelihood image searching for a radius of 15 pixels.

5.1 Preprocessing

Preprocessing is only performed *once* for each new set of images, whereas hypothesis testing requires one test for *every* target hypothesis generated by the particle filter. The preprocessing required for each frame is governed by the cues that are to be computed. These dependencies are illustrated by the network in Figure 2.

Figure 3 shows the output from the preprocessing of a stereo pair of 320×240 images.

5.1.1 Depth Map

A dense depth map is generated from intensity images using the approach of Kagami *et al* [4]. The depth map presents the depths as viewed from Camera 2.

5.1.2 Skin Colour Detection

A skin colour likelihood image is generated from one channel of the stereo image stream. The skin colour likelihood of each pixel is determined by reference to a precomputed skin colour histogram. The histogram was generated by plotting skin colour samples in *CIE Lab* colour space, discretising the colour space into $16 \times 64 \times 64$ bins, and blurring these by convolution with Gaussian spheres,

then normalising so the maximum value is unity. $173,000$ skin colour samples were used from 346 images of faces of people of varying race captured under different lighting conditions (none of these samples were from people later tracked by the system). This method is an extension of the approach presented by Cia and Goshtasby [1] who built a two-dimensional histogram of skin chrominance in the *CIE ab* chrominance space. The three-dimensional histogram used in this paper provides more reliable performance by reducing the number of false positives.

5.1.3 Radial Symmetry

Loy and Zelinsky's radial symmetry operator [7] is applied to the skin colour likelihood image to highlight regions of skin that are approximately circular and of a size appropriate to be a face.

The operator is also applied to the intensity image to highlight small dark regions such as the eyes. This output is then convolved with a blurred annulus to highlight the regions between potential eye pairs, and is referred to as the *facial symmetry image*.

Each application of the radial symmetry operator is performed at three different radiuses to detect targets at three ranges of depth away from the camera.

5.2 Hypothesis Testing

Each particle from the particle filter presents an hypothesis target location in state space. Using a pinhole camera model and the generic head model in Figure 4(a), the size, location and orientation of each hypothesis is determined in the image. All active cues are calculated for each hypothesis.

Each cue returns a set of probabilities $P(e_j|s_t)$ indicating the i^{th} active cue's belief in the j^{th} hypothesis. These probabilities are fused to determine the overall belief in the j^{th} hypothesis b_j as follows

$$P(e|s_t) = \prod_j (P(e_j|s_t)(1-\alpha)+\alpha)$$

where $\alpha \in (0,1)$ is included to prevent a zero value for a single $P(e_j|s_t)$ forcing $P(e|s_t)$ to zero. In this paper $\alpha = 0.1$ was used.

5.2.1 Cue Suite

The cues were chosen on the grounds of simplicity and efficiency. All cues use the head model dimensions shown in Figure 4(a). In the proceeding descriptions the *face region* and *face boundary* refer respectively to the light and dark grey regions in Figure 4(b).

1. **Elliptical Skin Region Cue** returns the average skin likelihood of the pixels within the face region.

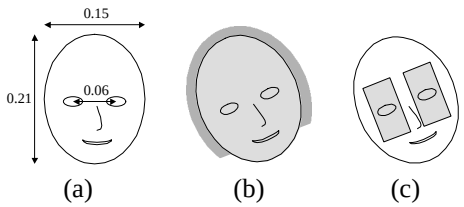

Figure 4. (a) Generic head target with dimensions in meters, (b) Elliptical face region (light) and face boundary region (dark), (c) Search regions for integral projection.

2. **Skin Detector Cue** returns 0.5 if any of the pixels sampled in the face region had skin likelihood values within the top 10% of values in the current skin likelihood image, and 0 otherwise.

3. **Non-skin Boundary Cue** returns a high value if there are few skin colour pixels in the face boundary region.

4. **Radially Symmetric Skin Cue** is the value of appropriate (as determined by the hypothesised depth) skin-based radial symmetry image at the target location.

5. **Radially Symmetric Intensity Cue** is the value of appropriate (as determined by the hypothesised depth) facial symmetry image at the target location.

6. **Radially Symmetric Eye Cue** is the value of appropriate (as determined by the hypothesised depth) radial symmetry image at the hypothesised eye locations.

7. **Eye Location Cue** uses integral projection [5] to search the regions in Figure 4(c) of the intensity image for the darkest bands aligned with the horizontal axis of the head. A high value is returned if these are close to the hypothesised eye locations.

8. **Head Depth Cue** compares the depths in the face region with the hypothesised depth of the target, returning a high value when these are in agreement.

9. **Head Boundary Depth Cue** compares the depths in the face boundary region to the hypothesised target depth, giving a high value when these are different.

6 Experimentation

An initial implementation of the system was developed as an object orientated algorithm in Matlab. To simulate real-time resource requirements the computational cost for each cue was estimated from the CPU time required. Stereo image sequences were captured at 30Hz with a pair of uncalibrated NTSC video cameras.

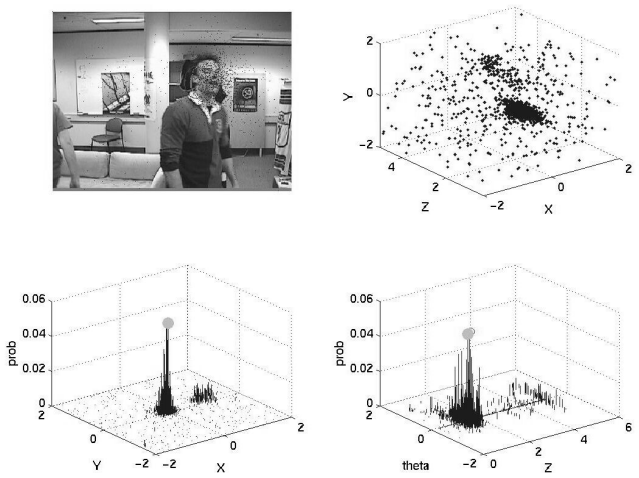

Figure 5. Frame in tracking sequence showing (clockwise) particles in image, in 3D space and particle distributions over x, y and z, θ states.

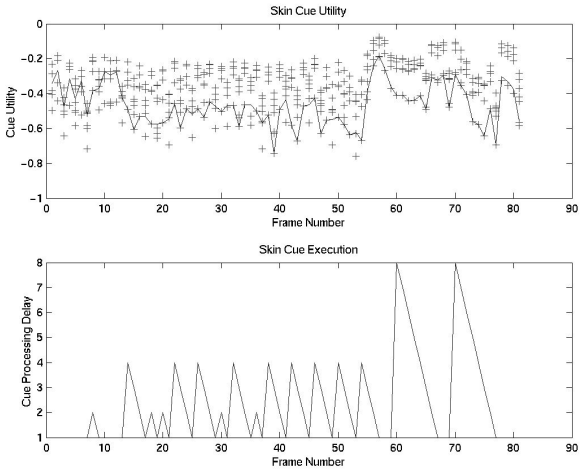

Figure 6. Top: Utility for the elliptical skin region cue (solid) and all other cues (+'s) during a portion of a tracking sequence. Bottom: Cue processing delay for the elliptical skin region cue.

The system's performance was demonstrated tracking a human face in a cluttered scene. A sample frame of the sequence is shown in Figure 5 along with particle distributions. The full sequence can be viewed online at www.syseng.anu.edu.au/rsl. Cues were dynamically scheduled to run once every 1, 2, 4 or 8 frames according to their calculated utility and computational cost.

Figure 6 shows the cue utility and processing delay for a cue during a tracking sequence. Note that as the cues utility decreases relative to the other cues (ie. from frames 50 to 60) its processing delay grows as it is allocated less resources.

The simplicity of the cues means no one cue is able to reliably track the head in 3D space, however, by fusing multiple cues the ambiguity in the target location is reduced. Furthermore, by adaptively rescheduling the cues the system is able to enhance the tracking performance possible under a given resource constraint.

7 Conclusions

A vision system has been presented that uses multiple cues to track a target in a multi-dimensional state space. Finite computational resources are efficiently allocated across the cues, taking into account the cue's expected utility and resource requirement. The system can accommodate for cues running at different frequencies, allowing cues performing less well to be run slowly in the background for added robustness with minimal additional computation.

References

[1] J. Cai and A. Goshtasby. Detecting human faces in colour images. *Image and Vision Computing*, 18:63–75, 1999.

[2] J. L. Crowley and F. Berard. Multi-modal tracking of faces for video communications. In *Computer Vision and Pattern Recognition, CVPR '97*, June 17-19 1997.

[3] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[4] S. Kagami, K. Okada, M. Inaba, and H. Inoue. Design and implementation of onbody real-time depthmap generation system. In *IEEE Int. Conf. on Robotics and Automation*, California, USA, 2000. IEEE Computer Press.

[5] T. Kanade. Picture processing by computer complex and recognition of human faces. Technical report, Kyoto University, Dept. of Information Science, 1973.

[6] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[7] G. Loy and A. Zelinsky. A fast radial symmetry operator for detecting points of interest in images. Submitted to *IEEE Trans on Pattern Recognition and Machine Intelligence*, 2001.

[8] A. Soto and P. Khosla. Probabilistic adaptive agent based system for dynamic state estimation using multiple visual cues. In *Proc. of Int. Sym. Robotics Research (ISRR)*, 2001.

[9] S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[10] J. Triesh and C. von der Malsburg. Self-organized integration of adaptive visual cues for face tracking. In *Proc. of IEEE International Conference on Face and Gesture Recognition*, pages 102–107, 2000.