ACCELERATED PEOPLE TRACKING USING TEXTURE IN A CAMERA NETWORK

Wasit Limprasert[1], Andrew Wallace[2] and Greg Michaelson[1]
[1]*School of Mathematical and Computer Sciences,Heriot-Watt University, Edinburgh, Scotland, U.K.*
[2]*School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh, Scotland, U.K.*

Keywords: People Tracking, Particle Filter, CUDA.

Abstract: We present an approach to tracking multiple human subjects within a camera network. A particle filter framework is used in which we combine foreground-background subtraction with a novel approach to texture learning and likelihood computation based on an ellipsoid model. As there are inevitable problems with multiple subjects due to occlusion and crossing, we include a robust method to suppress distraction between subjects. To achieve real-time performance, we have also developed our code on a graphics processing unit to achieve a 10-fold reduction in processing time with an approximate frame rate of 10 frames per second.

1 INTRODUCTION

There has been a dramatic increase of interest in video analytics, the observation and tracking of human (and other) subjects through video sequences. For example, CCTV networks are used to record and counteract criminal acts in town centres, public buildings and transport termini, and to observe shopping patterns in a supermarket. As a key component of such systems, we require the ability to track and identify multiple human subjects as they move not just within the field of view of one camera, but as they move from camera to camera through the network.

In this paper, we present an approach to multiple subject tracking in multiple camera views based on the well-established particle filter framework. Our first contribution is to introduce a new visual likelihood computation based on ellipsoid projection that incorporates texture acquisition. Our second contribution is to accelerate the tracking by implementation and evaluation on a graphics card using CUDA technology. We evaluate our approach in terms of efficiency, accuracy and speed of computation using the standard metric MOTA and PETS09 datasets, comparing our CUDA implementation with a standard CPU implementation.

1.1 Background

The condensation algorithm or particle filter is a well established method for tracking a subject in video sequences using Bayesian sequential estimation (Isard and Blake, 1996). The prior probability density function in the state space is predicted from previous knowledge of the subjects. This prior is combined with a likelihood function to generate the posterior in the context of Bayesian estimation. In the particle filter methodology, the prior and posterior states are represented by a group of particles, points in a multi-dimensional state space corresponding to the state vector. Generally, the state expressed by these particles is compared with an observation and the likelihood is computed from the similarity of a projected representation of the state with the observed images. This constructs the posterior density of the target state.

The particle filter has been used extensively in tracking within a 3D world using 2D video data, *e.g.* (Sidenbladh et al., 2000; Jaward et al., 2006; Peursum et al., 2007; Bardet and Chateau, 2008; del Blanco et al., 2008; Husz et al., 2011). For video analytics, there is a trade-off between the richness of a full articulated description of a human and a simple ”blob” tracker, that represents a human global state alone. For 3D articulated human tracking, the dimension of the state space, e.g. (Peursum et al., 2007), of a single person can be as high as 29, causing a huge problem of complexity of search. If either approach is extended to multiple target tracking, the joint-state can improve tracking quality when different targets are in close proximity. However, the search space grows exponentially, which makes articulated track-

Limprasert W., Wallace A. and Michaelson G..
ACCELERATED PEOPLE TRACKING USING TEXTURE IN A CAMERA NETWORK.
DOI: 10.5220/0003813802250234
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2012), pages 225-234
ISBN: 978-989-8565-04-4

ing extremely difficult, rarely attempted. To reduce the overall complexity of search, Kreucher (Kreucher et al., 2005) suggested an adaptive method to switch between independent and joint-states.

The likelihood or weight of a particle filter can be computed from the silhouette (Deutscher et al., 2000; Bardet and Chateau, 2008; Husz et al., 2011), edge (Deutscher et al., 2000), color distribution (del Blanco et al., 2008), or texture (Sidenbladh et al., 2000; An and Chung, 2008), often weighted by chamfer distance (Husz et al., 2011). Occlusion is a serious problem in visual tracking, either by other subjects or scene architecture. To solve problems of occlusion Vezzani (Vezzani et al., 2011) used pixel-wise colour distributions and assumptions on shape changing in 2D to label associations between pixel and target. However, such an approach relies inevitably on ad-hoc assumptions.

To summarise, the particle filter framework is powerful, but has issues of both robustness and complexity in practical implementation in cluttered scenes. In this paper we present new work on texture acquisition and likelihood computation to address the former problem, and a CUDA implementation to address the latter problem.

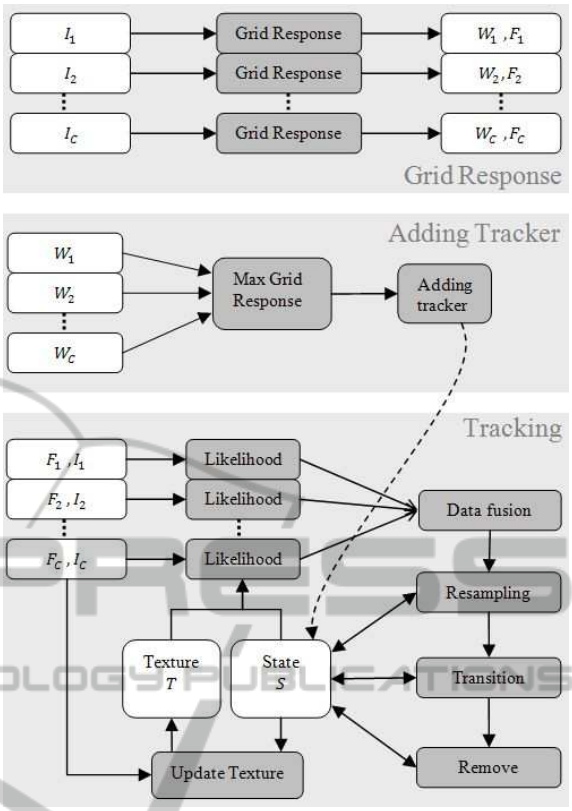

Figure 1: Overall processes from top to bottom are grid response preprocessing, adding tracker and tracking module.

2 DETECTION AND TRACKING

Our framework includes subject detection and tracking in a network of cameras. Figure 1 shows a schematic overview in which the white blocks represent data objects and the shaded blocks are functions. Input colour images, I_c, from calibrated cameras, c, are synchronized and saved to host memory. A pixel p of the image from a camera c is denoted by $I_{c,p} = (I_r, I_g, I_b)$ and $p = (p_x, p_y)$ indicates pixel position on the image plane. For detection of a new subject, I_c is analyzed by a grid response function to determine $W_c = (w_1, \ldots, w_g)$, where g is a grid index. This can activate and initiate an available tracker. Once detected and a tracker activated, control then passes to the tracking filter shown in the lower part of Figure 1. We now explain the detection (by grid response) and tracking functions in more detail.

2.1 Detection: The Grid Response Function

The 'grid' is a uniformly distributed set of 2D points on the ground plane $m_g = (x, y)$, where subscript g is grid index $g = 1, 2, \ldots, g_{max}$. The objective is to determine the response at each grid position, as shown in Figures 2 and 3. An ellipsoidal template of a person is projected onto the image plane at each point. From the projected ellipse in section 2.3.3, a rectangular bounding block $r_g = (p_1, p_2, p_3, p_4)$ is calculated. As the cameras are fixed and pre-calibrated, the set of grid positions, $M = (m_1, \ldots, m_g)$ and of rectangular boundaries $R = (r_1, \ldots, r_g)$ for any camera are precomputed and saved in a look up table.

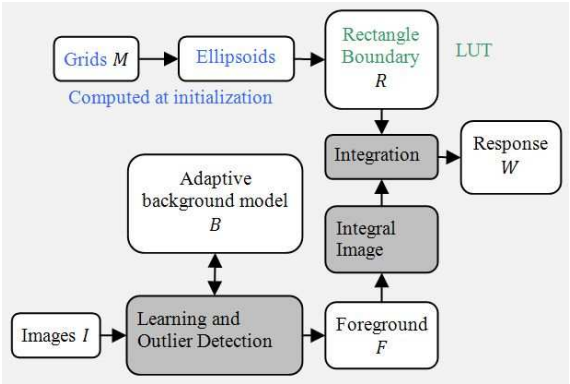

Figure 2: Internal structure of a grid response preprocessing unit in Figure 1.

To detect a subject, we employ a background model, computed from I. The color distribution of each background pixel is estimated from an extended version of an adaptive Gaussian mixture model (Stauffer and Grimson, 1999). For a particular pixel, this is $B_p = (\mu_r, \sigma_r, \mu_g, \sigma_g, \mu_b, \sigma_b)$. The confidence intervals for the background observation data are defined by the chi-square test with 3 degrees of freedom at p-value 0.05 significance, confidence interval $Q < 7.82$. Any pixel that lies out of this confidence interval is classified as foreground and saved to a binary foreground image F.

Given a rectangular boundary R in the look up table and the foreground image F, we determine the grid detection response w_g as described by (Viola and Jones, 2001; Derpanis, 2007). The integral image J must be calculated first then the detection response w_g is computed from J and r_g, where r_g is the set of corners $\{p_1, p_2, p_3, p_4\}$ as in (Viola and Jones, 2001).

$$w_g = \frac{1}{4h_x h_y}[J(p_1) - J(p_2) - J(p_3) + J(p_4)] \quad (1)$$

where J is the integral image.

$$J(p_x, p_y) = \sum_{y=1}^{p_y} \sum_{x=1}^{p_x} F(p_x, p_y) \quad (2)$$

The detection response w_g of a camera c is added into the detection response pool $W_c = (w_1, w_2, \ldots, w_g)$. Then the global maximum response is selected and added to the list of current subjects by the Adding Tracker function.

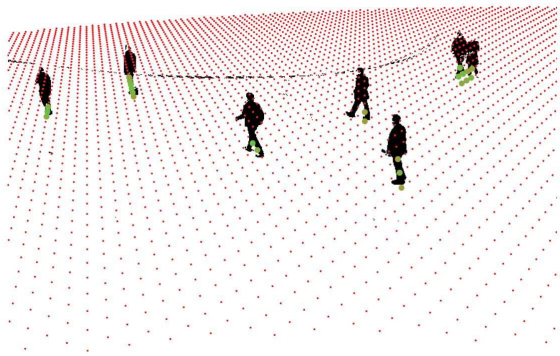

Figure 3: Grids (red dots) are projected on foreground image. Detection responses are represented by green circles.

2.2 Adding a Tracker

The maximum value in the detection response pool is selected. This maximum is verified by an empirical threshold at a fill-in ratio of 6.5 (typically between 0.6 and 0.8) as a trade off between missing a subject and the false alarm rate. A verified maximum grid response initiates a new tracker. A grid response in close proximity to a current tracker is excluded to prevent repeated detection.

When a new tracker is activated, a corresponding set of states, $S = \{s_{k,1}, s_{k,2}, \ldots, s_{k,n}\}$, is initialized. A single state, $s_{k,n}$ consists of position, velocity, height, visibility, persistence and ID,

$$s_{k,n} = [(\text{Pos})\ (\text{Vel})\ (\text{Ht})\ (\text{Vis})\ (\text{Per})\ (\text{ID})] \quad (3)$$

where n and k are the particle and tracker indices respectively. The position of all particles is set to the grid location. The velocity is uniformly distributed in all directions, with a uniform distribution of magnitude, in the absence of prior knowledge. The height of all particles is set to an average human height of 1.7m. The visibility and persistence are set to 1 and a unique ID is generated. The latter variables are used in tracking to allow for short period disappearance due to occlusion.

As stated earlier, we introduce a textured approach based on the projection of an ellipsoid as a target signature, using a similar pixel-wise adaptive Gaussian texture model to the background signature. At the detection stage, the texture is unknown. Thus the variance of the distribution is set to a maximum and the mean to a mid value.

2.3 Tracking using a Particle Filter Methodology

We use a SIR (sequential importance resampling) particle filter to perform sequential Bayesian estimation. This involves three functions: likelihood, resampling and transition.

Let $X = \{x_1, x_2, \ldots\}$ be an observable set of data and s_n is a state. Note that replacing density function parameter with s_n requires fundamental property of particular measurement. In our case we use camera model to transform s_n to density parameters on the image plane. The basic Bayesian equation of the particle filter methodology is given by

$$\mathrm{P}(s_n|X) \propto \mathrm{P}(s_n).\mathrm{P}(X|s_n). \quad (4)$$

The probability $\mathrm{P}(s_n)$ and collection of many states, $S = \{s_n; n = 1, 2, \ldots\}$, form a non-parametric representation of the prior density. A state s_n is a coordinate in a multi-dimensional state space. The coordinate is a point and is called a *particle*. We can represent a probability density function by points and density values. This non-parametric representation requires many particles to produce accurate estimation. The particle representation is also used for expressing another two terms, $\mathrm{P}(X|s_n)$ and $\mathrm{P}(s_n|X)$. The likelihood $\mathrm{P}(X|s_n)$ is the probability of the observations

X being measured with given state s_n. The product of the prior $\mathrm{P}(s_n)$ and the likelihood $\mathrm{P}(X|s_n)$ forms a posterior $\mathrm{P}(s_n|X)$.

The particles are re-sampled, where the number of offspring at the parent coordinate is proportional to the posterior. The re-sampling makes all particles have the same weight. The offspring particles are transformed by a stochastic transition function and Markov model to generate the prior of the next frame.

For a comprehensive description of the operation of the particle filter the reader is referred to the excellent text by (Ristic et al., 2004). In this paper, we describe only the details of our own implementation, including the contributed texture learning and tracking, and the accelerated implementation.

2.3.1 Transition

The transition function transforms the posterior in the previous frame to a prior density in a succeeding frame by an assumption of continuous motion with a noise perturbation model. In our tracking algorithm, we can use simple equations of motion of subjects on the ground plane with some perturbation to model progression from frame to frame of the video sequence. The stochastic equations Eq5 and Eq6 show the propagation of a particle $S_{k,n}$, where $\mathtt{W}$ is a Wiener process, which generates standard Gaussian noise in a stochastic model, and Δ is the time interval between the previous and current observation.

$$(\mathrm{Vel})_t = (\mathrm{Vel})_{t-1} + \sigma_{vel}\Delta^{\frac{3}{2}}\mathtt{W} \quad (5)$$

$$(\mathrm{Pos})_t = (\mathrm{Pos})_{t-1} + \Delta(\mathrm{Vel})_{t-1} + \sigma_{pos}\Delta^{\frac{1}{2}}\mathtt{W} \quad (6)$$

$$(\mathrm{Ht})_t = (\mathrm{Ht})_t + \sigma_{ht}\Delta^{\frac{1}{2}}\mathtt{W} \quad (7)$$

Acceleration is modelled as a Wiener process, which has a high probability to keep the previous velocity. The square root of Δ is a scaling factor of the Wiener process. Position is the integral of velocity over time and has some small noise due to observation. In our experiments σ_{pos} is small compared to σ_{vel}. We also allow height to vary slightly over time.

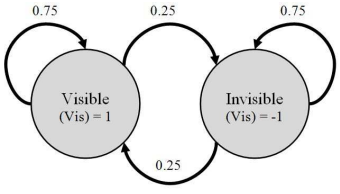

Figure 4: Markov chain model of visibility state.

The visibility (Vis) is modeled as a Markov chain as shown in the Figure 4. The visibility state has a probability 0.25 to switch between states. In equilibrium, the numbers of visible and invisible particles are equal. All invisible particles get a small constant likelihood. When a target is occluded, the likelihood of a visible state is low due to lack of observation. The invisibility state dominates and Bayesian interference generates pressure on the probability density forcing particles to change from visible into invisible states. When particles are in an invisible state the persistence level decreases.

The transition function for persistence (Per) does not include noise, depending on the previous (Per) and current (Vis) states. The visibility (Vis) can be positive or negative, and its integration over time is persistence (Per).

$$(\mathrm{Per})_t = (\mathrm{Per})_{t-1} + \Delta(\mathrm{Vis})_t \quad (8)$$

Persistence allows the system to estimate the existence level of a target. If a target disappears, (Per) reduces. When (Per) is below a threshold, it is reasonable to assume that the target is no longer visible. This could be due to occlusion or leaving the scene.

2.3.2 The Likelihood Function

The likelihood function, L, is the probability of occurrence of an observation configuration $\{x_1, x_2, \ldots, x_N\}$ given the prior probability density $g(x,\theta)$, where parameter θ is known a-priori. We define the observation density function $f(x)$ from the observation set and Eq.10 can be solved by Monte Carlo numerical integration and log-likelihood can be written as the integral transform in Eq.11, where $log[g(x,\theta)]$ is the integral kernel. The kernel integral transform and log-likelihood are analogous. Both are maximized when $f(x)$ and $g(x,\theta)$ are perfectly matched.

$$L = \prod_i g(x_i,\theta) \quad (9)$$

$$\log[L] = \sum_i \log[g(x_i,\theta)] \quad (10)$$

$$\log[L] = N\int f(x).\log[g(x,\theta)]\,dx \quad (11)$$

Therefore, the likelihood weight can be computed from the kernel integral.

2.3.3 The Ellipse Kernel

Eberly (Eberly, 1999) has shown the projection of a 3D ellipsoid to a 2D ellipse function. We express this function as a contour φ in Eq12, and use φ as an integral kernel to calculate the log-likelihood. This integrates the camera models into the projection framework.

$$\varphi = 1 - A(p_x - c_x)^2 - B(p_x - c_x)(p_y - c_y) - C(p_y - c_y)^2 \quad (12)$$

From camera calibration we know camera matrix K, rotation matrix R and translation vector T. We calculate eyepoint e from Eq13. The direction vector $d = (d_x, d_y, d_z)$, which passes through e and the pixel $p = (p_x, p_y, 1)$ is computed from Eq14.

$$e = -R^{\mathrm{T}}T \quad (13)$$

$$d = R^{\mathrm{T}}K^{-1}p \quad (14)$$

Note that Eq14 is not normalized. We insert e and d into Eberly's projection framework to calculate a set of arbitrary constants (c_x, c_y, A, B, C) in Eq12. The size and position of the ellipsoid are also used in contour calculation. Given pixel coordinate $p = (p_x, p_y)$ we can compute contour $\varphi(p)$ and the contour level is used in order to determine bounding block of target and penalty function.

To estimate and evaluate the texture of the ellipsoid, we have to map a pixel p to a coordinate on ellipsoid surface $q = (z, \theta)$. The intersection between a ray and an ellipsoid is described by (Schneider and Eberly, 2003). The ellipsoid is scaled to a unit sphere and the intersection point is mapped to q.

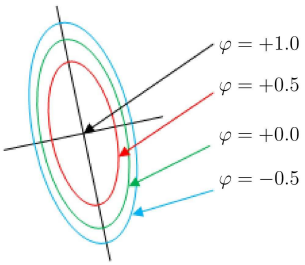

Figure 5: An ellipse contour function derived from a ellipsoid Eq12.

Given the ray traced mapping between the image surface and the ellipsoid surface that represents the bounding surface of the tracked target, we now describe the formation of the likelihood function.

2.3.4 Silhouette Likelihood

Given a foreground image F and a kernel $\varphi_{k,n}$ which is derived from state $s_{k,n}$, we estimate the silhouette log-likelihood λ^s by the inner product of Eq11. Assuming a normal distribution of foreground pixels, the log of the normal density is the ellipse contour function. Thus, the kernel function is constructed from φ in Eq12.

$$\lambda^s_{k,n}(S) = \frac{2}{\pi ab} \int_{\varphi=\varepsilon}^{\varphi=1} \varphi_{k,n}(p) F(p)\, dp \quad (15)$$

The log-likelihood is the summation of the product of φ and F in the boundary $\varepsilon < \varphi(p) < 1$. The boundary can be adjusted by setting ε as in Fig.5. Setting ε=0 is insensitive to misalignment. The tracking estimate tends to drifts away from the camera because of perspective effect. An ellipsoid further away from the camera generates a smaller ellipse on image plane, which requires a smaller number of foreground pixels to fill it in. This means the smaller (further) ellipse has high probability to be completely filled than the larger (closer) ellipse. To reduce this bias we set larger boundary, $\varepsilon = -0.5$, as it creates a negative value for any mis-aligned pixel. The silhouette likelihood is very robust in the case where targets are well separated, but is unable to handle overlaps between targets.

2.3.5 Distraction Suppression

A major problem in multiple target tracking is distraction when subjects are close on the image plane. We use the ellipse contour function as a membership probability, forming a penalty function to suppress distraction.

The membership probability is defined as the association between the observed foreground pixels and the projected ellipse. When two or more ellipses overlap, any foreground pixel possibly contributes to those ellipses. The membership probability is a decreasing function of the number of ellipse layers covering a particular foreground pixel. The pixel ownership can be calculated from the value φ, given the ellipse parameters and pixel location p.

Let the kernel function $\varphi_{k,n}(p)$ be a function of the state of tracker k and pixel position p. The membership probability of pixel p belonging to ellipse k of particle n is denoted by $m_{k,n}(p)$. This depends on the state of all active trackers i.

$$m_{k,n}(p) = \frac{H_{k,n}(p)}{\sum_{\forall i} H_{i,n}(p)} \quad (16)$$

$$H_{k,n}(p) = \begin{cases} 1 & \text{if } \varphi_{k,n}(p) > \varepsilon \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

The membership probability m suppresses distraction by penalising any pixel in the overlapping area. The penalty factor prevents many ellipses form collapsing into a single target at an early stage of interaction. The integral in Eq15 can be written as a sum in Eq18, where $\frac{2}{\pi ab}$ is a normalizing factor.

$$\lambda^s_{k,n} = \frac{2}{\pi ab} \sum_{p;\varepsilon<\varphi_{k,n}(p)<1} m_{k,n}\varphi_{k,n}F \quad (18)$$

The likelihood of mutual overlapping subjects is reduced by the penalty factor. The use of the penalty provides distraction suppression without joint state vectors of all subjects. The penalty prevents distraction as similar to a joint likelihood approximation while tracker are overlapping and returns independent likelihood in well separated targets. The approach described here returns high accuracy with a fixed number of particles.

2.3.6 Texture Estimation and Likelihood

Our texture model is a 2D image wrapped around the ellipsoid surface of the state vector. Each pixel on the image contains colour distribution. Given the camera parameters and estimated state, we can map from pixel p on the image plane to a particular surface coordinate q. The ellipsoid is scaled to a unit sphere and a point on this sphere is expressed in coordinates $q = (\theta, z)$, as this generates a uniform distribution (Weisstein, 1999), where z is the vertical axis and θ is the azimuth. The $q : (\theta, z)$ can be mapped to an observation image coordinate $p = (p_x, p_y)$. In order to register an observed subject to the textures surface, we assume the facing angle is the direction of subject movement.

Each texture pixel model, $T(q)$, has a multivariate 3D Gaussian distribution. $T(q) = (\mu, \Sigma, \omega)$, where μ, Σ and ω are the mean of colour vector, covariance matrix and confidence, respectively. To estimate the pixel values, each observation I(p) is used to recursively update the texture model T(q). First, p is mapped to q as described in section 2.3.3. Second, $I(p)$ is tested by the chi-square test given model $T(q)$ as the colour distribution. At p-value 0.01 the confidence interval of matching between $I(p)$ and $T(q)$ is $Q < 11.34$, where

$$Q = (I_p - \mu)^{\mathrm{T}} \Sigma^{-1} (I_p - \mu), \tag{19}$$

I_p is the observation vector at p. If the observation is in the confidence interval, the confidence ω is increased, updating the texture model $T(q)$ as Eq 20-22. If beyond the confidence interval, ω is decreased and T(q) is maintained. When ω drops below a threshold the texture model is reset by setting the mean to a mid value with large covariance.

$$d_q = \mu_q - I_p \tag{20}$$
$$\mu_{q,t} = (\rho - 1)\mu_{q,t-1} + \rho d_q \tag{21}$$
$$\Sigma_{q,t} = (\rho - 1)\Sigma_{q,t-1} + \rho d_q^{\mathrm{T}} d_q + c \tag{22}$$

Perspective projection means that q may be mapped to many p. The number of observation pixels per texture model depends linearly on the size of the ellipse. To avoid over fitting, the learning rate ρ is scaled down by the size of ellipse. Hence, the new leaning rate is $\rho = \frac{\rho'}{ab}$,where ρ' is the original learning rate and a, b are the ellipse radii. Each tracker k has a separate texture model T_k.

During tracking, to compute the texture likelihood, we classify an observation pixel as matched to the texture model or not by checking the Q value, where $Q < 2.37$ is a matching threshold at p-value 0.5. Any matched observation contributes to the log-likelihood λ^t.

$$\lambda_k^t = \frac{1}{\pi ab} \sum_{p;\varphi_k(p)>0} D(p) \tag{23}$$
$$D(p) = \begin{cases} 1 & \text{if } Q < 2.37 \\ 0 & \text{otherwise} \end{cases} \tag{24}$$

The final likelihood w for a single camera c is a function of both silhouette and texture likelihood as in Eq25. At the early stages of tracking we have an inadequate information of texture. The sensitivity α_t is set to 0 to disable texture likelihood and it will be enabled when reliable texture is obtained by checking the covariance of the texture model is getting smaller than a threshold.

$$w_{c,k,n} = \exp[-\alpha_s(\lambda_{k,n}^s - 1)^2 + \alpha_t \lambda_{k,n}^t] \tag{25}$$

Figure 6: The posterior density of position is represented by particles on ground plane and the red ellipse shows the estimated result. The mean of texture obtained during tracking, shown in the top-left panel, is mapped to the ellipsoid surface.

2.4 Combining Data from Multiple Cameras

Thus far, we have described the tracking framework form the perspective of a single camera. In our network model, a subject can be seen by many cameras so we need to combine the likelihood values from all

the cameras into single global likelihood. The cameras are installed at different places and their observable areas may or may not overlap. A likelihood weight $w_{c,k,n}$ from particle n of tracker k of camera c may possibly be zero due to the target being outside the detectable zone or occlusion. So, validating each camera likelihood must be done before combining these to compute a global likelihood.

Visibility on a subject can be measured form the average weight $\bar{w}_{c,k}$ of all particles. The average weight of single subject must be greater than a threshold α_w All trusted measurements are combined into global likelihood $W_{k,n}$.

$$W_{k,n} = \prod_{\forall c} w^*_{c,k,n} \tag{26}$$

where $w^*_{c,k,n}$ is filtered weight determined from confidence in likelihood measurement.

$$w^*_{c,k,n} = \begin{cases} w_{c,k,n} & \text{if } \bar{w}_{c,k} > \alpha_w \\ 1 & \text{otherwise} \end{cases} \tag{27}$$

$$\bar{w}_{c,k} = \frac{1}{N} \sum_{\forall n} w_{c,k,n} \tag{28}$$

2.5 Removing a Tracker

A tracker may be discontinued for three reasons:

- if the persistence is zero because a target has not reappeared in an expected time;
- if the variance of the state distribution is very high, which happens when the tracker is distracted by another target or there is lack of observation. This leads to high uncertainty of estimation so we withdraw the tracker;
- two or more trackers may be distracted and follow the same target. The target which has smaller likelihood will be terminated. Multiple trackers following a single target can be detected by calculating the distance between the expected states.

3 IMPLEMENTATION

The particle filter approach has high complexity which may be mitigated by using multiple processors. Alternatives include multiple nodes on a cluster, multiple cores on a single platform or specialised hardware. A very promising technology for image processing is found in graphics processing units (GPUs) in standard grahics cards. For example in (Boyer et al., 2009; Zechner and Granitzer, 2009) a graphic card provides additional speed up from 10 to 50 times. This technology requires a developer to design an algorithm in parallel to achieve full utilization.

We have implemented our system on a graphic processing unit (GPU) GTS250 from NVIDIA using CUDA tools for programming and optimization. The GPU has 16 stream multiprocessors (SMs), where each SM contains 8 stream processors. Overall, there are 128 cores able to process concurrently and each core has a 1.62GHz clock frequency. The cores are controlled by the host CPU. A set of instructions is transfered from the CPU to GPU and then all the cores produce threds to process the same instruction with different input data, in the SIMT (single instruction multi thread) model. We have implemented our system using CUDA (NVIDIA, 2010), a C like language for graphic card programming.

The tracking algorithm in Fig.1 can be divided into a sequence of functions as in Fig.7. A function is then divided into smaller pieces based on the number of output elements. Each output element is saved to global memory on the GPU and waits for the next execution cycle. In the implementation we also use texture and restrict memory to accelerate data transfer.

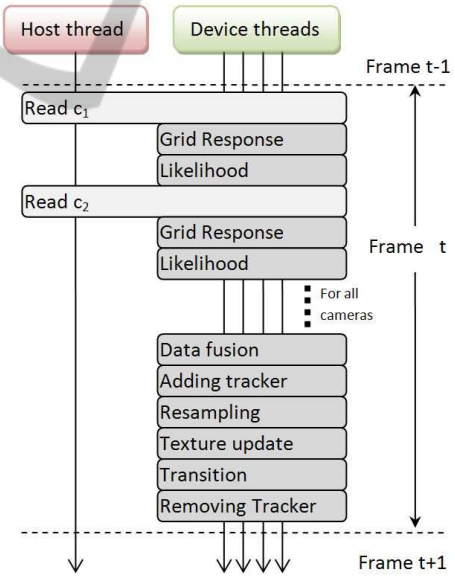

Figure 7: Main loop sequence diagram.

4 EXPERIMENT AND RESULT

We have tested the system in both off-line and in real-time modes. In off-line mode, images were saved to the hard-drive and loaded frame-by-frame while performing tracking.

The result was evaluated by MOTA (Bernardin

and Stiefelhagen, 2008) (Multiple object tracking accuracy), a standard metric for multi-target tracking,

$$\text{MOTA} = 1 - \frac{\sum_t (f_t + m_t + s_t)}{\sum_t g_t} \quad (29)$$

,where f_t, m_t and s_t are the number of false alarms, miss detection and switch-id in frame t and g_t is the total number of targets in the scene at frame t. Specifically, miss detection is when a target appears in one or more camera views but the system fails to detect the target. A False alarm is when a tracker is active but there is no target associated with the tracker. Switch-ID is an event when a target gets a new tracker ID resulting from losing the ability to track (excluding miss detection). Causes of Switch-ID are false-alarm, distraction, reappearance or occlusion.

4.1 PETS09 Results

The PETS09 dataset (Ferryman and Shahrokni, 2009) provides camera parameters and image sequences captured from 7 different cameras. We chose the first two views to test our algorithm. The image resolution is 768×576 pixels. To study the effect of texture likelihood we tested the algorithm with likelihood disabled and enabled. the number of particles is fixed at 512. The experiment was repeated 10 times with different random seed numbers. Table 1 shows that the result of tracking without texture information has an average MOTA of 86.5%. While using texture the average MOTA is 90.7%.

Table 3 shows time profiling on particular functions. From the table we can see significant speedup in Preprocessing, GridResponse, AddTracker and Likelihood. In particular, the acceleration of the Preprocess function is from 514ms to 16ms, giving around 32 times speed up. This is because the process can be divided into a large number of independent pixel-wise tasks. During the Preprocess the utilization of cores is restricted to 40%. Using the *CUDA Occupancy Calculator* identifies a problem of lack of local shared memory and registers. We could improve utilization by dividing the process into further successive sub-functions.

In the Likelihood function we have 512 particles per tracker and need about 8KB of memory. By splitting the computation according to the number of particles we can also reduce the computation time from 346ms to 32ms. Processor utilization is about 17%, due to using local memory. The occupancy is fairly small. However large numbers of cores (128 cores) can improve the speed of the Likelihood about 10 times.

Overall, there is around 10 times speedup for the whole process.

Table 1: Result from tracking without texture likelihood.

f(%)	m(%)	s(%)	MOTA(%)
7.16±5.33	5.83±2.60	0.46±0.37	86.5

Table 2: Result when texture likelihood is enabled.

f(%)	m(%)	s(%)	MOTA(%)
5.20±3.64	3.76±1.48	0.26±0.14	90.7

Table 3: Comparison of average subtotal time for single time frame between CPU and GPU.

Function(call)	CPU time	GPU time	Speed-up	Util (%)
Read(2)	36.0	40.0	0.9	-
Preprocess(2)	514.0	16.0	32.1	40
GridResponse(2)	60.0	3.6	16.7	33
Likelihood(2)	346.0	32.0	10.8	17
DataFusion(2)	4.0	1.0	4.0	4
AddTracker	45.0	2.7	16.7	33
Resample	4.0	0.5	8.0	4
TextureUpdate(2)	70.0	7.6	9.2	20
Transition	5.0	3.3	1.5	4
RemoveTracker	1.0	1.0	1.0	10
Total time (ms)	1085.0	107.7	10.1	

4.2 Real-time Test Results

For on-line testing we have used 3 Point Grey Flea2 cameras connected to the computer via a 1394b hub, which is installed on a PCI Express slot on the host platform. This configuration can capture images from different views in synchrony. All images pass directly from the host to the GPU at a frame rate of 7.5 multi-views per second. All cameras are static and well calibrated. The resulting MOTA is about 90% which is the same as for the PETS09 data set. Some result images are shown in Fig9.

5 CONCLUSIONS AND FUTURE WORK

Our algorithm provides a tracking accuracy MOTA of about 90%, offering improvements on other methods e.g. 75% in (Berclaz et al., 2009) and 80% in (Breitenstein et al., 2010). From our results, augmenting the image with texture information reduces the error rate from 13.5% to 9.3%.

The penalty function makes a hybrid estimation using independent and joint posterior densities. The estimation suppresses mutual distraction resulting in

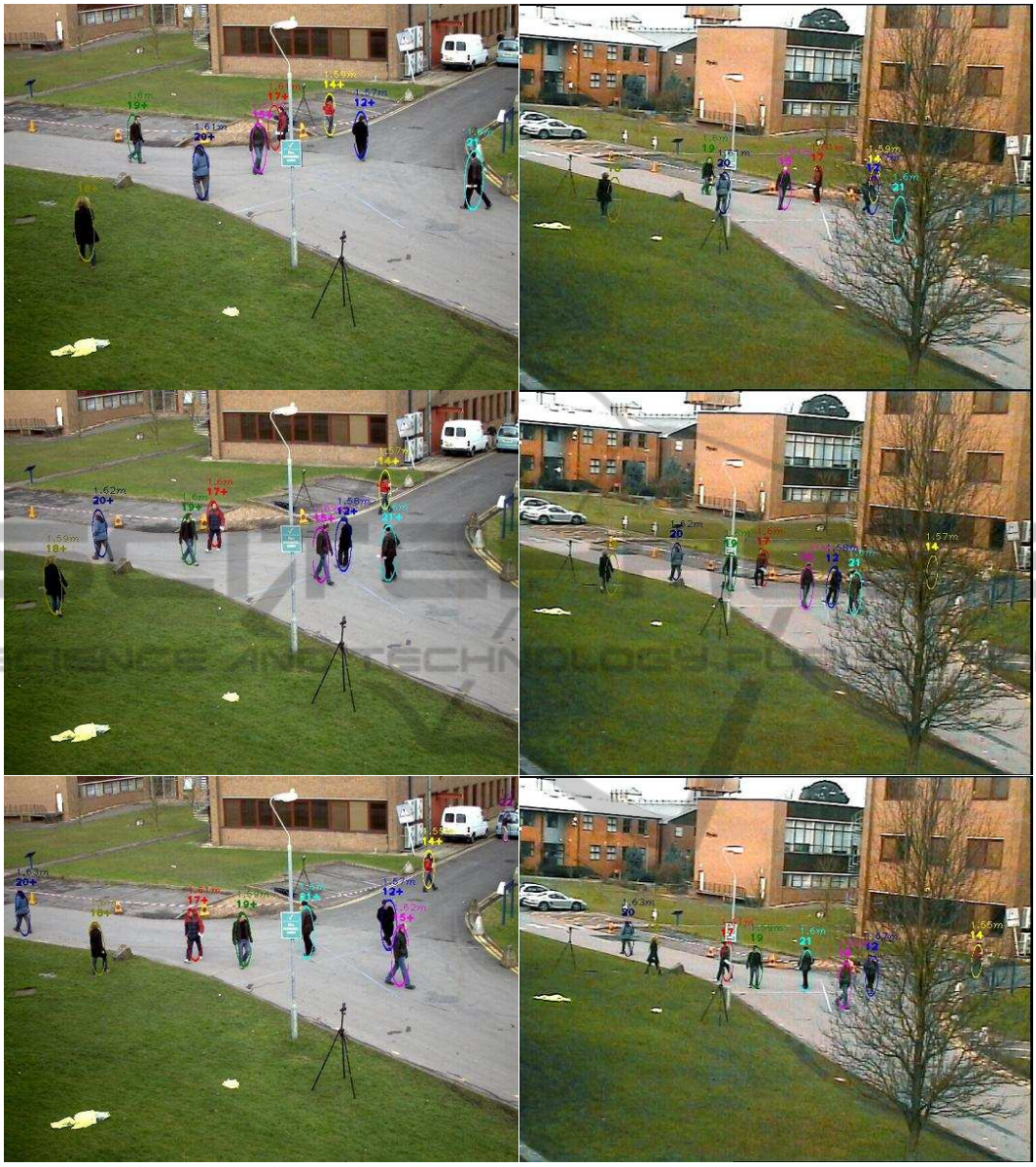

Figure 8: Evaluating with PET09 dataset. First and second columns are result sequences from camera1 and camera3, from top to bottom are frame 700, 720 and 740, respectively. The numbers over ellipse show ID and height in unit meter.

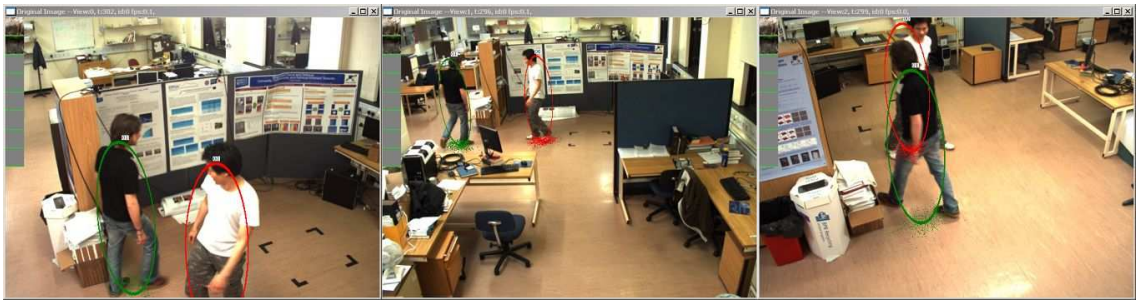

Figure 9: The ellipses show estimated positions projected on each view and panel on top-left are texture images of the subjects.

improved accuracy in the early tracking stage.

We have gained significant speedup by deploying parallel processing on the GPU, from 1 frame per second to 10 frames per second which is broadly comparable to current standard CCTV installations. This allows near-real-time tracking of large numbers of objects.

We will next investigate further improvements to our implementation to optimise thread occupancy on the GPU. In the longer term we will explore real time object tracking between non-overlapping cameras where we think our texture approach will improve object handover.

REFERENCES

An, K. H. and Chung, M. J. (2008). 3d head tracking and pose-robust 2d texture map-based face recognition using a simple ellipsoid model. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems IROS 2008*, pages 307–312.

Bardet, F. and Chateau, T. (2008). Mcmc particle filter for real-time visual tracking of vehicles. In *Proc. 11th International IEEE Conference on Intelligent Transportation Systems ITSC 2008*, pages 539–544.

Berclaz, J., Fleuret, F., and Fua, P. (2009). Multiple object tracking using flow linear programming. In *Proc. Twelfth IEEE Int Performance Evaluation of Tracking and Surveillance (PETS-Winter) Workshop*, pages 1–8.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: the clear mot metrics. *J. Image Video Process.*, 2008:1–10.

Boyer, M., Tarjan, D., Acton, S. T., and Skadron, K. (2009). Accelerating leukocyte tracking using cuda: A case study in leveraging manycore coprocessors. In *Proc. IEEE International Symposium on Parallel & Distributed Processing IPDPS 2009*, pages 1–12.

Breitenstein, M., Reichlin, F., Leibe, B., Koller-Meier, E., and Van Gool, L. (2010). Online multiperson tracking-by-detection from a single, uncalibrated camera. *PAMI*. Early Access.

del Blanco, C. R., Mohedano, R., Garcia, N., Salgado, L., and Jaureguizar, F. (2008). Color-based 3d particle filtering for robust tracking in heterogeneous environments. In *Proc. Second ACM/IEEE International Conference on Distributed Smart Cameras ICDSC 2008*, pages 1–10.

Derpanis, K. G. (2007). Integral image-based representations. *Department of Computer Science and Engineering York University Paper*.

Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, volume 2, pages 126–133.

Eberly, D. (1999). Perspective projection of an ellipse. www.geometrictools.com.

Ferryman, J. and Shahrokni, A. (2009). Pets2009: Dataset and challenge. In *Proc. Twelfth IEEE Int Performance Evaluation of Tracking and Surveillance (PETS-Winter) Workshop*, pages 1–6.

Husz, Z. L., Wallace, A. M., and Green, P. R. (2011). Tracking with a hierarchical partitioned particle filter and movement modelling. *SMCB*, pages 1–14. Early Access.

Isard, M. and Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume I*, pages 343–356, London, UK. Springer-Verlag.

Jaward, M., Mihaylova, L., Canagarajah, N., and Bull, D. (2006). Multiple object tracking using particle filters. In *Aerospace Conference, 2006 IEEE*, page 8 pp.

Kreucher, C., Kastella, K., and Hero, A. O. (2005). Multitarget tracking using the joint multitarget probability density. *AES*, 41(4):1396–1414.

NVIDIA (2010). Nvidia cuda c best practice guide.

Peursum, P., Venkatesh, S., and West, G. (2007). Tracking-as-recognition for articulated full-body human motion analysis. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR '07*, pages 1–8.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyound the Kalman Filter Particle Filter for Tracking Application*. DSTO.

Schneider, P. J. and Eberly, D. H. (2003). *Geometric Tools for Computer Graphics*. Elsevier.

Sidenbladh, H., De la Torre, F., and Black, M. J. (2000). A framework for modeling the appearance of 3d articulated figures. In *Proc. Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 368–375.

Stauffer, C. and Grimson, W. E. L. (1999). Adaptive background mixture models for real time tracking. In *Proc. IEEE Computer Society Conference on. Computer Vision and Pattern Recognition*, volume 2, page 252 Vol. 2.

Vezzani, R., Grana, C., and Cucchiara, R. (2011). Probabilistic people tracking with appearance models and occlusion classification: The AD-HOC system. *Pattern Recognition Letters*, 32(6):867–877.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2001*, volume 1.

Weisstein, E. (1999). Sphere point picking. From MathWorld–A Wolfram Web Resource.

Zechner, M. and Granitzer, M. (2009). Accelerating k-means on the graphics processor via cuda. In *Proc. First International Conference on Intensive Applications and Services INTENSIVE '09*, pages 7–15.