Real Time Parallel Implementation of Particles Filter Based Visual Tracking

Joel Falcou, Thierry Chateau, Jocelyn Sérot, and Jean-Thierry Lapresté

LAsSMEA, UMR6602, CNRS, Blaise Pascal University,Clermont-Ferrand, France
{Surname.NAME}@lasmea.univ-bpclermont.fr

1 Introduction

Particle filtering is a widely used method to solve vision tracking problems. However, to be able to run in real-time on standard architecture, the state vector used in the particle filter must remain small [1]. We propose a parallel implementation of a 3D tracking algorithm operating on a stereo video stream and running in real-time on a cluster architecture. We demonstrate the efficiency of this implementation with a pedestrian tracking application.

2 Principle of the Method

2.1 Particle Filter

Particle filtering [2] is a sequential importance sampling algorithm for estimating properties of hidden variables given observations in a hidden Markov model. Standard particle filter assumes that posterior $P(\mathbf{X}_t|\mathbf{Z}_t)$ can be approximated by a set of samples (particles). Moreover it also assumes that the observation likelihood $P(\mathbf{Z}_t|\mathbf{X}_t)$ can be easily evaluated. A particle filter approximates the posterior using a weighted particle set $\{(\mathbf{X}_t^n, \pi_t^n) : n = 1, .., N\}$.

2.2 State Space and Dynamics

We want to track an object in a 3D space defined in a reference frame R_w. Left and right camera projection matrices between R_w and the image plane are given by $\mathtt{C}_l$ and $\mathtt{C}_r$. At time t, the state vector is defined by $\mathbf{X}_t \doteq (\mathbf{P}_t, \mathbf{V}_t)^t$, where $\mathbf{P}_t$ is the 3D position of the center of a bounding box associated with the object to be tracked and $\mathbf{V}_t$ is the associated velocity. For a state $\mathbf{X}_t$, the corresponding 2D points $\mathbf{p}_t^l$ and $\mathbf{p}_t^r$ of the center of an image bounding box for left and right camera are given by : $\left((\mathbf{p}_t^{(l,r)})^t 1\right)^t \propto \mathtt{C}_{(l,r)} \left((\mathbf{P}_t^{(l,r)})^t 1\right)^t$. Since height and width of the 3D bounding box are assumed to be constant, the corresponding height and width of each image bounding box is computed using projections matrices. The dynamic model of the system $(p(\mathbf{X}_t|\mathbf{X}_{t-1}))$ is given by $\mathbf{X}_{t+1} = \mathtt{A}\mathbf{X}_t + \mathtt{B}\mathbf{v}_t$, $\mathbf{v}_t \sim \mathcal{N}(0, \Sigma)$, where matrix $\mathtt{A}$ can be learnt or can be set using a constant position or constant velocity model, and matrices $\mathtt{B}$ and Σ can be estimated from a set of sequences for

which the position of the object is known. To initialize the system, we constrain the initial 3D position of the tracked object to a 3D bounding box and discretize this space at a fine resolution[1]. Particle filter is then initialized with the N particles associated to the N best weights.

2.3 Observation Likelihood

This section describes the tracker likelihood function $P(\mathbf{Z}_t|\mathbf{X}_t)$ which is defined as the likelihood that the state of the object is $\mathbf{X}_t$ according to an observed couple of images $\mathbf{Z}_t$. We propose to use the output of an Adaboost based Classifier [3] with multiscale Haar wavelets descriptors [4]. A compact description of the object is selected from the Adaboost offline learning step [5]. This classifier $\mathbf{s}_t(\mathbf{X}_t) = (s_t^l(\mathbf{X}_t), s_t^r(\mathbf{X}_t))^t$ returns an uncalibrated vector [2] of values for the input 3D state $\mathbf{X}_t$. We propose to build the likelihood function used to evaluate weights of the particle filter from $\mathbf{s}_t(\mathbf{X}_t)$. Since the likelihood function used by the particle filter is a probability, $P(class|input)$ must be produced from the output of the classifier. A sigmoid is used to build calibrated probabilities from $\mathbf{s}_t(\mathbf{X}_t)$ [6] :

$$P(\mathbf{Z}_t|\mathbf{X}_t) \doteq \frac{1}{1+\exp(A.s_t^r(\mathbf{X}_t)+B)}.\frac{1}{1+\exp(A.s_t^l(\mathbf{X}_t)+B)} \quad (1)$$

3 Parallel Implementation

3.1 Architecture Synopsis

Our cluster architecture [7] includes fourteen compute nodes. Each node is a dual-processor Apple G5 XServe Cluster Node running at 2 GHz with 1Gb of memory. Nodes are interconnected with Gigabit Ethernet and provided with digital video streams, coming from a pair of digital cameras, by a *Firewire IEEE1394a* bus. This approach allows simultaneous and synchronized broadcasting of input images to all nodes, thus removing the classical bottleneck which occurs when images are acquired on a dedicated node and then explicitly broadcasted to all other nodes. Programming relies on a hybrid three-level parallel programming model, involving a fine-grain SIMD-type parallelism within each processor [8], a coarse grain shared-memory multi-processing model between the two processors of a node (using pThread) and a coarse grain message passing based multi-processing between two processors of distinct nodes (using MPI).

3.2 Parallelisation Strategy

We use a pure data-parallel strategy, in which the particles distribution $\{(\mathbf{X}_t^n, \pi_t^n) : n = 1, .., N\}$ is scattered among the compute nodes. Each node therefore performs the prediction, measure and weighting steps on a subset of the initial

[1] typically around 1cm

[2] two values corresponding to classifier score associated to left and right camera

particule distribution. On each node the left and right projections and measures are themselves computed in parallel at the SMP level, each by one processor. Moreover, on each processor, computations are vectorized whenever possible at the SIMD level. A final merging step is used to update $\mathbf{X}_t$.

4 Results

Table 1 shows results obtained on 640×480×8bits video streams for several size of particles distribution. Framerate is given for a single processor machine and compared to a 2,8 and 14 nodes topology.

	200 part.	1000 part.	2000 part.	6000 part.
Sequential ref.	26.94 FPS	6.09 FPS	2.44 FPS	0.91 FPS
1 Node	48.5 FPS	10.9 FPS	4.6 FPS	1.6 FPS
2 Nodes	66.01 FPS	14.37 FPS	6.42 FPS	2.22 FPS
8 Nodes	53.19 FPS	21.86 FPS	9.71 FPS	2.99 FPS
14 Nodes	35.33 FPS	23.18 FPS	11.29 FPS	3.15 FPS

These are preliminary results and further experiments will assess the relation between the number of nodes, the size of the distribution and the execution time but we may already notice that near real-time performances are achievable with a low number of nodes : a 17 frames per second rate is achieved by using approximatively 1500 particles scattered on 14 nodes.

References

1. Granmo, O.C., Eliassen, F., Lysne, O., Eide, V.S.: Techniques for parallel execution of the particle filter. In: Proceedings of Scandinavian Conference on Image Analysis (SCIA 2003). Lecture Notes in Computer Science, Springer-Verlag (2003)
2. Arulampalam, S., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing **50** (2002) 174–188
3. Viola, P., Jones, M.J., Snow, D.: Detecting pedestrians using patterns of motion and appearance. In: Int. Conf. Computer Vision, Nice, France (2003) 734–741
4. C. Papageorgiou, M. Oren, T. Poggio: A general framework for object detection. In: IEEE Conference on Computer Vision. (1998) 555–562
5. Tieu, K., Viola, P.: Boosting image retrieval. International Journal of Computer Vision **56** (2004) 17–36
6. Niculescu-Mizil, A., Caruana, R.: Obtaining calibrated probabilities from boosting. In: Proc. 21st Conference on Uncertainty in Artificial Intelligence (UAI '05), AUAI Press (2005)
7. J. Falcou, J. Serot, T. Chateau, F. Jurie: A Parallel Implementation of a 3D Reconstruction Algorithm for Real-Time Vision. Parallel Computing 2005 (2005)
8. J. Falcou, J. Serot: E.V.E., An Object Oriented SIMD Library. Scalable Computing: Practice and Experience **6** (2005)