Accelerating particle filter-based object tracking algorithms using parallel programming

Mai Thanh Nhat Truong*, Sanghoon Kim*
*Department of Electrical, Electronic, and Control Engineering, Hankyong National University

Abstract

Object tracking is a common task in computer vision, an essential part of various vision-based applications. After several years of development, object tracking in video is still a challenging problem because of various visual properties of objects and surrounding environment. Particle filter is a well-known technique among common approaches, has been proven its effectiveness in dealing with difficulties in object tracking. However, particle filter is a high-complexity algorithms, which is an severe disadvantage because object tracking algorithms are required to run in real time. In this research, we utilize parallel programming to accelerate particle filter-based object tracking algorithms. Experimental results showed that our approach reduced the execution time significantly.

1. Introduction

Object tracking has important roles in many vision-based applications, such as industrial inspection, human-computer interfaces, traffic monitoring, surveillance systems. Generally, the goal of object tracking is locating and producing a trajectory record of the objects in image sequences. Among common approaches for object tracking, particle filter is a robust solution. Particle filter has been proven to produce high performance when applied to nonlinear and non-Gaussian estimation problems [1]. This method approximates a posterior probability density of the state, in this case it is the positions of the object in image sequence. The approximation is done by using point mass representations of probability densities, which are called particles. Usually contours, color features, or appearance models are used as particles when applying particle filter to object tracking [1-4]. The color histogram is often used because of its robustness against noise and occlusion, but suffers from illumination changes. This disadvantage can be overcome by using other color spaces which are less sensitive to light conditions.

Although particle filter have been commonly used in recent years, they have considerable disadvantages [5]. One of those is sampling impoverishment. In this phenomenon, high-weight particles have higher chance to be drawn multiple times during resampling, whereas low-weight particles have low chance to be drawn at all. This makes the diversity of the particles decrease after several resampling steps. In the worst case scenario, all particles might be “merged” into a single particle. Sampling impoverishment reduces the performance of tracking process drastically. Another disadvantage of particle filter is computational complexity. The complexity increases as the area of tracked region and the number of particles increase. When the dimensionality of the state space increases, the number of particles required for the sampling increases exponentially.

This research presents a method for utilizing modern multi-core computer systems and parallel programming technique in object tracking tasks. We use parallel programming to implement particle filter algorithm, for the purpose of increasing the performance of particle filter based object tracking method. In the following sections, we provide details for the parallel implementation of object tracking system and experimental results.

2. Object tracking system

2.1 Particle filter

The particle filter in this research, which is developed to track movement of objects, is based on Condensation algorithm [1]. Let X_t be the state vector which describes the quantities of a tracked object, and Z_t be the vector which stores all the observations of object movements $\{z_1, \dots, z_t\}$ up to time t. Usually, the posterior density $p(X_t \mid Z_t)$ and the observation density $p(Z_t \mid X_t)$ are non-Gaussian, which increases the complexity of tracking process.

In particle filter algorithm, the probability distribution, which presents the object movements, is approximated by a weighted particle set $S = \{(s_n, \pi_n) \mid n = 1 \dots N\}$. For each particle, element s represents the hypothetical state, i.e. location, of the tracked object, and a corresponding discrete sampling probability π. The movement of the tracked objects is described by a statistical model. In tracking process, each particle is weighted depending on observations, then N particles are drawn using resampling techniques. The estimation of mean state is calculated at each time step by:

$$E[S] = \sum_{n=1}^{N} \pi_n s_n \qquad (1)$$

Because particle filters have ability to model the uncertainty of object movements, it can provide a robust tracking framework. It can consider multiple state hypotheses simultaneously. On the other hand, particle filters are able to produce high accuracy prediction from previous observations, hence it can deal with short-time occlusions and sudden changes in object movements. In this research we use color models for likelihood calculation. We combine particle filter with color model by integrating local color distribution into particles. The model is based on [6].

2.2 Parallel implementation

Parallelization can utilize the multi-core architecture of modern embedded systems, in which all cores of the processor take part in the calculation process. Parallel implementation can decrease the processing time of these steps, which leads to higher performance due to higher processed frames per second. However, parallelization has its own limitation. Generally in parallel computing, a task is split up into several threads, then the split tasks will be solved separately in each thread. After completing the computation, those threads will communicate with each other to produce final results. In some cases, the time required for communication is higher than the time of solving split tasks. Overall execution time of parallel implementation in these cases may be higher than sequential implementation. This phenomenon is called parallel slowdown.

For the purpose of avoiding parallel slowdown, we only use parallel implementation for resampling particles and calculating likelihood between particles and target object. The detailed algorithms for these steps are listed below. Since there are no data dependencies between particles in these algorithms, parallel threads do not need to communicate with each other in processing. For example, in resampling step, `newArr(1)`, `newArr(2)`, `newArr(3)`, and `newArr(4)` can be calculated at the same time using four threads and so on, hence reduce the execution time of whole process.

```
Algorithm 1. Systematic resampling.
Create newArr for storing output of resampling process
parallel-for i = 1 to (number of particle)
    j ← 0
    while (probabilities of particle(j)) < (likelihood i with target object)
        j ← j + 1
    endwhile
    newArr(i) ← particle(j)
endfor
```

```
Algorithm 2. Likelihood calculation.
parallel-for i = 1 to (number_of_particle)
    Get image region of particle i
    Calculate likelihood between obtained image region and target object
endfor
```

3. Experiments

In this section, we evaluate the performance of parallel programming. Both versions of particle filter algorithm, sequential and parallel, are implemented in C++ under Linux operating system. OpenCV library is used for processing video frames. For parallelization, we use OpenMP feature of GNU compiler. The system has 8GB of RAM and a quad-core Intel CPU running at 3.0GHz. The video used for testing is acquired from [7]. It has 630 frame, and a resolution of 320x240. The task is to track a cup which was moving in front of a wall with complex textures, as shown in Figure 1.

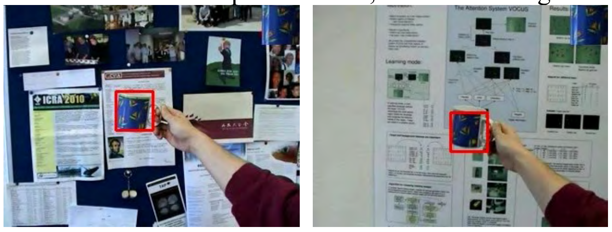
Figure 1. The tracking task in experiment.

We performed the tracking task with different numbers of particles, ranging from 500 to 2500 particles. The comparison in execution time between parallel programming and sequential programming is illustrated in Figure 2, lower is better. As can be seen, at 500 particles, parallel implementation is not much better than sequential implementation because of parallel slowdown. However, when the number of particles increased, parallel implementation showed its advantage over sequential implementation.

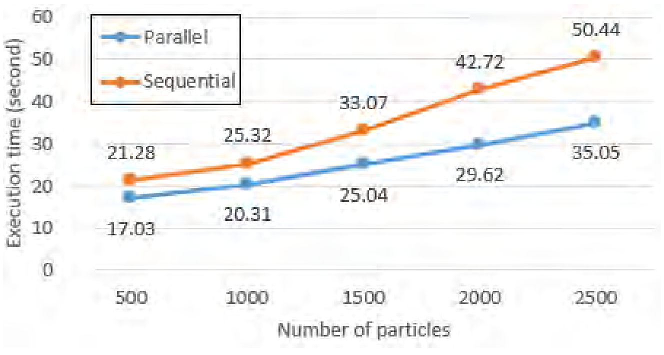

Figure 2. Comparison in execution time.

4. Conclusions

This research aims to increase the performance of particle filter-based object tracking method by using parallel programming. Particle filter is a well-known technique among common approaches, has been proven its effectiveness in dealing with difficulties in object tracking. However, particle filter is a high-complexity algorithms, which is an severe disadvantage because object tracking algorithms are required to run in real time. In this research, we have implemented parallel particle filter for object tracking in order to utilize the multi-core architecture. The experimental results show that multi-core systems can produce higher performance if the hardware is used at its maximum potential.

5. Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(2015R1D1A1A01057518).

References

[1] Michael Isard, Andrew Blake, Condensation - conditional density propagation for visual tracking, International Journal of Computer Vision 29 (1998) 5–28.
[2] Michael Isard, Andrew Blake, ICondensation: Unifying low-level and high-level tracking in a stochastic framework. In Proc. European Conf. Computer Vision, pages 893–908, 1998.
[3] John MacCormick, Andrew Blake, A probabilistic exclusion principle for tracking multiple objects. In Proc. Int'l Conf. Computer Vision, pages 572–578, 1999.
[4] Ying Wu, Robust visual tracking by integrating multiple cues based on co-inference learning, International Journal of Computer Vision 58 (2004) 55–71.
[5] King O., Forsyth D., How does condensation behave with a finite number of samples?, Proceedings of the 6th European Conference on Computer Vision, pages 695–709, 2000.
[6] Katja Nummiaro, Esther Koller-Meier, Luc Van Gool, Object Tracking with an Adaptive Color-Based Particle Filter, Lecture Notes in Computer Science, Pattern Recognition 2449 (2002) 353–360.
[7] Tracking Dataset. (ONLINE) Available at: http://cmp.felk.cvut.cz/~vojirtom/dataset/tv77/. (Accessed 01 March 2018).