Confidence-driven Architecture for Real-time Vision Processing and Its Application to Efficient Vision-based Human Motion Sensing

Yoshimoto, Hiromasa
Department of Intelligent Systems, Kyushu University

Date, Naoto
Department of Intelligent Systems, Kyushu University

Arita, Daisaku
Department of Intelligent Systems, Kyushu University

Taniguchi, Rin-ichiro
Department of Intelligent Systems, Kyushu University

https://hdl.handle.net/2324/5979

出版情報：Proceedings of the International Conference on Pattern Recognition. 1, pp.736-740, 2004-08
バージョン：
権利関係：

Confidence-driven Architecture for Real-time Vision Processing and Its Application to Efficient Vision-based Human Motion Sensing

Hiromasa Yoshimoto, Naoto Date, Daisaku Arita, Rin-ichiro Taniguchi
Department of Intelligent Systems,Kyushu University, Japan
{yosimoto,naoto,arita,rin}@limu.is.kyushu-u.ac.jp

Abstract

In this paper, we discuss a real-time vision architecture which provides a mechanism of controlling trade-off between the accuracy and the latency of vision systems. In vision systems, to acquire accurate information from input-images, the huge amount of computation power is usually required. On the other hand, to realize real-time processing, we must reduce the latency. Therefore, under given hardware resources, we must make difficult trade-off between the accuracy and the latency so that the quality of the system's output keeps appropriate. To solve the problem, we propose confidence-driven scheme, which enables us to control the trade-off dynamically and easily without rebuilding vision systems. In the confidence-driven architecture, the trade-off can be controlled by specifying a generalized parameter called confidence, which relatively indicates how accurate the analysis should be. Here, we present the concept of confidence-driven architecture, and then, we show a shared memory which uses confidence-driven scheme. Using confidence-driven memory, we can use imprecise computation model to reduce the latency without a large decrease of accuracy.

1. Introduction

Using computer vision techniques, we can acquire various information of the real world such as object types, or the position and pose of the object. In general, to increase the accuracy of the acquired information, we need complex vision algorithms, which leads to increase of computation time. On the other hand, for vision applications requiring real-time analysis, the latency, or time required to obtain the processed result, is of critical importance. Thus, we have to make difficult trade-off between the accuracy and the latency. Since the trade-off depends on situation where the vision system works, from a practical point of view, a computational technique is required in which we can easily and dynamically control the trade-off without changing an application program itself.

In principle, such computation technique can be formulated as imprecise computation model[1], in which the computation accuracy increases as allowable computation time increases. Its important feature is that even uncompleted computation produces meaningful results, which can be used for further computation. Based on this feature, we can solve the trade-off between the accuracy and the latency. However, no general software framework to realize the imprecise computation has been established. Here, to solve this problem, and to provide a general software framework for real-time vision processing based on the imprecise computation model we have proposed confidence-driven architecture.

In this paper, we outline the confidence-driven architecture and then we present its application to a real-time vision-based human motion sensing system. In this application, prediction mechanisms built in confidence-driven memories, which are implementation of the confidence-driven architecture, make the system less computational and make the system have less latency without a large decrease of accuracy. Although prediction-based frameworks such as Dynamic Memory[2] and Dead Reckoning[3] provide similar latency-free features, they do not take account of the accuracy of the computation. The important point of the confidence-driven architecture is that depending on system environments and goals, we can dynamically and easily control the accuracy-latency trade-off without changing the system softwares.

2. Confidence-driven Architecture

2.1. Latency vs Accuracy

Any vision systems contain a difficult trade-off: a large computation is required to get accurate information from an image; on the other hand, fast computation is desired to acquire information quickly. The trade-off is influenced by many factors: whether a user assigns high priority to higher

0-7695-2128-2/04 $20.00 (C) 2004 IEEE

precision or to lower latency; how many computational resources such as computation time and bandwidth of the network are available. We have to make difficult trade-off between the accuracy and the latency in dynamically so that the quality of the system's output keeps appropriate.

For example, when we detect faces in an input image, we have several options such as (1) searching for a skin color region with a certain size (2) searching for a region including sub-regions of eyes and a mouth with appropriate positional relations. (1) is faster but less accurate and (2) is slower but more accurate. We have to select an appropriate method depending on the situation where the system works. In another aspect, when we deal with real time image sequences, we can often assume that a target, or a face, moves continuously, or that the target position is a continuous variable. Then, reduction of the redundancy which the variable implicitly has can lead to decrease of the computation time. Trade-off here is to determine the ratio of the use of the estimator to that of execution of a vision algorithm for face searching in an image. Since the trade-off depends on the situation, we require a software framework in which we can easily control the trade-off without changing an application program itself.

2.2. Confidence-driven Architecture

It is important that both of the above examples, the face searching and the estimator, have the same characteristic: as the time increases the accuracy of the output becomes higher monotonically. The monotonic characteristic is a basis of imprecise computation model[1]. If a precise relation between allocated computation time and the accuracy are known, we can control the trade-off according to the relation. However, since usually the relation is affected by several factors and is not known precisely in advance, we can only hypothesize a simple relation such as linearity between the time and the accuracy. Therefore, in practice, to make the trade-off, we have to iteratively re-build a vision system through trial and error in order to adjust the system to a given environment and goal, and it is almost impossible to adjust the system on the fly.

Here, we estimate a relative accuracy heuristically, and represent it as "confidence", ranging from 0 to 1. The confidence is an abstraction of the accuracy, so the higher the confidence becomes, the more accurate the estimation will be. "confidence", Based on the "confidence", we can formalize various trade-off problems as a simple producer-consumer problem. To clarify the description, we have introduced several functions. Fig1 shows the relation among those functions. Target information is denoted as a time function $x(t)$. $c(t)$ is the confidence of the $x(t)$. In the system, the producer computes a value of $x(t_n)$ from an input stream. The heavier computation method the producer uses, the higher confidence $c(t_n)$ of $x(t_n)$ becomes. The producer provides a history h of the stream, a set of $\{t_n, x(t_n), c(t_n)\}$.

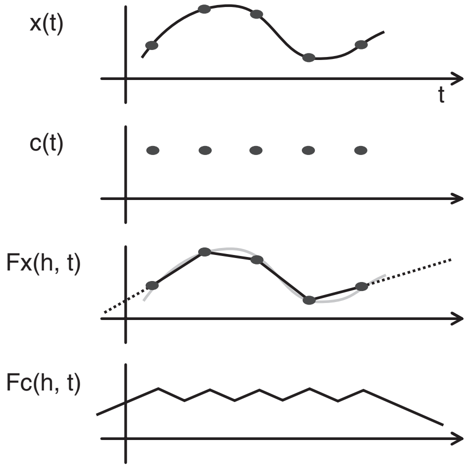


Figure 1. Relation among $x(t)$, $c(t)$, $F_x(h,t)$ and $F_c(h,t)$.

Then, we have introduced two estimators, $F_x(h,t)$ and $F_c(h,t)$, which are given by a system developer. $F_x(h,t)$ is an estimator of $x(t)$, which refers to the history h. When the history has not enough information to estimate $x(t)$, the estimation of $F_x(h,t)$ has less confidence. $F_c(h,t)$ is an estimator of $c(t)$, and, in general, the confidence is getting lower as the estimator predicts data of more distant future. Since the confidence only has a relative meaning of the accuracy, $F_c(h,t)$ is required to present the relative characteristic of the accuracy, not to present the absolute characteristics.

The key idea of our confidence-driven architecture is that it is not the data but the confidence which the consumer waits for. The data $x(t)$ is estimated by $F_x(h,t)$ in any time, and the consumer waits until the $F_c(h,t)$ becomes high enough. We call this scheme as "confidence-driven." Using the confidence-driven scheme, the trade-off between the accuracy and the latency is converted to the trade-off between the confidence and the latency. The point is that "confidence" only conveys a relative meaning about the accuracy of an acquired result and that its value need not have a precise physical meaning. The confidence is just used to control the trade-off and is decided by empirically or experimentally depending on the situation.

2.3. Confidence-driven Memory

Confidence-driven memory (CDM for short) is a shared memory based on the confidence-driven scheme. Producer and consumer are implemented as threads and can commu-

```
1 cdm_read(m, t, Cr, deadline) {
2   while ( Fc(m.h, t) < Cr ){
3     cond_wait(m.cond, deadline);
4     if (is_timedout)  break;
5   }
6   return {Fx(m.h, t), Fc(m.h, t) };
7 }
```

```
1 cdm_write(m, t, x, c) {
2     update_history(m.h, {t, x, c});
3     cond_signal(m.cond);
4 }
```

Figure 2. Pseudo codes of cdm_read and cdm_write operations.

nicate over the confidence-driven memory with three operations, cdm_read, cdm_write and cdm_poll with a pair of user-defined functions $F_x(h,t)$ and $F_c(h,t)$. Fig.2 shows pseudo codes of cdm_read and cdm_write operations. The consumer thread executes the cdm_read operation and waits until deadline for the confidence becomes high enough. When the consumer is suspended because of waiting, the producer thread computes data to make the confidence bigger. Thus each threads synchronized according to the confidence. In general, there is a certain overhead for the threads to share the data completely. However, in the confidence-driven scheme, each thread shares estimated data without the overhead while the confidence $F_c(h,t)$ is higher than the consumer requires.

A cdm_poll operation is for waiting for confidences of some confidence memories. It specifies an array of M CDMs and their required confidences and waits until deadline for the confidences of $N(\leq M)$ CDMs become higher than the required confidences. In other words, the waiting process at lines 2 to 5 in pseudo code of cdm_read operation is the case of $N = M = 1$ of cdm_poll.

3. Application to Vision-based Real-time Human Motion Sensing

3.1. Overview

Vision-based real-time human motion sensing[5] is a promising approach to measure human postures and motions. Because it does not require any special markers or attachments, it does not impose any physical restrictions on a user and then the user has no unnatural feelings to the system. However, there are several problems to be solved. From an architectural viewpoint, efficient computing mechanism which makes real-time processing possible is very important. This is because we have to adopt a multi-view approach to solve the self-occlusion problem in human motion sensing, and because the multi-view approach causes heavy computation. In addition, to apply the vision-based sensing to interactive applications, not only the throughput but the latency is quite important. Here, to make the human motion sensing system more efficient, we have applied confidence-driven architecture. At first, we briefly overview algorithms of our vision-based human motion sensing.

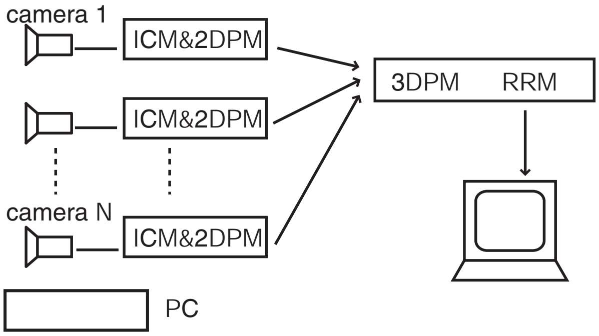


Figure 3. Overview of vision-based human motion sensing.

3.2. Algorithms for Human Motion Sensing

To solve the self-occlusion problem we have adopted a multi-view approach and the basic algorithm flow of our real-time motion capturing is illustrated in Fig. 3. To realize real-time processing, we have implemented the algorithm on a PC cluster.

Detection of perceptual cues

Usually, a limited number of perceptual cues of a human body can be detected robustly from images. Here, we have employed skin-color blobs and sock-color blobs are used as major perceptual cues, which correspond to a head[1], hands and feet. The blobs are detected based on color similarity analysis after a silhouette of a human body is extracted by background subtraction. The centroids of the blobs are used to calculate the 3D positions of those cues.

In general, the 3D positions of the color blobs can be calculated based on binocular stereo. However, with only two views, the self-occlusion problem can not be solved, and, we have adopted a multi-view approach. In addition, a torso position is also estimated based on the head position. The center of torso is estimated as a point just below the head with the distance of a predetermined value.

3D human motion synthesis

Since information acquired in the perception process is just 3D positions of a torso, a head, hands and feet of a human body, we have to estimate the body posture from these cues, the number of which is less than the degree of freedom of the body. Actually, we have to estimate 3D positions of elbows and knees. This problem can be solved in a framework of inverse kinematics, in which arms and legs are manipulators and the 3D blob positions are goal positions of

[1] Strictly speaking, it corresponds to a face.

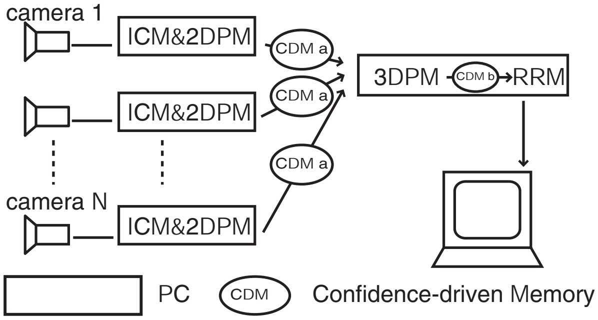


Figure 4. Human motion sensing implemented with CDM.

end effectors. robotics. According to a very simple human body model, which has 14 parts and 23 DOFs, we have established a simple model matching algorithm based on inverse kinematics and estimated human body postures. Inverse kinematics is solved by a popular cyclic coordinate descent method[4].

3.3. Confidence-driven Implementation

Since we have adopted a pipeline structure in our PC-cluster-based implementation to achieve high throughput, the latency of the system is not reduced yet. As mentioned, our real-time motion sensing is based on the observation of blob positions. Therefore, assuming the continuity of those values, we can apply the confidence-driven architecture to communication of the blob positions in the system and we expect the reduction of delay thanks to the confidence-driven architecture. In addition, the confidence-driven memory makes asynchronous execution of read and write operations, with which read operations can be issued more frequently than write operations are issued. Thus, rendering of a CG avatar in the system can be done frequently, which makes the avatar's motion smooth and natural.

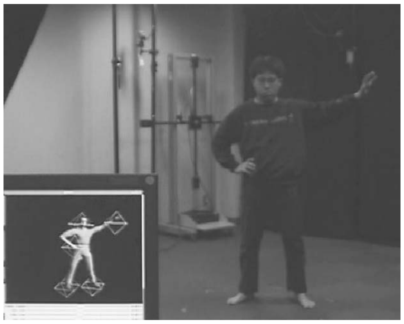
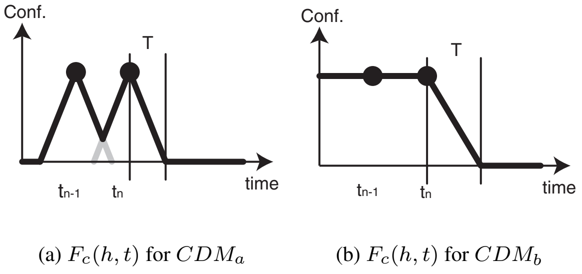


Figure 5. Environment for experiments.



(a) $F_c(h,t)$ for CDM_a (b) $F_c(h,t)$ for CDM_b

Figure 6. Definitions of two $F_c(h,t)$.

4. Experiment and Evaluation

Here, we evaluate the effectiveness of the confidence-driven architecture when it is applied to vision-based human motion sensing. Fig.5 shows our human motion sensing environment (At the left bottom corner, an estimated posture is displayed.). We have used 9 IEEE-1394 cameras and a PC-cluster with 10 PCs.

In this experiment, we use two confidence-driven memories, CDM_a and CDM_b. CDM_as are used for sharing the 2D positions of the blobs and for synchronizing image information acquired by asynchronous cameras, while CDM_b is used just for sharing the 3D positions of the blobs.

$F_{x_a}(h,t)$ is an estimating function used with `cdm_read` and `cdm_poll` operation via CDM_a. It returns an array of 2D positions of blobs which are used as candidates of the head, hands and feet. Fig.6(a) shows an outline of $F_{c_a}(h,t)$, which is defined according to heuristics that the error $\|x(t) - F_{x_a}(h,t)\|$ becomes large as the time difference between observed (or captured) time and accessed time t becomes large. Among 2D blobs detected different cameras, the most confident pair of 2D blobs at time t are selected, which are used to calculate the 3D position of the blob based on binocular stereo. In other words, we can select the most and the second most confident CDM_a from 10 CDM_as by `cdm_poll` operation with $F_{c_a}(h,t)$, and, thus, we can acquire the 3D position of each blob with high confident.

On the contrary, $F_{x_b}(h,t)$ is a linear prediction function. It returns the estimated 3D positions of the head, hands and feet. The confidence of the estimation is calculated by $F_{c_b}(h,t)$ shown in Fig.6(b), which always assigns 1 to the past estimation. This is because estimated 3D positions, or interpolated 3D positions, between observed times are always confident enough. Thus, postures of the user, which are estimated from the 3D positions of the blobs, can be estimated anytime, and, as a result, the CG avatar can be generated frequently, i.e., the CG avatar can move quite smoothly.

Fig.7 illustrates the relation between the required confi-

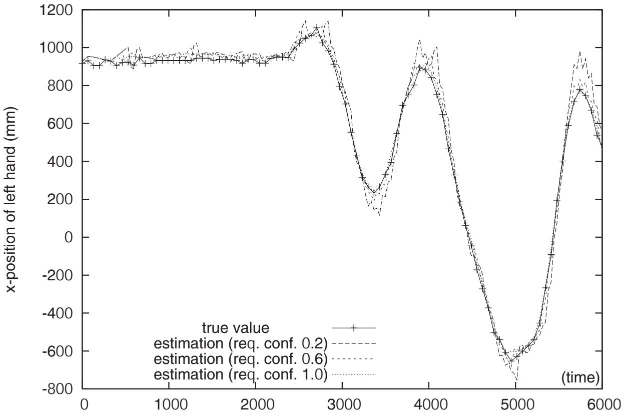


Figure 7. Required confidence and estimated left hand 3D position (x-axis).

dence and the error of a 3D blob (a left hand) position estimation in an image sequence. Table 1 shows the relation between the required confidence and the latency and several other statistics. These results show that the higher the required confidence becomes, the higher the accuracy of estimation becomes. On the other hand the lower the required confidence becomes, the smaller the latency becomes. In other words, low latency can be achieved by sacrificing the accuracy. However, it achieves relatively high accuracy due to an estimation mechanism built in confidence-driven memories. When the required confidence is too small, overshoots exhibit apparently, which cause unnatural motion recovery. For estimation of elbow position, the errors do not change drastically depending on the required confidence. This is because a method to solve the inverse kinematics is not very accurate, and because the difference of hand position does not make a large difference in elbow position estimation.

Since the confidence-driven memory makes asynchronous execution of read and write operations, rendering of a CG avatar in the system is done frequently, which makes the avatar's motion smooth and natural, and the lower the required confidence becomes, the more frequently the avatar is rendered. It is important to notice that by changing the required confidence, we can make the trade-off between the latency and the accuracy dynamically and smoothly.

5. Conclusion

In this paper, we have presented confidence-driven architecture to efficiently control difficult trade-off between the accuracy and the latency of real-time vision processing. Confidence-driven memory provides a simple but useful software interface for implementing several computational techniques known as imprecise computation model. The important point is that it provides a mechanism of synchronization with respect to confidence, which is an abstracted value of the accuracy. The confidence-driven architecture provides a mechanism to control the trade-off by just changing the confidence without rebuilding the system, and can increase the system execution efficiency in terms of throughput and latency.

Applying the confidence-driven architecture to vision-based real time human motion sensing, we can reduce the latency of a real-time motion sensing system by imprecise computation motion without a large decrease of the accuracy, which is shown in experimental results stated here. Currently, to examine the effectiveness of the confidence-driven architecture, we are going to combine several different vision algorithms using confidence-driven memories and to control the trade-off dynamically.

required confidence	0.2	0.6	1.0
latency(msec)	50	170	230
error (mm) in left hand pos. (average)	117.9	59.0	40.2
error (mm) in left hand pos. (std. dev.)	62.9	25.8	17.6
error (mm) in left elbow pos. (average)	124.7	119.4	116.4
error (mm) in left elbow pos. (std. dev.)	85.2	82.6	81.8

Table 1. Required conf. vs latency and estimation errors.

References

[1] J. Liu, W. Shih, K. Lin, R. Bettati, and J. Chung. Imprecise computations. *Proceedings of the IEEE*, 82(1):83–94, Jan. 1994.

[2] T. Matsuyama, S. Hiura, T. Wada, K. Murase, and A. Yoshioka. Dynamic memory: Architecture for real time integration of visual perception, camera action, and network communication. In *Proceedings of Comuter Vision and Pattern Recognition Conference*, pages 728–735, Jun. 2000.

[3] S. Singhal and M. Zyda. *Networked Virtual Environments*. Addison Wesley, 1999.

[4] L. Wang and C. Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, Aug. 1991.

[5] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, Jul. 1997.