Real-Time Imaging **7**, 529–544 (2001)
doi:10.1006/rtim.2001.0268, available online at http://www.idealibrary.com on IDEAL®

A Pyramid Approach to Motion Tracking

This paper presents a multiresolution approach to visual motion tracking. In the approach, the foveation mechanism of the human visual system is used to model the multiresolution information perception algorithms of a Transputer-based pyramid visual tracking system. The video images of a moving target are transformed into pyramidal data structures, each of those images consists of multiple image layers with different resolutions by a Gaussian pyramid generation algorithm. The tracking of a moving target over an image sequence is accomplished by performing a foveal search that is based on an iterative intensity pattern correlation along the multiple resolution levels of the Gaussian pyramids of two successive images. Analyses are given as to the efficiency and accuracy of our tracking algorithm, showing that the algorithm is over 160 times faster than conventional mono-resolution tracking methods, with the tracking error within one pixel. To demonstrate the superiority of the multiresolution tracking algorithm in the connection to parallel computation, a scheme for mapping the tracking algorithm into a Transputer-based pyramidal parallel computing structure is proposed in the paper. Experimental results demonstrate good performance of the proposed approach.

© 2001 Academic Press

Jason Z. Zhang and Q.M. Jonathan Wu

Vancouver Innovation Centre
National Research Council of Canada (NRC),
Vancouver, B. C., V6T 1W5, Canada
E-mail: {Jason.Zhang, Jonathan.Wu}@nrc.ca.

Introduction

In the case of visual motion tracking and analysis, the vast amount of visual information is a challenge to the processing capabilities of most computers we use in our offices or laboratories. However, it is commonly believed by the vision community that in most cases only a small fraction of the available visual information is relevant to the visual activities of a vision system performing particular vision tasks [1–6]. Neurophysiological studies into mammalian visual systems reveal that a mammal perceives its surrounding scene with different visual sensing resolutions [7–10]. A mammal observes the details of an object with its fovea, which is the area of maximum sensor density in the retina. The discriminability of the optic neurons in the retina decreases exponentially outwards to the peripheral area away from the fovea area. Nonetheless, the peripheral optic neurons are extremely sensitive to the motion cues in the field of view.

With the multiresolution visual mechanism, the natural visual system perceives environments in a highly efficient and active manner. In the visual system, the peripheral visual neurons remain alert for the appearance of any new event in the field of view of the observer, while the fovea is focused on a target of interest with a much higher sensing resolution than that of the peripheral visual area. Upon the emergence of any new event in the field of view, the peripheral vision mechanism signals an alarm to the sense-center in the brain of the observer. The foveo-peripheral visual mechanism works in a integrated spatio-temporal sensing manner [7,11], and the brain guides the oculomotor action as well as the body motion (if necessary) accordingly so that the newly emerging event

1077-2014/01/060529+16 $35.00/0
© 2001 Academic Press

in the scene falls into the foveal areas for more careful observation. This multiresolution visual mechanism works very well in dynamic environments.

For an artificial visual tracking system with similar multiresolution vision capability, it is expected to be able to selectively discard the excessive visual information details, adapt its visual resolving power to different vision tasks, and intelligently keep the concentration of the tracking system on the area where a target object appears in the images. Nevertheless, the integrated spatio-temporal visual mechanism in the mammalian vision systems is rarely used in computer vision because of the difficulties in its implementation; instead we will use separated spatial and temporal processing methods to approximate the joint spatio-temporal visual information processing mechanism.

Previous work on multiresolution motion tracking has been reported. In [2] and [12], multiresolution visual energy change information represented in Laplacian and Gaussian image pyramids was used in motion detection and tracking. Particularly in [2], Burt implemented a coarse-to-fine multiresolution target search scheme in a pipeline architecture that is based on specifically-purposed parallel processors, demonstrating an efficient solution to the parallel multiresolution motion sensing problem. However, because a relatively complex technique of the integration of Gaussian and Laplacian image pyramids was used, Burt's method is not efficient in image representation. In addition, because of the specific parallel computing architecture employed, Burt's vision system is not easily applied in general application cases. Moreover, quantitative analysis of the efficiency improvement brought about by the utilization of the multiresolution technique was reported neither in [2] nor in [12]. In contrast, in this paper we propose a motion tracking method based on a Gaussian pyramid correlation technique. The method is implemented in a pyramidal parallel vision system that is a universally-purposed Transputers-based architecture. Detailed efficiency analyses for our algorithms and parallel architecture are also given.

More recent work towards the implementation of multiresolution motion tracking has also been accomplished by researchers. Bandera *et al*. [4] presented a physiologically inspired connectionist neuron model to exploit a foveal retinotopology to reduce the computational complexity of local motion estimation and attentional connectionist tracking. However, in their work, neither any experimental results on real image data nor any theoretical analyses on the computational complexity reduction were reported. Efforts on the application of log-polar multiresolution image representation schemes to motion tracking were presented in Lim *et al*. [13] and Vincze and Weiman [14]. Lim *et al*. [13] put forth foveation schemes for tracking a single target as well as multiple targets in a simulated indoor scene sensed by a simulated pan/tilt active camera under a log-polar image coordinate framework. Vincze and Weiman [14] investigated the influence of log-polar multiresolution imagery on tracking performance in the circumstances of visual servoing and concluded that the log-polar imagery based tracking technique can demonstrate the best performance as compared to the window-size tailoring schemes for optimizing the tracking performance.

Although we have to admit the advantages of the log-polar multiresolution imagery techniques showing in the above cases of vision application, the techniques in fact largely rely on the spatially variant sensors [15,16], whose sensing element distribution pattern is logarithmic rather than Cartesian. Implementing a metric vision task, e.g., three-dimensional structure recovery from visual measurements, in the logarithmic polar frame implies extra effort to transform the Cartesian to its log-polar counterpart, an open problem in computer vision.

In this paper, we present a strategy for implementing a parallel pyramid solution to the multiresolution motion tracking in a real clustered outdoor scene in a Cartesian image coordinate framework. We derive a method for generating discrete Gaussian image pyramids. We present a coarse-to-fine target search strategy that is implemented as an open-loop motion-tracking algorithm operating within the Gaussian pyramids of successive image pairs in the video image flow. Quantitative analyses are given demonstrating the efficiency improvement and the tracking accuracy of the algorithm. We also propose a scheme for the parallel implementation of our multiresolution tracking algorithm in a parallel computing structure called the *Elementary Pyramid* consisting of four parallel Transputer processors. The performance of the Elementary Pyramid is investigated through an analysis of its operational efficiency. Theoretical investigation together with the experimental results presented in this paper show the superiority of our multiresolution pyramidal method over conventional methods.

We discuss the methodology of multiresolution image representation in the following section. Next, the strategy of the coarse-to-fine object search for motion tracking is presented. The Elementary Pyramid structure for the parallel implementation of the multiresolution motion tracking algorithm is then given. In the penultimate section the experimental results are presented to verify different aspects of the work in the paper. Finally, concluding remarks are made.

Pyramidal Image Structure

Resolution is a basic concept in signal processing problems and may be given different definitions in various environments. In this paper, the resolution of a signal $f(x)$ is defined as the amount of change in the independent variable x, corresponding to the smallest discriminatory change in the value of $f(x)$. If the resolution of an original signal $f(x)$ is regarded as 1, and the resolution of an approximation of $f(x)$ at the $(-j)$-*th* level is denoted by 2^j, where $j \in \boldsymbol{Z}^-$, 2^j is termed *dyadic resolution* of $f(x)$, and $\boldsymbol{Z}^-$ is the filed of negative integers plus zero. Wherever no confusion can be caused in the paper, we will briefly use *resolution* to refer to the dyadic resolution. We define A_{2^j} as a linear operator such that, for any $f(x) \in \boldsymbol{L}^2(\boldsymbol{R})$, where $\boldsymbol{L}^2(\boldsymbol{R})$ is the vector space consisting of all measurable and square integrable one-dimensional functions, $A_{2^j}f(x)$ is a resultant approximation of $f(x)$ at resolution $2^j (j \in \boldsymbol{Z}^-)$. Similarly $\boldsymbol{l}^2(\boldsymbol{Z}^2)$ denotes the vector space consisting of all square summable 2D sequences.

This section focuses on the construction of a Gaussian pyramidal structures of digital images for the uses in multiresolution tracking application that will be described in the next section.

Let $f_d(m,n) \in \boldsymbol{l}^2(\boldsymbol{Z}^2)$, where $0 \le m < C$, $0 \le n < R, C$ and $R \in \boldsymbol{Z}^+$, be a discrete version of $f(x,y)$ obtained by sampling with a pulse array

$$P(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x - mT_s, y - nT_s), \text{ where}$$

$$\delta(x,y) = \begin{cases} 1, & x = 0,\ y = 0 \\ 0, & elsewhere. \end{cases}$$

i.e.,

$$f_d(m,n) = f(x,y)P(x,y) = \sum_{m,n \in z} f(mT_s, nT_s)\, \delta(x - mT_s, y - nT_s). \tag{1}$$

A mapping sequence $(I_2{}^j)_{j \in z^-}$ that is a 2D iteration formulation is defined as below:

$$I_{2^j} \quad A_{2^j} f_d(m,n) = \sum_{p=-N}^{N} \sum_{q=-N}^{N} w(p,q) A_{2^{j+1}} f_d(2m+p,\ 2n+q) \tag{2}$$

where $A_{2^0} = 1$, $N \in \boldsymbol{Z}^+$, $w(p,q)$ is a weighting window function that meets the following conditions:

1. Decomposition: $w(p,q) = w(p)w(q)$;
2. Unitary: $\sum_{p=-N}^{N} w(p) = 1$
3. Symmetry: $w(p) = w(-p)$, and
4. Equal-distribution:

$$w(0) + 2w(2) + \cdots + 2w(k_e) = 2w(1) + 2w(3) + \cdots 2w(k_o),$$

where k_e and k_o are the maximum even and odd integers smaller than N, respectively. Then $[A_{2^j} f_d(m,n)]_{j \in \boldsymbol{Z}^-}$ is called the Gaussian multiresolution representation of $f_{\mathrm{d}}(m,n)$.

In (2) the sum at the boundary points, $(p,q) \in \{(m,n) | m = 0 \text{ or } C_j - 1, \text{ or } n = 0 \text{ or } R_j - 1\}$, where $C_j = 2^j C$, $R_j = 2^j R, j \in z^-$, can, for the purpose of simplicity, be performed in a cyclic manner. For example, for $(m,n) = (0,0)$, (2) is calculated as:

$$A_{2^j} f_d(0,0) = \sum_{p=0}^{N} \sum_{q=0}^{N} w(p,q) A_{2^{j+1}} f_d(p,q) + \sum_{p=-N}^{-1} \sum_{q=-N}^{-1} w(p,q) A_{2^{j+1}} f_d(C_j + p, R_j + q).$$

The cyclic sums at other boundary points can be performed in the same way. The illustrative diagram of a discrete image pyramid generated by this method is given in Figure 1.

Since the weighted window function $w(m,n)$ in (2) is separate, the 2D convolution sum can thus be decomposed into two single dimensional convolution sums performed over the two dimensions of the images, i.e.,

$$\begin{aligned} & A_{2^j} f_d(m,n) \\ &= \sum_{p=-N}^{N} w(p) \sum_{q=-N}^{N} w(q) A_{2^{j+1}} f_d(2m+p, 2n+q) \\ &= \sum_{p=-N}^{N} w(p)\, g(2m+p, 2n), \end{aligned} \tag{3}$$

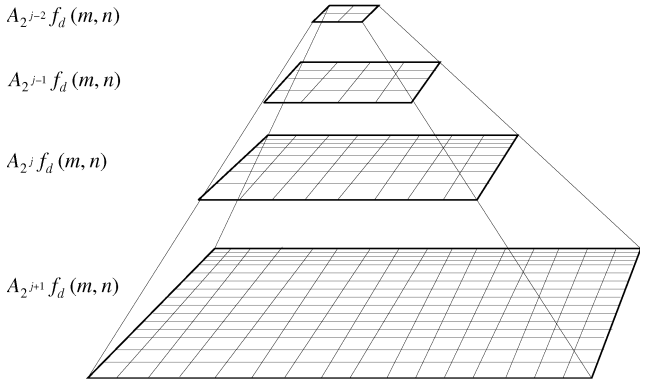

Figure 1. The illustration for the Gaussian discrete image pyramid.

where $g(2m+p, 2n) = \sum_{q=-N}^{N} w(q) A_{2^{j+1}} f_d(2m+p, 2n+q)$.

The number of multiplication operations required to generate the blurred image layer at resolution $2^j, (j \in \boldsymbol{Z}^-)$ is $(2N+1)$ $(C_j + R_j)$ instead of $(2N+1)$ C_j R_j. When $\frac{1}{C_j} + \frac{1}{R_j} \leq 1$ holds, a computational efficiency of $(\frac{C_j R_j}{C_j + R_j} \geq 1)$ can be gained.

We have described the method for generating discrete Gaussian image pyramids. Based on the results, we will present a multiresolution motion-tracking algorithm in the following section.

Multiresolution Tracking

In this section, we present an approach to motion tracking based on the correlation matching of image patches under a multiresolution framework. We utilize the image patches (or called *intensity patterns* hereafter when no confusion is raised) instead of other image patterns because we gain a higher tracking efficiency using simpler methods. The correlation matching method is simple but highly efficient. Moreover, due to its simple nature, it facilitates an analysis of the properties of multiresolution tracking algorithms.

Issues in multiresolution motion tracking

Following are the essential issues we need to deal with while designing the motion-tracking algorithm.

1. Pyramid representation of images. As we pointed out previously, the image pyramids form the basis of the multiresolution motion-tracking algorithm. From the point of view of facilitating the multiresolution image representations, we utilize the Gaussian pyramid in our tracking algorithm.

2. Coarse-to-fine search strategy. The strategy refers to an operational scheme for the search of a visual target performed within the image pyramids. In summary, the strategy possesses three advantages over the mono-resolution target-search methods:

- An iterative search process is performed in the multiresolution domain to increase the accuracy of the location of the target.
- A trial verification of the hypotheses about a visual phenomenon made at some resolution level is allowed at a higher resolution level during the multiresolution search process. With this feature, it is possible to increase the overall reliability of a visual analysis made by a motion-tracking algorithm. Moreover, new information useful to the current task may be found during the multiresolution search.
- The search at each resolution level (with the possible exception of the coarsest one) can be restricted within one or more particular area(s) to reduce the total computational cost of the search process. It is worth noting that the result of the coarse-to-fine search process partially depends on the initial resolution. Because a useful image feature appearing at a higher resolution level may disappear at a lower resolution level due to the blurring effect of the low-pass filter, the number of the resolution levels should be carefully selected so that essential image features can be kept at all resolution levels.

3. Correlation matching algorithm. Let $I_{t_{k-1}}(i, j)$ and $I_{t_k}(i, j)$, $(i, j, \in \boldsymbol{Z}^+)$ denote two successive digital images taken at the instants t_{k-1} and t_k, respectively. $\boldsymbol{q}_1$ denotes the intensity pattern of a motion target appearing in $I_{t_{k-1}}(i,j)$, and let $\boldsymbol{q}_2$ be the area covered by $\boldsymbol{q}_1$ after a displacement (Δ_i, Δ_j) in $I_{t_k}(i, j)$. The $\boldsymbol{q}_1$ in the first image is selected manually, serving as the template for matching. Let $E(x)$, $\sigma(x)$, and $cov(x, y)$ be the expectation, standard deviation, and covariance of x and y, respectively, then the normalized correlation coefficient of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ is calculated as:

$$R(\ _i,\ _j) = \frac{cov(\boldsymbol{q}_1, \boldsymbol{q}_2)}{\sigma(\boldsymbol{q}_1)\sigma(\boldsymbol{q}_2)}$$

$$= \frac{E(\boldsymbol{q}_1\boldsymbol{q}_2) - E(\boldsymbol{q}_1)E(\boldsymbol{q}_2)}{\sigma(\boldsymbol{q}_1)\sigma(\boldsymbol{q}_2)}, \quad (4)$$

where $E(\boldsymbol{q}_1\boldsymbol{q}_2)$ is the expectation of the pixel products of $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$.

If $\quad_i \in [-I, I]$ and $\quad_j \in [-J, J]$, where $I, J \in \boldsymbol{Z}^+$, then the final displacement $(\quad'_i, \quad'_j)$ of the target pattern over the two successive instants is determined by the following equation:

$$R(\quad'_i, \quad'_j) = \max_{\substack{-I \le \quad_i \le I \\ -J \le \quad_j \le J}} R(\quad_i, \quad_j). \tag{5}$$

Theoretically, the above correlation algorithm works under a pure translation model. The algorithm in fact also works as well with a more general motion model where both the translation and rotation of a target object with respect to the cameras are present, given that the image capturing rate is fast enough as compared to the target motion speed. Under this condition, the visual motion between two successive images is nearly pure translation and the algorithm is therefore applicable despite the rotation component in the motion. This feature has been demonstrated in our experiments.

4. Information propagation among resolution levels. Because the image pyramids we use are generated in accordance with the rule of dyadic resolution reduction, one pixel in an image layer with a lower resolution in the image pyramid corresponds to 2×2 pixels in its beneath image layer with a higher resolution, as shown in Figure 2. Sequentially, the projection of a pixel (i, j) in the k-th resolution level onto the m-th level ($m < k$) is a square image area consisting of $2^{k-m} \times 2^{k-m}$ pixels. The coordinates of the upper-left pixel of the projected image area (u, v) is related to (i, j) with the following equation:

$$\left.\begin{aligned} u &= 2^{k-m} i, \\ v &= 2^{k-m} j. \end{aligned}\right\} \tag{6}$$

Thus, the information propagation among different image layers for pixel correspondence, search path selection, and result exchanges should obey the relationship defined in (6).

5. Search window.
For high efficiency, the search process at every resolution level (except for the top one) should be restricted to a limited area, i.e. a search window, as shown in Figure 3. Let a_0 and b_0 be the superior limits of the two visual displacement components of the target to be tracked in initial image, i.e.,

$$\left.\begin{aligned} a_0 &= \sup_x |\quad_i|, \\ b_0 &= \sup_y |\quad_j|, \end{aligned}\right\} \tag{7}$$

where Δ_i and Δ_j can be determined empirically. The size of the corresponding search window at the k-th resolution level can be derived according to (6) as:

$$\left.\begin{aligned} a_k &= 2^{-k} a_0, \\ b_k &= 2^{-k} b_0, \end{aligned}\right\} \tag{8}$$

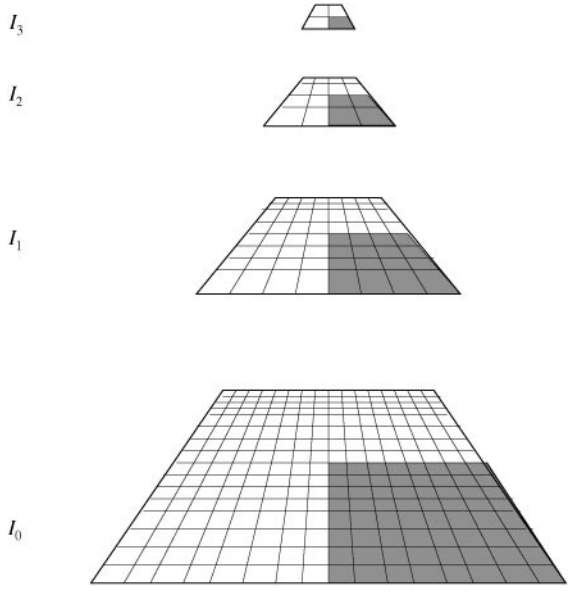

Figure 2. The relationship between the pixels in different resolution levels.

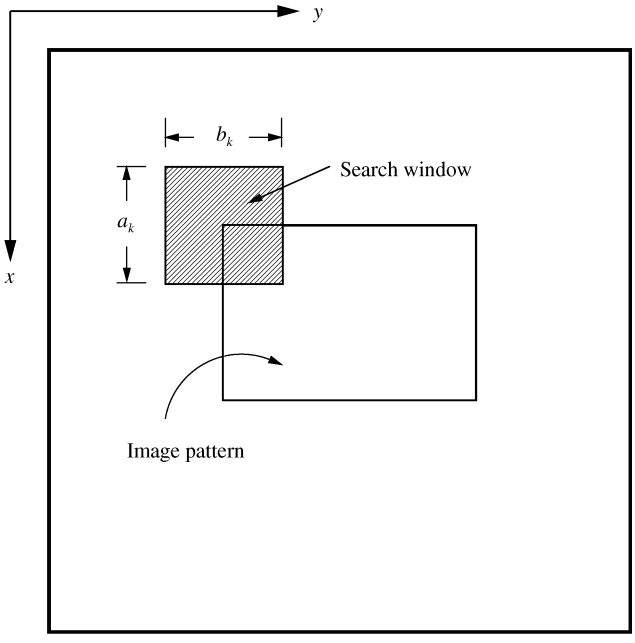

Figure 3. The schematic diagram for the determination of the search window.

Using the inter-level projection relations of pixel and image length in (6) and (8), the position and the size of a search window at a given resolution level in an image pyramid can be determined.

Based on the solutions of the above background mathematics, we can design a multiresolution motion-tracking algorithm. Given an image sequence $\{I_{t_k} | t_k = 0, \ ,2 \ , \ \ ,k \ , \ \}$, where Δ is the image-capture interval, and assume the location of the target in I_0 is manually determined by the system operator. The target is regarded as being tracked if its positions in the sequential images $\{I_{t_k} | t_k = \ ,2 \ , \ \ ,k \ , \ \}$ are successfully determined.

To accomplish the tracking algorithm, we must first design an algorithm for accomplishing the multiresolution correlation matching of the intensity-level image patterns. In the following subsections, the design of the correlation-matching algorithm and the issues related to algorithm performance analysis are presented.

The correlation matching algorithm

The matching algorithm is outlined as below:
Algorithm 1 (Correlation Matching)

1. Construct the Gaussian pyramids of two successive images.
2. Compute the displacement of a moving target at the coarsest resolution level using (4)–(5).
3. Map the displacement obtained in the last step onto the next resolution level using (6). Then compute the displacement at the current resolution level using (4)–(5).
4. Check up if the matching process has reached at the finest resolution level. If so, then stop the matching process and output the matching results; otherwise, go back to Step 3.

Performance analyses of the matching algorithm

1. Efficiency Estimation. In order to quantitatively estimate the computational efficiency of the Algorithm 1, we define a *computational cost index* of the algorithm in terms of the number of the multiplication operations necessary for completing a cycle of the multiresolution matching of the target. The computational cost index S_p is calculated using following equation:

$$S_p = S_p^{(1)} + S_p^{(2)} + S_p^{(3)}, \tag{9}$$

where $S_p^{(1)}$ is the cost index for computing the Gaussian pyramid of the reference target pattern in Step 1 of the matching algorithm, $S_p^{(2)}$ the index for computing the pyramid of the image to be matched in Step 2 of the algorithm, and $S_p^{(3)}$ the computational index of the correlation matching operations.

As shown in Figure 4, assume the image pyramids have n layers, the size of the top layer is $p \times p$, the size of the search windows at all layers except for the top one is $a \times a$, the size of the target pattern at the top layer is

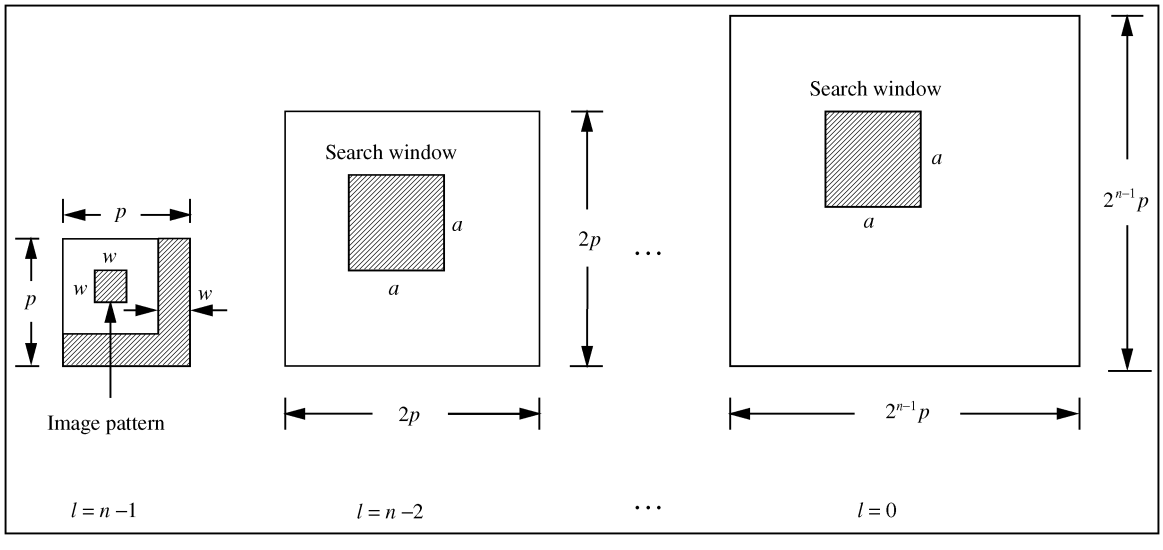

Figure 4. The schematic diagram of the coarse-to-fine correlation matching of the image patterns.

$w \times w$, and the number of the non-zero components of the Gaussian weight function $w(n)$ in (3) is w_1. Thus, we have

$$S_p^{(1)} = w^2 w_1^2 (1 + 4 + \cdots + 2^{2(n-2)}) = w^2 w_1^2 \frac{4^{n-1} - 1}{3}. \quad (10)$$

Similarly, the index $S_p^{(2)}$ is calculated using the following equation:

$$S_p^{(2)} = p^2 w_1^2 (1 + 4 + \cdots + 2^{2(n-2)}) = p^2 w_1^2 \frac{4^{n-1} - 1}{3}. \quad (11)$$

From (4) we can see that the number of the multiplication operations needed for accomplishing the correlation matching for a image pattern consisting of $c \times d$ pixels is roughly equal to $3cd + 52$. Thus, the cost index of the image matching operations at the top layer is $(p-w)^2(3w^2+52)$, and the index for the similar operations at each lower layer will be a^2 $(3 \times 2^{2(n-1-l)} w^2+52)$, where $l=0,1, \quad ,n-2$. Therefore the total computational cost index of the multiresolution matching operations at these layers is

$$S_p^{(3)} = (p - w)^2(3w^2 + 52) + \sum_{l=1}^{n-1} a^2(3\times 4^l w^2+52) = (p - w)^2(3w^2 + 52) + 26n(n - 1)a^2 + a^2w^2(4^n - 4). \quad (12)$$

Therefore we can get the total computational cost index of our multiresolution matching algorithm as below:

$$S_p = S_p^{(1)} + S_p^{(2)} + S_p^{(3)} = (w^2 + p^2)w_1^2 \frac{4^{n-1} - 1}{3} + (p = w)^2(3w^2 + 52) + 26n(n - 1)a^2 + a^2w^2(4^n - 4). \quad (13)$$

To estimate the efficiency of the multiresolution-matching algorithm, we compute the computational cost index of a mono-resolution correlation-matching algorithm that is supposed to apply to the same images as below:

$$S_d = (2^{n-1}p - 2^{n-1}w)^2(3\times 2^{2(n-1)}w^2 + 52) = 4^{n-1}(p - w)^2(3\times 4^{n-1}w^2 + 52). \quad (14)$$

From (13) and (14), we can estimate the efficiency of the multiresolution matching algorithm using a ratio computed with following equation:

$$\gamma = \frac{S_d}{S_p}. \quad (15)$$

If let $n = 4$, $p = 16$, $w_1 = 3$, $w = 4$, and $a = 5$, then the γ in (15) is roughly equal to 170.

If n is big enough, (13) can be calculated with the following approximation:

$$S_p \approx 4^{n-1}\left(\frac{(w^2 + p^2)w_1^2}{3} + 4a^2w^2\right). \quad (16)$$

By substituting (15) with (14) and (16) we have,

$$\gamma \approx k4^n + q, \quad (17)$$

where

$$k = \frac{12w^2(p - w)^2}{\frac{(w^2+p^2)w_1^2}{3} + 4a^2w^2}, \qquad q = \frac{52(p - w)^2}{\frac{(w^2+p^2)w_1^2}{3} + 4a^2w^2}.$$

From (17) we can see that the efficiency of the multiresolution correlation matching algorithm increases exponentially as the number of the resolution levels increase. Nonetheless, the number of the resolution levels can not be too large. Otherwise, the information of the image patterns could completely disappear at the top layer of the pyramids, resulting in a sharp decline of the matching accuracy of the algorithm. Therefore a trade-off should be taken between the efficiency and accuracy when determining the layer number of the pyramid. Experimental results indicate that, for images with the dimensions of 128×128 pixels, $n=3$ or 4 are ideal choices, and for the images with the dimensions of 256×256 pixels, $n=4$ is a good choice.

2. Remarks on the reduction of matching error. The matching error refers to the differences between the displacement ($'_i$, $'_j$) calculated by the algorithm and the real displacement (Δ_x, Δ_y) of the moving target, i.e.,

$$\left.\begin{aligned} e_x &= {}'_i - {}_x, \\ e_y &= {}'_j - {}_y. \end{aligned}\right\} \quad (18)$$

Although a correlation-based image matching algorithm is generally sensitive to noise, Algorithm 1 is robust to noise because of its multiple image level information processing mechanism. Algorithm 1 reduces noise sensitivity in two ways. First, the repetitive low-pass filtering process in the algorithm can eliminate most random noises in the images. As the follow-up multilevel matching process starts from the top image layer, the influence of the noises on the match accuracy can be reduced to a very low level. Second, the matching errors on different image layers are isolated from each other.

That is, since the matching process at a certain resolution level is performed within a big enough search window rather than at a single point, errors in positioning the target on the immediately above level is suppressed or eliminated at a current level. Therefore, the possibility of the matching error accumulation over the image layers is removed. The overall mismatching possibility is thus reduced significantly as compared to mono-resolution algorithms.

Later, in the "Experiments" section, we will present our experimental results on the matching errors of our algorithm applied to synthetic image data. The results show the low matching error of our algorithm.

The motion tracking algorithm

With the image-matching algorithm presented previously, our goal of accomplishing a multiresolution motion-tracking algorithm can easily be achieved. First, find the position of the image area in image I_{t_1} that matches with the original image pattern of the target using the multiresolution matching algorithm proposed. Then, using this information, find the position of the image area in image I_{t_2} that matches with the new image template using the same image matching method as used for I_{t_1}. Continue the matching process with $I_{t_3}, I_{t_4},$, performing the motion tracking over time.

Pyramid Parallel Computing

Despite the enormous efforts by vision researchers to improve the efficiencies of vision algorithms, parallel computation is commonly regarded as a key means for machine vision systems to match human visual processing. Pyramid computing techniques that derive from the concept of multiresolution image processing seem to be one of promising parallel vision computing techniques [17,18].

The concept of pyramid parallel computing for image processing and analysis was first proposed by Uhr [19] while he was investigating the application of a parallel architecture called "recognition cone" to the problem of multiresolution image recognition. In the "recognition cone" architecture, multiple processing elements (PEs) are connected in the way the pixels connect to each other in an image pyramid. Efforts towards to the implementations of different parallel computing hardware architectures have been taken by other researchers [20–22] thereafter. In this section we present a pyramid parallel computing architecture that is based on the Transputer processors to demonstrate the benefits of mapping our multiresolution-tracking algorithm into a parallel computing architecture.

Pyramidal computing architectures

We implement a pyramid computing structure using IMS T800 Transputer as the processing element. Because an IMS T800 Transputer has four bi-directional 10 Mbits/s communication links, we use four T800's to construct a primary structure that is called an *Elementary Pyramid*. Figure 5(a) depicts the topology of the Elementary Pyramid. In the structure, there is a bi-directional communication link between each pair of PEs. The PC is a host that interfaces the system operator to the pyramid. The host is connected to a *root node* and other nodes called *leaf nodes* are connected as shown in Figure 5(a). The command and data flows from the host reach every PE in the parallel structure via the root node, and conversely, the processing results and status reports from each PE enter the host via the root node.

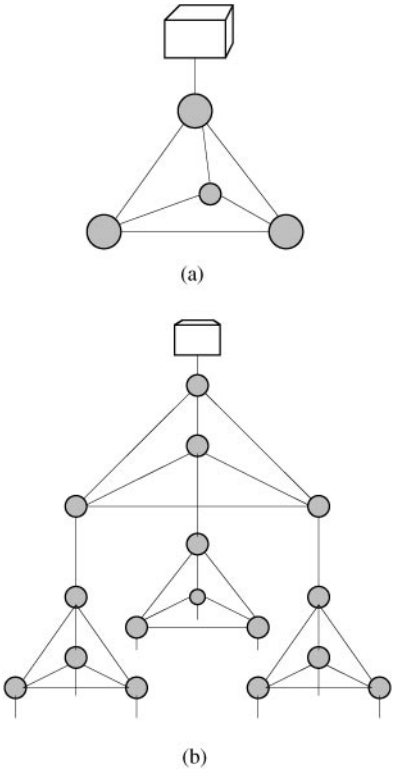

Figure 5. The topology of the pyramid parallel computing structures. (a) The Elementary Pyramid; (b) the Diamond Pyramid. □ Host; ◯ processor node; - comm. link.

Because every leaf PE node has one open link, more Elementary Pyramids can be connected as shown in Figure 5(b) to form a larger pyramid with multiple layers of Elementary Pyramids. The resultant structure is a hierarchy consisting of many individual processor tetrahedrons (Elementary Pyramids). The connection topology of the processor elements in Figure 5(b) resembles the carbon atoms in a diamond. Therefore we call the pyramid a *Diamond Pyramid*.

The Diamond Pyramid can be regarded as a pipeline structure whose processing units are processor arrays. Each of the arrays consists of $3^n (n \in \boldsymbol{Z}^+)$ Elementary Pyramids that are not connected to each other directly but through the PEs of the upper array. The PEs in each array can be programmed in single-instruction-and-multiple-data (SIMD) manner. The Diamond Pyramid works in a top–down manner for data assignment and query, and in a bottom–up manner for feedback.

We will not discuss the Diamond Pyramid further in the paper. Instead, for the sake of simplicity, our discussion will focus on the Elementary Pyramid.

Programming the Elementary Pyramid

The Elementary Pyramid works in the Host-Node fashion and is programmed in a hybrid language environment. The host PC runs in DOS/Windows and is programmed in Microsoft C. The Transputer processors, on the other hand, work in ParaSoft Express and are programmed in Express C. The two object codes are produced separately and linked by a Microsoft linker to form the integrated executables running the whole Elementary Pyramid. The Transputer nodes are treated as subroutines of the main program running in the host. The Transputer codes and the data segments are loaded into the individual nodes by the host. The processor nodes are started by the host to execute concurrently and feed the results back sequentially.

To make the pyramid computer work most efficiently, it is needed to segment an image-processing task into an "once-done" part and "repetitively-done" parts and assign the first to the host and the second to the parallel-processing unit, as illustrated in Figure 6. Following are some instances illustrating the task segmentation in the implementation of the multiresolution motion-tracking program in the Elementary Pyramid.

1. Task segmentation. In the tracking program, the "once-done" operations, such as system initialization, data I/O, information displaying, etc., are assigned to the host computer; the "repetitively-done" tasks, such as image pyramid construction and multiresolution image matching, are left for the Transputer processors. The system's bottle-neck lies in the data swap between the host and the Transputer processors. To reduce the bottle-neck, the root processor takes a more complex tasks than the leaf processors. In addition to performing the Host-node communication tasks, operation decision-making and similar data processing tasks as for the leaf nodes, the root node is responsible for analyzing and synthesizing the output of the leaf nodes.

2. Image data assignment. The Elementary Pyramid reaches its maximum processing efficiency when all the processors start and terminate simultaneously without

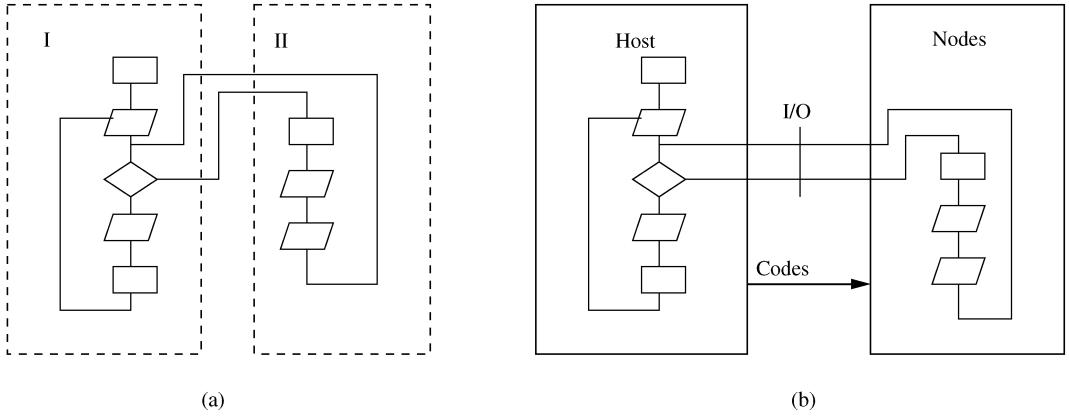

Figure 6. The task segmentation scheme for the Elementary Pyramid. (a) A vision task consisting of the "once-done" and the "repetitively-done" parts; (b) the task segmentation scheme for the Host-Node parallel processing structure.

any waiting time. An even data segment should be assign to each of the processors when they need to complete an image processing task. Figure 7 depicts one approach to the assignment of image data to the Elementary Pyramid. The rims added to each data segment are used for compensating the vestiges due to the data segmentation in the convolution operations at the image borders.

3. Search window decomposition. To accomplish the multiresolution pattern matching task in the processor array, the search window must be evenly segmented over the four processors. Furthermore, in order to reduce programming complexity, the window segments should be made with the same dimensions so that the program for each node (at least the three leaf nodes) is the same. Figure 8 shows an example of such a segmentation of the search window. The pixels around the central pixel within the search window are segmented into four parts, and each is assigned to one processor node of the Elementary Pyramid. Each node performs the search task within its own search region and reports the search result to the root node upon the termination of the program. The root node in turn, determines the final search result according to the results coming from the individual nodes.

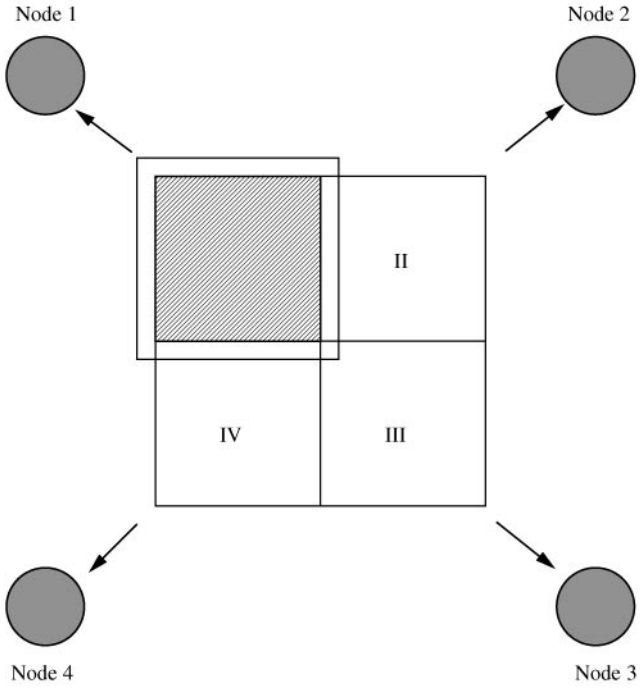

Figure 7. The image assignment over the processor array of the Elementary Pyramid.

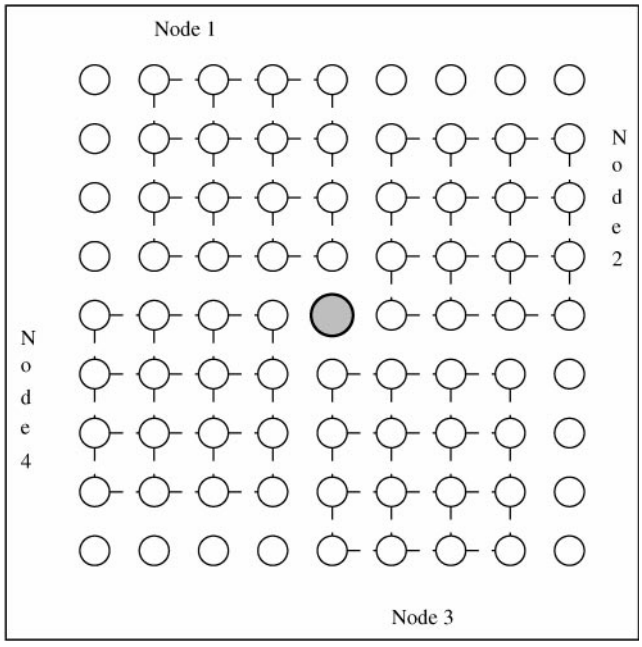

Figure 8. The decomposition of the search window. ⊙ central pixel; ○ neighbor pixel; search area.

Implementing the parallel motion tracking algorithm

The motion tracking algorithm can be implemented in the Elementary Pyramid, as schematically illustrated in Figure 9. The algorithm is decomposed into two parts according to functionality. The first part, which consists of the system management and decision-making modules is performed in the host. The second part, which contains the major data-intensive processing tasks is assigned to the Transputer array. The two parts dynamically interact one another during a tracking session via two kinds of links: the command flow and the data flow. To reduce the bottle neck of the system, local memories of the Transputers are used to buffer the immediate data produced by the parallel nodes.

Performance efficiency of the Elementary Pyramid

The performance efficiency of the Elementary Pyramid refers to the quantity of the improvement in system operation speed in comparison with a serial reference computer. In order to estimate the performance efficiency of the parallel machine, we define the following variables:

T: total time needed for the Elementary Pyramid to accomplish a given task;
T_h: time consumed by the host;

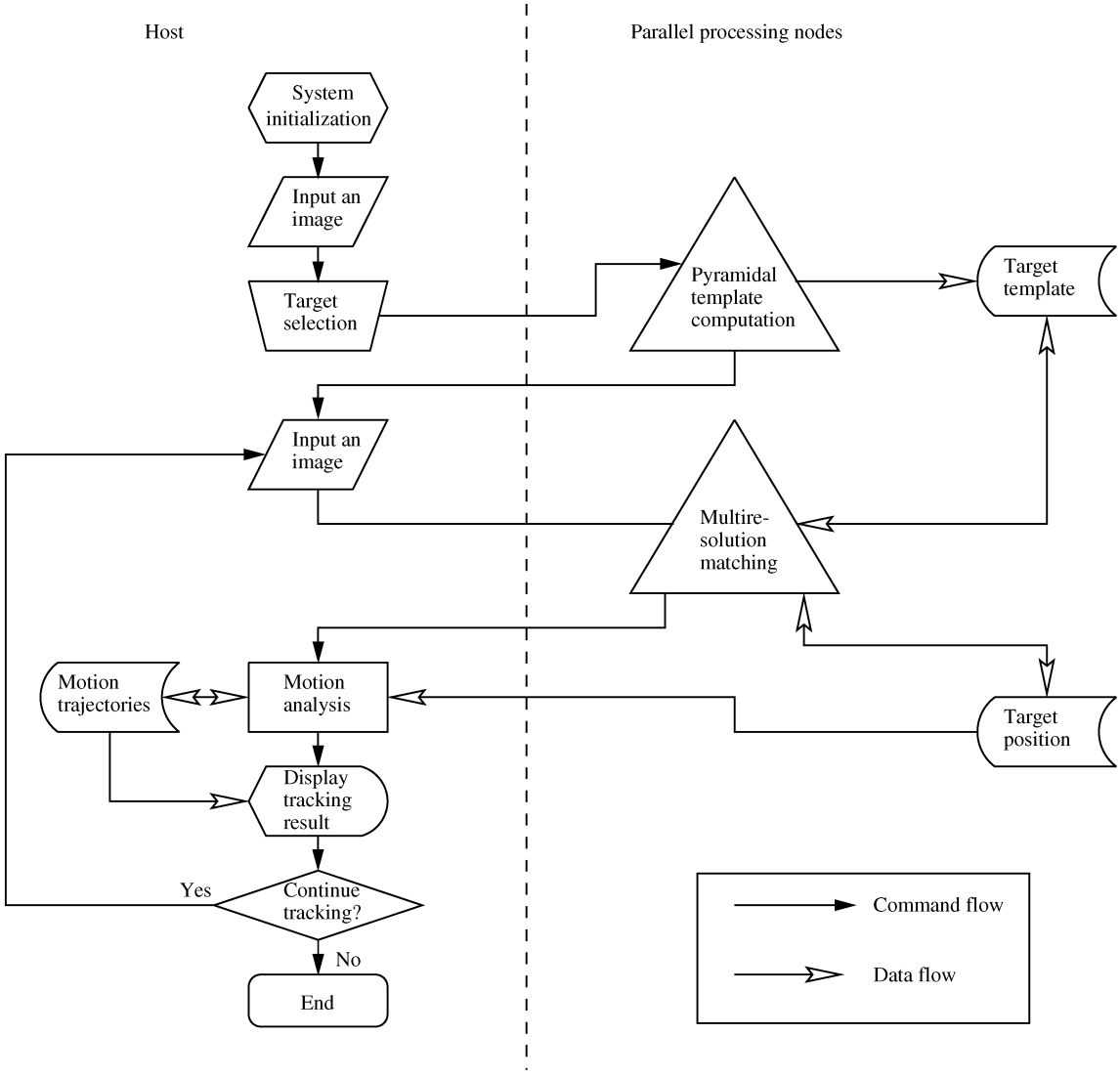

Figure 9. The block diagram of the parallel motion tracking algorithm implemented in the Elementary Pyramid.

T_a: time consumed by the processor array;
T_n: time consumed by one PE;
T_{ovl}: the overlapping portion of T_h and T_a;
τ_n: time needed for the data swap among PEs (suppose between the root node and a leaf node).

From the operational mechanism of the Elementary Pyramid we can see

$$T = T_h + T_a - 2T_{ovt}. \tag{19}$$

and

$$T_a = T_n + \tau_n. \tag{20}$$

The performance efficiency of the processor array η_a is calculated using following equation:

$$\eta_a = \frac{4T_n}{T_a}, \tag{21}$$

where the denominator stands for the time needed for the processor array to accomplish a given task, and the numerator refers to the equivalent time needed for one node to accomplish the same task performed by the processor array. By substituting (21) with (20) we have

$$\eta_a = \frac{4}{1 + (\tau_n / T_n)} \tag{22}$$

When $\tau_n \ll T_n, \eta_a \approx 4$. In this case, the processor array runs with its maximum performance efficiency. Therefore, the parallel algorithms for the processor array

should be designed to handle least data swaps among the processors so that the maximum performance efficiency can be achieved.

Let T_s denote the time needed for the host computer itself to complete the given task. Suppose the processing time consumed by the host t_h is proportional to the time needed by one node in the processor array with respect to the task performed by the node, i.e.,

$$t_h = sT_n, \tag{23}$$

where s is a constant that depends on the operational speeds of the two computers. Again given $\tau_n \ll T_n$, the equivalent time T'_h of the time needed for the processor to accomplish the whole task performed by the array can be computed by the following equation:

$$T'_h = 4sT_n. \tag{24}$$

Thus we have

$$T_s = T_h + T'_h = T_h + 4sT_n. \tag{25}$$

If the performance efficiency of the Elementary Pyramid η_p is evaluated against the host computer, then we have

$$\eta_p = \frac{T_s}{T} = \frac{T_h + 4sT_n}{T_h + T_a - 2T_{ovl}}. \tag{26}$$

If $T_{ovl} \ll \min\{Th, Ta\}$, then (26) reduces to

$$\eta_p = \frac{T_h + 4sT_n}{T_h + T_n}. \tag{27}$$

Eqn. (27) is a theoretical formula for computing the performance efficiency of the Elementary Pyramid. From the equation we can see that η_p depends on the properties of the two computers in comparison, the nature of the vision task to be performed, and the design style of the parallel program. In practice, T_h and T_n are easily estimated but, unfortunately, s is not. Sometimes, people may be more interested in s rather than η_p when they try to compare the apparent performances of two processors. In this case, η_p is instead estimated first by comparing the times needed by the two computers to accomplish a specific task, so that the estimate of s can be obtained through (27) from the measurements of η_p, T_h, and T_n. In the later subsection "Parallel computing", we will give our experimental results on η_p estimated from the measurements of T_s and T.

Experiments

In this section, we present the results of a group of the experiments we carried out to verify the different aspects of our work in this paper.

Motion tracking results

We applied our multiresolution motion tracking algorithm for the Elementary Pyramid to different image sequences that were produced with a Pericular 2000 Image Processing System from the video tapes of different motion scenarios. The size of the image frames of the image sequences is $128 \times 128 \times 8$ bits. The number of the resolution levels in the tracking algorithm is 4. For the target patterns shown in Figure 10 and Figure 11, our algorithm can reach a tracking speed up to $3 \sim 4$ frames per second.

Figure 10 illustrates the tracking result for an image sequence of a car coming towards the camera. The images in Figure 10 correspond to every 10th frame of the video sequence. The tracking marks with a same size remain on the moving target in spite of the changes in the target size and the position. The background remains nearly unchanged as the camera was fixed throughout the recording process. Notably, Figure 10 demonstrates the good performance of the algorithm in the presence of the obvious relative rotation between the car and the camera, as we addressed in a previous section.

Figure 11 shows the tracking result for the scenario of a bridge in the view of a "flying bird". The image sequence was taken from a clip of a 3D animation video generated by graphic software. In this case the whole picture keeps changing during the tracking process. However, the algorithm is still able to gaze at the correct target (the second pier from left selected manually before the program starts) throughout the tracking process.

The matching efficiency of Algorithm 1

We estimate the matching efficiency of Algorithm 1 by comparing the CUP time needed by our multiresolution correlation matching algorithm and a conventional mono-resolution correlation matching algorithm in a same computer. Since for each algorithm the CPU time is roughly proportional to the numbers of the multiplication operations in the algorithm, the estimates of the efficiency we get here should be near to that we obtained with (15).

The dimensions of the original images we used are $128 \times 128 \times 8$ bits, and the number of resolution levels of the image pyramids is 4. The parameter values of the algorithms used in the experiment are selected as the

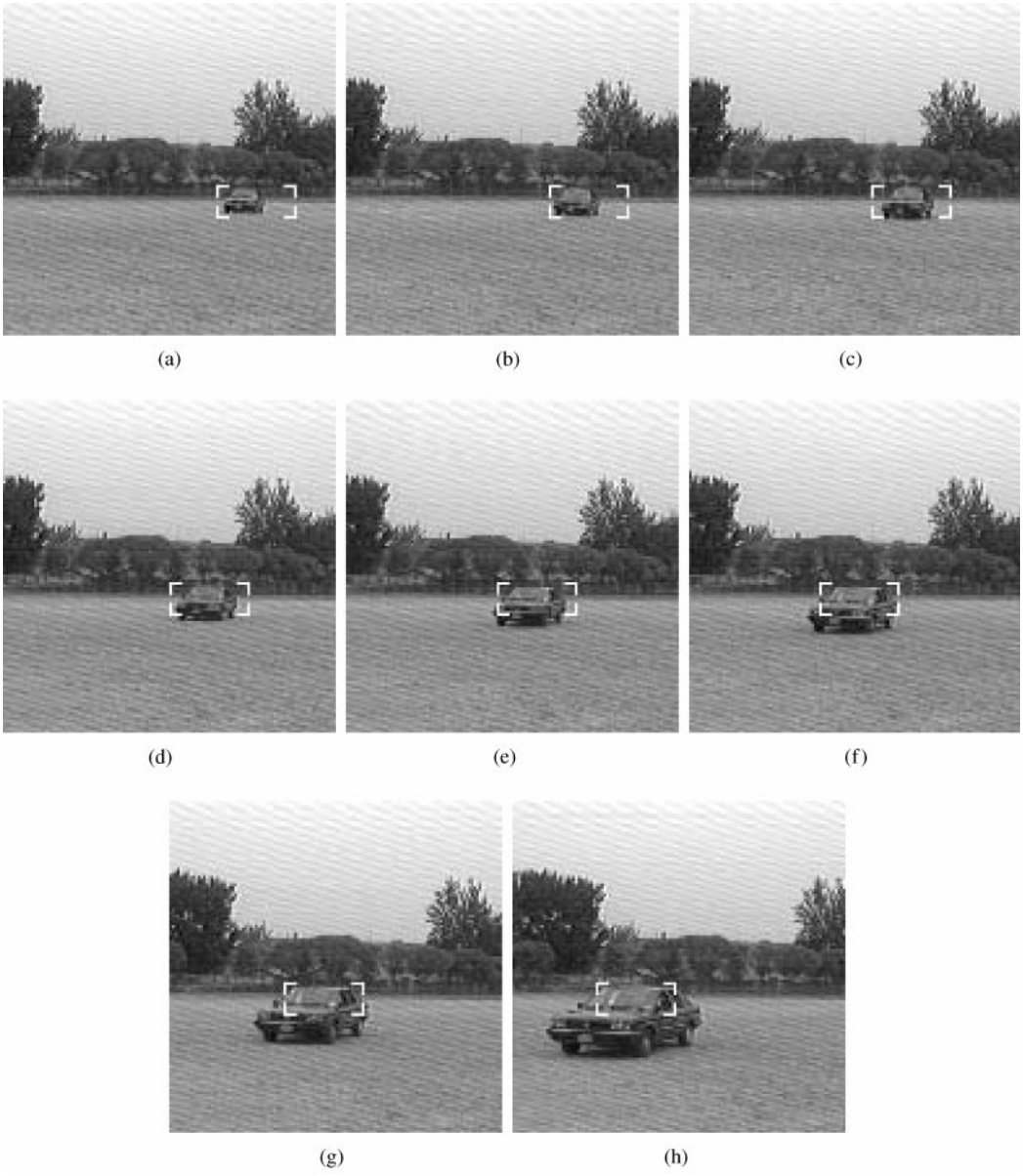

Figure 10. The tracking result of the parallel multiresolution tracking algorithm with a real image sequence of a moving car. (a)–(g) are the pictures numbered 10, 20, 80 in the image sequence.

same as those given previously. Table 1 lists the efficiency estimates for the target patterns with different sizes. From the table we can see that the estimates of the efficiency are around 160, which is quite close to that we obtained with (15). The CPU time estimates were achieved with an AST Premium 386/25 personal computer.

The matching error estimates of Algorithm 1

The purpose of the experiment is to quantitatively verify the matching accuracy of Algorithm 1. To generate known target positions for comparison, we synthesize an image sequence of a light rectangular spot moving along an elliptical orbit in a black background. We calculate the positions of the spot at 10 points on the ellipse; each is represented with one image. Each of the images has a size of $128 \times 128 \times 8$ bits. The ellipse locates at the center of the images and its major and minor axes are 43 pixels versus 32 pixels. The number of resolution levels used in the matching algorithm is 3. Table 2 lists the actual coordinates (x, y), estimated coordinates by the algorithm (x, y), and the absolute errors between the

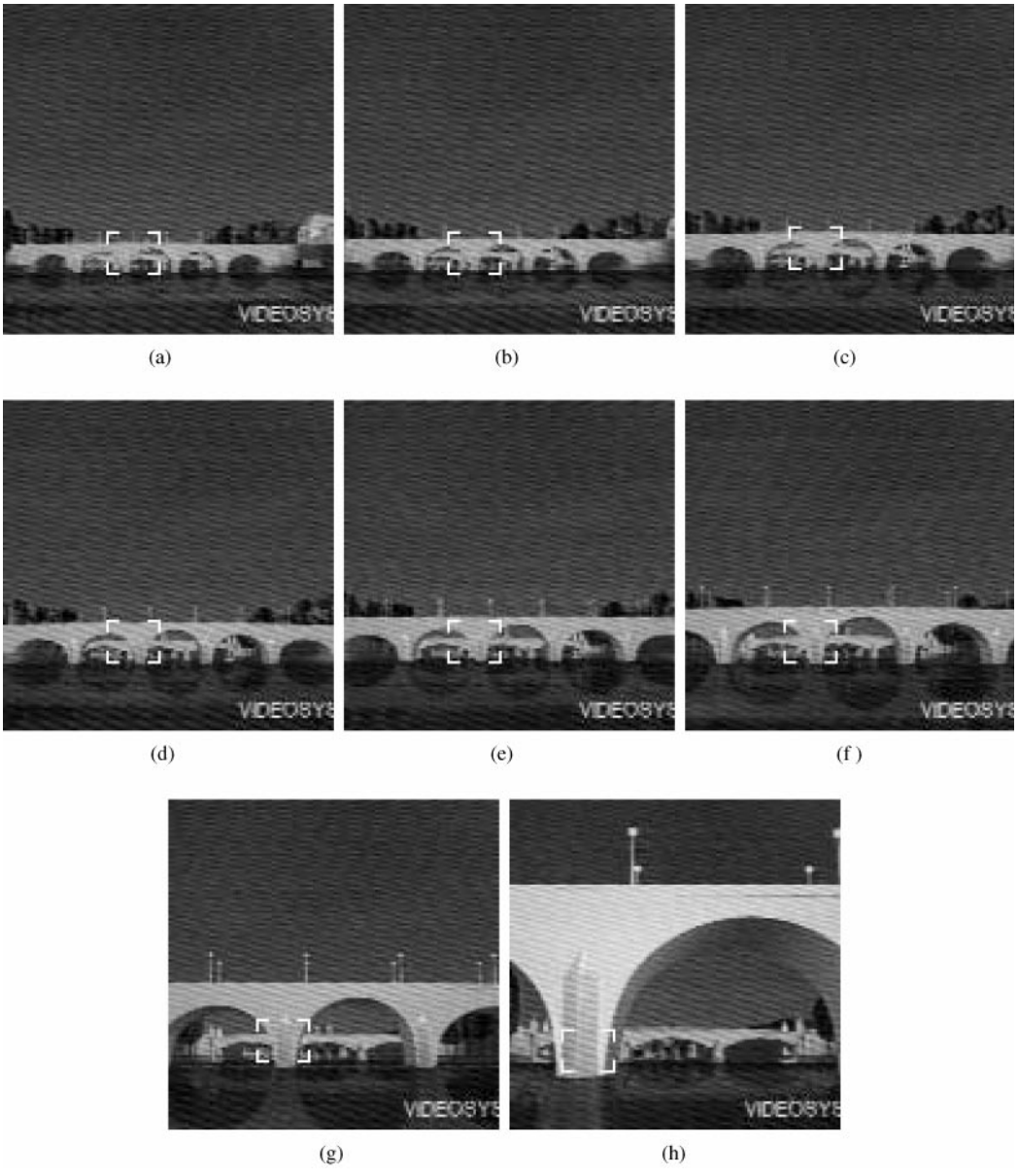

Figure 11. The tracking result of the parallel multiresolution tracking algorithm with a synthesized image sequence of bridge in the view of a ''flying bird''. (a)–(g) are the pictures numbered 10, 20, , 80 in the image sequence.

two set of coordinates (e_x, e_y). From the figures in the table we can see that the algorithm achieves an almost 100% correct matching rate under circumstances of ideal images. The maximum absolute matching error is 1 pixel.

Parallel computing

The experiment is to verify the performance efficiency of the Elementary Pyramid with respect to the following serial computers:

1. AST Premium 386/25 plus 387;
2. AST PII 386/33 plus 387;
3. SGI 4D/25 GT.

The AST Premium 386/25 was compared with the Elementary Pyramid with respect to the multiresolution matching algorithm (Algorithm 1). Table 3 shows the measurements of the CPU time for the two computers

Table 1. Estimates of the matching efficiency of Algorithm 1. (Unit: matching time: sec; efficiency: times)

Size	40×40	35×35	30×30	25×25	20×20	15×15
Mono-resolution Algr.	1189.52	1055.98	930.50	770.00	612.75	439.84
Multiresolution Algr.	7.47	6.70	5.87	4.89	4.01	2.93
Efficiency	159.24	159.25	158.52	157.46	152.81	150.12

Table 2. The Matching errors of Algorithm 1. (Unit: pixels)

Position	(x, y)	(x,y)	(e_x, e_y)
1	(105, 63)	(105, 64)	(0, 1)
2	(97, 45)	(97, 45)	(0, 0)
3	(76, 33)	(76, 33)	(0, 0)
4	(50, 33)	(50, 33)	(0, 0)
5	(29, 45)	(29, 45)	(0, 0)
6	(21, 63)	(21, 63)	(0, 0)
7	(29, 81)	(29, 81)	(0, 0)
8	(50, 93)	(50, 93)	(0, 0)
9	(76, 93)	(76, 93)	(0, 0)
10	(97, 81)	(97, 81)	(0, 0)

given the target patterns with different sizes. The η_p values derived from these time measurements are also listed in the table. From the results in the table we notice that the η_p values increase as the size of the target patterns increase.

The performance efficiency of the Elementary Py ramid was then verified with respect to the AST PII and the SGI 4D/25 GT respectively. The test algorithm utilized in the experiment is a Laplacian-of-Gaussian (LOG) filter for image edge detection. Table 4 and Table 5 list the two sets of results with respect to the two serial computers. In the tables, w refers to the 1D size of the convolution window of the LOG filter. In comparison with Table 3 we can see that the estimates of η_p in Table 4 and Table 5 are more stable. These experimental results substantiate the remarks we made on the performance efficiency of the Elementary Pyramid.

Conclusion

In this paper we have presented our work on a pyramid approach to motion tracking over an extensive topic region that covers the theory of multiresolution image representation, multiresolution motion tracking algorithm development, and the parallel computing solution to the motion-tracking algorithm.

More specifically, we regard the multiresolution image presentation as the problem of function approximation in the sense of functional analysis. According to this point of view, we define the multiresolution representation of an image as an approximation of a 2D function at multiple resolutions, and sequentially derive an algorithm for generating a discrete Gaussian pyramid of digital images. We develop an image-matching algorithm based on the correlation of image intensity patterns under the multiresolution image representation framework. This image matching algorithm, designed for the use of the motion tracking, proves theoretically and experimentally to increase the algorithm efficiency by a factor of about 160 as compared to the mono-resolution methods. A multiresolution motion-tracking algorithm is developed

Table 3. Performance efficiency of the Elementary Pyramid w.r.t. AST 386/25. (Unit: CPU time: sec; efficiency: times)

Size	40×40	35×35	30×30	25×25	20×20	15×15
AST 386/25	11.93	10.06	8.15	6.61	4.98	3.67
Pyramid	0.73	0.65	0.57	0.50	0.44	0.39
Efficiency	16.34	15.48	14.55	13.49	11.32	9.41

Table 4. Performance efficiency of the Elementary Pyramid w.r.t. AST 386/33. (Unit: CPU time: sec; efficiency: times)

$w \times w$	15×15	19×19	21×21	23×23	27×27	31×31
AST PII/33	61.6	75.1	82.1	89.0	102.7	116.7
Pyramid	19.4	22.5	24.1	25.6	28.8	32.1
Efficiency	3.2	3.3	3.4	3.5	3.6	3.6

Table 5. Performance efficiency of the Elementary Pyramid w.r.t. SGI 4D/25 GT. (Unit: CPU time: sec; efficiency: times)

$w \times w$	15×15	19×19	21×21	23×23	27×27	31×31
SGI 4D/25 GT	17.5	22.5	24.5	26.0	30.0	35.0
Pyramid	19.4	22.5	24.1	25.6	28.8	32.1
Efficiency	0.90	1.00	1.01	1.02	1.04	1.10

based on the image-matching algorithm. A parallel computing solution to the multiresolution motion-tracking problem is also proposed. A pyramid parallel computing structure based on Transputers is investigated with respect to its architecture, algorithm design, and performance efficiency. Experimental results with real image data show good tacking performances of the parallel computing structure.

It is proven by the work presented in this paper that the multiresolution image processing technique is remarkably effective for the problem of visual motion information processing. This is because the technique can better deal with some key problems in vision such as the compromise between local and global information in images, top–down and bottom–up information processing modeling, and parallel visual information compu ting. Its merits make the multiresolution image processing technique a promising approach to efficient computer vision.

References

1. Rosenfeld, A. (Ed.) (1984) *Multiresolution image processing*. New York: Springer-Verlag.
2. Burt, P.J. (1988) Smart sensing within a pyramid vision machine. *Proc. IEEE* **16**: 1006–1015.
3. Swain, M.J. & Stricker, M.A. (1993) Promising direction in active vision. *Int'l. J. of Computer Vision*. **11**: 109–126.
4. Bandera, C., *et al.* (1995) A multiacuity connectionist model for attentional control and tracking. *Proc. Int'l. Conf. on Engr. Appl. of Artif. Neural Networks*, Espoo, Finland, pp. I-234–I-237.
5. Reid, I.D. & Murray, D.W. (1996) Active tracking of foveated feature clusters using affine structure. *Int'l. J. of Computer Vision* **18**: 1–20.
6. Geisler, W.S. & Perry, J.S. (1998) A real-time foveated multiresolution system for low-bandwidth video communication. *Proc. SPIE*, **3299**: 294–305.
7. Hubel, D.H. & Wiesel, T.N. (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. Physiology* **160**: 106–154.
8. Hubel, D.H. & Wiesel, T.N. (1965) Receptive fields and functional architecture in two non-striate visual areas (18 and 19) the cat. *J. Neurophysiology* **28**: 229–289.
9. Marr, D. (1982) *Vision*. New York: Freeman and Company.
10. Hubel, D.H. (1995) *Eye, Brain and Vision*. Scientific American Library, No. 22., New York: WH Freeman.
11. Adelson, E. (1991) Mechanisms for motion perception. *Optics and Photonics News* **2**: 24–30.
12. Anderson, H., Burt, P.J. & van der Wal, G.S. (1985) Change detection and tracking using pyramid transformation techniques. *Proc. SPIE Conf. on Intell. Robots and Computer Vision*, Boston, MA, pp. 72–78.
13. Lim, F.L., West, G.A. & Venkatesh, S. (1996) Tracking in a space variant active vision system. *Proc. Int'l. Conf. on Pattern Recogn*. Los Alamitos, CA, pp. 745–749.
14. Vincze, M. & Weiman, C.F. (1997) On optimizing tracking performance for visual servoing. *Proc. IEEE Int'l. Conf. on Robotics and Automation*, Albuquerque, New Mexico, pp. 2856–2861.
15. Spiegel, V.D., *et al.* (1989) A foveated retina-like sensor using ccd technology. In: C. Mead and M. Ismall (Eds) *Analog VLSI Implementation of Neural Systems*, pp. 189–211.
16. Lim, F.L., Venkatesh, S. & West, G.A. (1996) Resolution consideration in spatially variant sensors. *Proc. Int'l. Conf. on Pattern Recogn*. Los Alamitos, CA, pp. 745–749.
17. Cantoni, V. & Levialdi, S. (1988) Multiprocessor computing for images. *Proc. IEEE* **16**: 959–969.
18. Weems, C., *et al.* (1992) Image understanding architecture: Exploring potential parallelism in machine vision. *IEEE Computer*, February: 65–68.
19. Uhr, L. (1972) Layered recognition cone networks that preprocess, classify, and describe. *IEEE Trans. on Computer* **21**: 758–768.
20. van der Wal, G.S. & Burt, P.J. (1992) A VLSI chip for multiresolution image analysis. *Int'l. J. of Computer Vision* **8**: 177–189.
21. Li, Z.N. & Zhang, D. (1993) *Fast line detection in a hybrid pyramid. Pattern Recogn. Letters* **14**: 53–63.
22. Jolion, J.-M. & Rosenfeld, A (1994) *A Pyramid Framework for Early Vision*. Dordrecht: Kluwer Academic Publishers.
23. Mallat, S. (1989) A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. on Patt. Analys. and Mach. Intell.* **11**: 674–693.
24. Marr, D. & Hildreth, E. (1980) Theory for edge detection. *Proc. R. Soc.*, **B207**: 187–217.
25. Babaud, J., *et al.* (1986) Uniqueness of the Gaussian kernel for scale-space-filtering. *IEEE Trans. Patt. Analy. and Mach. Intell.* **8**: 26–33.
26. Burt, P.J. & Adelson, E.H. (1988) The Laplacian algorithm as a compact image code. *IEEE Trans. Communications* **31**: 532–540.
27. Clark, M. & Levialdi, S. (1988) Multiprocessor computing for image. *Proc. IEEE* **16**: 959–969.