Dynamic visual tracking based on multiple feature matching and g–h filter

MING-YANG CHENG¹,*, MI-CHING TSAI² and CHIA-YANG SUN²
Departments of ¹Electrical Engineering and ²Mechanical Engineering, National Cheng Kung University, Taiwan

Received 18 November 2005; accepted 27 December 2005

Abstract—The area-based matching approach has been used extensively in many dynamic visual tracking systems to detect moving targets because it is computation efficient and does not require an object model. Unfortunately, area-based matching is sensitive to occlusion and illumination variation. In order to improve the robustness of visual tracking, two image cues, i.e., target template and target contour, are used in the proposed visual tracking algorithm. In particular, the target contour is represented by the active contour model that is used in combination with the fast greedy algorithm. However, to use the conventional active contour method, the initial contour needs to be provided manually. In order to facilitate the use of contour matching, a new approach that combines the adaptive background subtraction method with the border tracing technique was developed and is used to automatically generate the initial contour. In addition, a g–h filter is added to the visual loop to deal with the latency problem of visual feedback so that the performance of dynamic visual tracking can be improved. Experimental results demonstrate the effectiveness of the proposed approach.

Keywords: Area-based matching; visual tracking; active contour model; greedy algorithm; g–h filter.

1. INTRODUCTION

In visual tracking algorithms, image features, e.g., target template, target contour and color, are used for similarity matching to locate the current target position in the image plane. Among visual tracking algorithms, area-based matching (also called template matching or region-based matching) [1] is one of the most popular algorithms. However, in practical applications, area-based matching suffers from drawbacks such as sensitive to occlusion and illumination variation. In addition to area-based matching, contour matching is also very popular. One of the advantages of contour matching is that it is robust to occlusion. However, if a fixed target contour is used throughout the tracking process, only the target with a specific shape

*To whom correspondence should be addressed. E-mail: mycheng@mail.ncku.edu.tw
can be effectively tracked [2]. In order to apply the contour matching method to track objects that do not have specific geometric shapes, Kass et al. [3] proposed the active contour model which can change its contour according to the variation in the target’s shape. Unfortunately, most visual tracking algorithms based on the active contour method are limited to track slow moving targets [4]. In fact, visual tracking based on single-feature matching is not robust.

The common approach to overcome the aforementioned difficulty is to employ multiple-feature matching. For instance, Kragic and Christensen [5] proposed a multiple-visual-cues approach. By integrating the information from different visual cues, the most likely target position can be found. Triesch and Malsburg [6] developed a visual tracking system that can adjust the weightings for different visual cues according to individual confidence level. In addition, Chen [7] proposed a hybrid matching approach that combines template matching with contour matching to improve the robustness of visual tracking. On the other hand, in video surveillance applications, Kim and Moon [8] employed multi-resolution motion estimation to determine the motion vector of the object. The motion vector is used to help the active contour method to track the moving target. Kim and Lee [9] proposed the jump mode idea, where the block matching method is used to compute the optical flow of the moving target. Based on the obtained optical flow data, the target position in the next frame can be predicted and then the active contour model performs contour updating at the predicted target position.

In order to improve the robustness of visual tracking, this study develops a visual tracking algorithm that integrates area-based matching with contour matching. In addition, the adaptive background subtraction approach [10] is combined with the border tracing technique [11] to automatically generate the initial contour for the active contour model. Moreover, the fast greedy algorithm is employed to update the active contour model to approximate the target contour. Unlike the conventional area-based matching approaches adopt sums of squared difference (SSD) or sums of absolute difference (SAD) as the similarity measurement, in the proposed approach, spatial distribution of Gaussians (SDG)-based matching, template updating and template mask are used.

In general, the sampling time for the outer visual loop of a dynamic visual tracking system is dictated by the frame rate, in which it is much larger than the sampling time for the inner servo loop. Namely, the visual tracking system can be considered as a multi-rate digital control system [12]. In addition, the latency in the visual loop may pose problems to visually guided systems [13]. Sim et al. [14] proposed a multi-rate predictor control scheme to improve the visual tracking performance. In this study, in order to deal with the latency problem in the visual loop, a feedforward controller is adopted and a g–h filter is added to the visual loop. A real-time pan–tilt visual tracking system developed in our lab is used as a test platform and several experiments have been conducted to evaluate the performance of the proposed approach.
The remainder of the paper is organized as follows. Section 2 gives a review of the active contour method and the fast greedy algorithm. Automatic initial contour generation for the active contour method is introduced. Section 3 focuses on the proposed visual tracking algorithms. Section 4 provides the details about the visual servo structure, g–h filter and experimental setup. Experimental results and conclusions are included in Sections 5 and 6, respectively.

2. ACTIVE CONTOUR MODEL, FAST GREEDY ALGORITHM AND AUTOMATIC INITIAL CONTOUR GENERATION

2.1. Introduction to the active contour model

The active contour model, which is also referred to as the Snake [3], is basically a continuous curve governed by an energy-like function. It will undergo shape variations or movement if its energy changes. When applied to visual tracking problems, the active contour model is often used in combination with a search algorithm. When the iterative solution provided by the search algorithm results in a minimum energy, this particular solution is considered the target contour. The energy-like function of the active contour model is described mathematically as:

\[
E_{\text{snake}}^*(v) = \int_0^1 E_{\text{snake}}(v(s)) \, ds = \int_0^1 [E_{\text{int}}(v(s)) + E_{\text{ext}}(v(s))] \, ds,
\]

where \( E_{\text{int}} \) is the internal energy, \( E_{\text{ext}} \) is the external energy, \( v(s) \) is the position coordinate of the active contour model. In the two-dimensional (2-D) case, \( v(s) = (x(s), y(s)) \), where \( s \in [0, 1] \) is the arc length.

The internal energy relates to the geometric shape of the active contour model. It is used to set a constraint on the continuity and smoothness of the active contour model. Equation (2) gives the mathematical expressions of the internal energy.

\[
E_{\text{int}}(v(s)) = \frac{1}{2} [\alpha(s)|v_s(s)|^2 + \beta(s)|v_{ss}(s)|^2],
\]

where \( v_s(s) \equiv \frac{dv}{ds}, v_{ss}(s) \equiv \frac{d^2v}{ds^2}, \alpha(s) \) and \( \beta(s) \) are their associated weightings.

On the other hand, the external energy is dependent on the image data. If the goal is to approximate the edge of the target, the external energy can be defined as:

\[
E_{\text{ext}} = -|\nabla [G_\sigma * I(x, y)]|^2,
\]

where \( \nabla I(x, y) \) represent the gradient of the image intensity at pixel \( (x, y) \), and \( G_\sigma \) is a 2-D symmetric Gaussian function with standard deviation \( \sigma \).
2.2. Greedy algorithm

In Ref. [3], the method of calculus of variation is used to find a suitable contour model that approximates the target contour. However, as pointed out by Amini et al. [15], using calculus of variation to solve active contour problems may result in several drawbacks. One drawback is that numerical computation may become unstable so that no converged solution can be found. Another drawback is that the obtained ‘Snake’ tends to form a cluster around the portion of the image with strong edge contents. In order to overcome these difficulties, Williams and Shah proposed the greedy algorithm [16]. A brief introduction to the greedy algorithm is given in the following.

Assume that the contour of the Snake consists of \( n \) discrete control points \( v_i = (x_i, y_i), i = 1, 2, \ldots, n \), and \( v_1 = v_n \). Equation (1) is rewritten as:

\[
E_{\text{snake}}^n(v) = \sum_{i=1}^{n} E_{\text{snake}}(v_i). \tag{4}
\]

The greedy algorithm is based on the concept of local search. When applied to the Snake, the greedy algorithm calculates \( E_{\text{snake}} \) for each neighborhood pixel of the control point \( v_i \). The neighborhood pixel with the smallest \( E_{\text{snake}} \) is selected as the new control point. By repeating these procedures for each control point, theoretically, the minimum of \( E_{\text{snake}}^n \) in (4) can be obtained. In the greedy algorithm, the energy of the Snake is redefined as:

\[
E_{\text{snake}}(v_i) = \alpha_i E_{\text{cont}}(v_i) + \beta_i E_{\text{curv}}(v_i) + \gamma_i E_{\text{edge}}(v_i), \tag{5}
\]

where \( E_{\text{cont}} \) is the continuity energy and \( E_{\text{curv}} \) is the curvature energy and \( E_{\text{edge}} \) is the edge energy. In addition, \( \alpha_i, \beta_i, \) and \( \gamma_i \), are the associated weightings for \( E_{\text{cont}}, E_{\text{curv}} \) and \( E_{\text{edge}} \), respectively.

\[
E_{\text{cont}}(v_i) = \frac{|\bar{d} - |v_i - v_{i-1}||}{\max_j \{ |\bar{d} - |v_{i,j} - v_{i-1}|| \}}, \tag{6}
\]

\[
E_{\text{curv}}(v_i) = \frac{|v_{i-1} - 2v_i + v_{i+1}|^2}{\max_j \{ |v_{i-1} - 2v_{i,j} + v_{i+1}|^2 \}}, \tag{7}
\]

\[
E_{\text{edge}}(v_i) = \frac{\min_j \nabla I(v_{i,j}) - \nabla I(v_i)}{\max_j \nabla I(v_{i,j}) - \min_j \nabla I(v_{i,j})}. \tag{8}
\]

In (6), \( \bar{d} \) represents the average distance between two adjacent control points, i.e.:

\[
\bar{d} = \frac{\sum_{i=2}^{n} |v_i - v_{i-1}|}{n - 1}. \tag{9}
\]

In addition, in (6), \( j \) represents the index number of neighborhood pixels of \( v_i \), and \( v_{i,j} \) represents the possible destination of \( v_i \) for the \( j \)th searched neighborhood pixel.
2.3. Fast greedy algorithm

Lam and Yan proposed the fast greedy algorithm [17], which is a modified version of the greedy algorithm. The only difference between the fast greedy algorithm and the greedy algorithm is the selection of neighborhood pixels during the search process. If a $3 \times 3$ window centered at the control point is chosen as the neighborhood, then the greedy algorithm will evaluate $E_{\text{snake}}$ for each neighborhood pixel of the control point $v_i$, i.e., (5) will be evaluated a total of 9 times. In contrast, the fast greedy algorithm employs two search modes as shown in Fig. 1. No matter which search mode is used, (5) will be evaluated only 5 times. These two search modes are used interchangeably during the neighborhood search. Based on Fig. 1, it is easy to find that one of the search modes performs a ‘cross search’, while the other performs a ‘diagonal search’. Together these two search modes cover all nine neighborhood pixels.

A simple experiment has been conducted to evaluate the performance of the conventional greedy algorithm and the fast greedy algorithm. The target image used in the experiment is shown in Fig. 2a, while the white dotted circle shown in Fig. 2b is the initial contour. In the experiment, parameter values $\alpha = 1$, $\beta = 1$, and $\gamma = 1$ are used. When the number of control points with changes is less than 10% of the total number of control points, the updating process will be terminated. The search results are illustrated in Fig. 2c and 2d. Clearly, the Snakes obtained using both methods fit the edge of the target nicely after several cycles of iteration.

Table 1 illustrates the performance comparison between the conventional greedy algorithm and the fast greedy algorithm in terms of computation time. Clearly, the computation time for the fast greedy algorithm is shorter than that for the

![Figure 1. Two search modes of the fast greedy algorithm: (a) cross search and (b) diagonal search.](image-url)
Figure 2. (a) Target image. (b) Initial contour of the Snake. (c) Experimental results of the conventional greedy algorithm. (d) Experimental results of the fast greedy algorithm.

Table 1. Performance comparison between the conventional and fast greedy algorithms

<table>
<thead>
<tr>
<th>Weighting $(\alpha, \beta, \gamma)$</th>
<th>Conventional greedy algorithm</th>
<th>Fast greedy algorithm (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0, 1.0, 10)</td>
<td>0.027</td>
<td>0.022</td>
</tr>
<tr>
<td>(1.2, 1.0, 1.0)</td>
<td>0.028</td>
<td>0.021</td>
</tr>
<tr>
<td>(1.0, 1.2, 1.0)</td>
<td>0.024</td>
<td>0.021</td>
</tr>
<tr>
<td>(1.0, 1.0, 1.2)</td>
<td>0.024</td>
<td>0.022</td>
</tr>
</tbody>
</table>

conventional greedy algorithm. It suggests that the fast greedy algorithm is more suitable for real-time dynamic visual tracking applications.

2.4. Automatic initial snake contour generation

Before the Snake starts the iterative process to search for a solution, an initial contour must be provided in advance. In general, the user can determine the initial contour manually through a human–machine interface. However, choosing a proper initial contour is crucial. If the initial contour is too far away from the contour of the target image, very likely the Snake will be trapped into a local minimum during the search process so that it may not converge to the actual contour of the target image.

On the other hand, the adaptive background subtraction approach [10] updates the background model over time. It performs better than the conventional background subtraction method. In general, the moving blobs detected by the background subtraction-based method may contain several tiny holes. In order to obtain a more complete moving blob, one can perform the close operation of morphological filtering [18] on the detected moving blob. If there is more than one moving blob, each moving blob will be labeled and the border tracing technique [11] is employed to detect the contour of each moving blob. The contour of the moving blob that is of interest can be used as the initial contour of the Snake. Two advantages can be obtained by integrating the adaptive background subtraction method with the border tracing technique: (i) the initial contour of the Snake can be automatically generated and (ii) the initial contour is close to the actual contour of the target image so that the Snake likely will converge to the actual contour of the target image.
3. VISUAL TRACKING BASED ON TEMPLATE AND CONTOUR MATCHING

As illustrated in Fig. 3, the visual tracking system developed in this study consists of two operation modes — detection and dynamic tracking. When operated in the detection mode, the adaptive background subtraction method is employed to detect moving objects. If there is more than one object, the visual tracking system has to determine which moving object is the intended one. For this intended moving target, the system generates a target template and also extracts the target contour using the active contour method. On the other hand, when the visual tracking system is operated in the dynamic tracking mode, the proposed visual tracking algorithm is performed to locate the intended moving target. The estimated target position in the image plane is inputted to a g–h filter to obtain a prediction about the position of the moving target in the next frame. The predicted position is used as the feedback signal of the visual loop controller which is designed to lock the moving target’s image in the center of the image plane. Note that the image grabber used in this study provides a sequence of images of size $640 \times 480$ pixel$^2$. In order to speed up the computation process, the image pyramid technique [18] is performed to reduce the size of image from $640 \times 480$ to $160 \times 120$ pixel$^2$.

3.1. Target contour extraction and target template generation

The procedures for target contour extraction and target template generation are elaborated in the following. Consider Fig. 4a, where a picture of car attached to
a small plate is used as the intended moving target. By applying the adaptive background subtraction method, a moving blob is detected as shown in Fig. 4b. Performing border tracing on the image shown in Fig. 4b will yield a rough contour of the moving blob as shown in Fig. 4c. This contour is used as the initial contour of the Snake. By choosing some of the points on the initial contour as the control points and performing the fast greedy algorithm, the contour of the moving target can be obtained as shown in Fig. 4d. The centroid of the Snake contour is considered the center of the target. A rectangular window centered at the centroid of the Snake contour is chosen as the target template (the region inside the white rectangle of size $20 \times 20$ pixel$^2$ in Fig. 4e).

### 3.2. Proposed visual tracking algorithms

In order to improve the robustness of visual tracking, two image features, i.e. target template and target contour, are employed in the proposed visual tracking algorithm.

#### 3.2.1. Modified area-based matching

In general, the conventional area-based matching approach consists of a similarity measurement and a search algorithm. Many reported area-based matching approaches adopted SSD as the similarity measurement. However, even for the case that the moving target does not lie within the search area, the visual tracking algorithm will still output a search minimum. In this case, if the pixel with a search minimum is considered the location of the moving target, a misdetection may occur and eventually lead to a tracking failure.
To deal with this problem, in this study, SDG-based matching [19, 20] is adopted rather than the commonly used SSD.

**SDG-based matching.** The matching error $E_{\Delta u, \Delta v}$ at the pixel $(u, v)$ using the SDG-based matching (Fig. 5) can be expressed as:

$$E_{\Delta u, \Delta v}(u, v) = \left| \sum_{A, B \in D} w(u + \Delta u + A, v + \Delta v + B) \times I_c(u + \Delta u + A, v + \Delta v + B) - I_t(u, v) \right|,$$

where $I_c$ is the current image, $I_t$ is the template, $w$ is the weighting, $D$ represents the neighborhood area of pixel $(u, v)$ and $(\Delta u, \Delta v)$ is the displacement vector.

In (10), the intensity contents of neighborhood pixels are included to reduce the noise effect and thus increase the matching accuracy. The predetermined threshold $k_1$ is used to determine whether the result obtained from (10) is acceptable or not, which is described by:

$$\hat{I}_{\Delta u, \Delta v}(u, v) = \begin{cases} 1, & \text{if } E_{\Delta u, \Delta v}(u, v) \leq k_1 \\ 0, & \text{if } E_{\Delta u, \Delta v}(u, v) > k_1. \end{cases}$$

In (11), if $\hat{I}_{\Delta u, \Delta v}(u, v) = 1$, the pixel $(u, v)$ belongs to the moving target, otherwise the pixel $(u, v)$ belongs to the background. Note that (11) is applied to every pixel inside the search window to obtain a binary image after matching.

The similarity between the template and a candidate target region in the current image frame can be defined as:

$$S_f(\Delta u, \Delta v) \equiv \sum_{u, v \in I_t} \hat{I}_{\Delta u, \Delta v}(u, v).$$

A larger value of $S_f(u, v)$ corresponds to a higher similarity between the template and the candidate target region.
On the other hand, existing search algorithms include the full search, the three-step hierarchical search [21], the diamond search method [22], etc. In this study, the full search method is used. However, performing ‘full search’ on the entire image may require a considerable computation time such that real-time visual tracking is not achievable. In order to prevent from this happening, only the pixels inside the widow centered at the target position in the previous image will be searched. In this study, the size of the search window is set to $50 \times 24$ pixels. By performing the full search, the goal is to find the displacement vector so that (12) assumes the maximum similarity, i.e.:

$$\begin{align*}
(\Delta u, \Delta v) &= \arg \max_{\Delta u, \Delta v \in W} S_f(\Delta u, \Delta v), \quad \text{if } S_f(\Delta u, \Delta v) \geq s_1,
\end{align*}$$

(13)

where $W$ is the search window and $s_1$ is the predetermined threshold that can be used to determine whether the moving target lie within the search window or not.

**Template update with memory.** As mentioned previously, for area-based matching, a fixed template may lead to a tracking failure. Lipton et al. [10] proposed a template updating method to deal with this problem. However, when an object occludes the moving target, the Lipton’s approach will result in a tracking failure. To cope with this difficulty, the template updating method is modified by adding an initial template [20], which is described by:

$$P_{k+1} = \frac{1}{2} \left( \lambda I_0 + (1 - \lambda) I_k + P_k \right), \quad 0 < \lambda < 1,$$

(14)

where $P_{k+1}$ is the template after updating, $\lambda$ is the similarity coefficient, $I_0$ is the initial template, $I_k$ is the current image and $P_k$ represents the previous template.

With the inclusion of the initial template $I_0(u, v)$, even under the scenario that the moving target is occluded by an object completely, the proposed approach can still track the moving target.

**Template mask.** In general, the template used for tracking consists of a moving target and some background contents. If the background contents occupy too great a portion of the template, it may jeopardize the visual tracking process. One possible approach to tackle this problem is to use a template mask [20]. Namely, the portion of the template used in similarity matching is the moving target itself. In this way, the pixel belonged to the background will be skipped during the matching process so that the computing efficiency can be much improved. Details concerning the template mask are explained in the following.

Consider two consecutive templates $P_k(u, v)$ and $P_{k+1}(u, v)$ at the $k$th and $(k + 1)$th time instants, respectively. The image difference between $P_{k+1}(u, v)$ and $P_k(u, v)$ can be expressed as:

$$m(u, v) = P_{k+1}(u, v) - P_k(u, v)$$

$$= (1 - \lambda) \left\{ \frac{1}{2} I_k(u, v) - \frac{1}{2} \sum_{n=1}^{k-1} \frac{1}{2^{k-n}} I_n(u, v) \right\}.$$

(15)
Equation (15) suggests that if \( m(u, v) \approx 0 \), then the pixel \((u, v)\) belongs to the moving target, otherwise it belongs to the background. The above observation can be expressed as

\[
P_{\text{mask}} \equiv \begin{cases} 
1, & \text{if } m(u, v) \leq \varepsilon \\
0, & \text{if } m(u, v) > \varepsilon, 
\end{cases} \tag{16}
\]

where \( P_{\text{mask}}(u, v) \) is the binary image referred to as the template mask and \( \varepsilon \) is a prescribed threshold. Only the pixels with \( P_{\text{mask}}(u, v) = 1 \) need to perform the SDG-based similarity matching. That is, (10), (11) and (13) are constrained by the condition: \( \{(u, v) \mid P_{\text{mask}}(u, v) = 1\} \).

### 3.2.2. Snake-based contour matching.

Before proceeding contour matching, the edges of the current image must be obtained in advance. The Gaussian filtering is performed on the obtained edge image to eliminate the noise effect (Fig. 6a). Full search is then performed to find the pixel location that yields the maximum similarity between the Snake and the obtained edge image (Fig. 6b). The gradients for all control points of the Snake are calculated. For each pixel location, the total number of control points with gradients larger than the predetermined threshold \( k_2 \) can be computed using (17) and (18):

\[
\hat{G}_{\Delta u, \Delta v}(i) = \begin{cases} 
1, & \text{if } g_{\Delta u, \Delta v}(i) \geq k_2 \\
0, & \text{if } g_{\Delta u, \Delta v}(i) < k_2, 
\end{cases} \tag{17}
\]

\[
S_{\hat{G}}(\Delta u, \Delta v) \equiv \sum_{i=1}^{n} \hat{G}_{\Delta u, \Delta v}(i), \tag{18}
\]

where \((\Delta u, \Delta v)\) is the displacement vector and \( g_{\Delta u, \Delta v}(i) \) is the gradient of the \( i \)th control point.

A larger \( S_{\hat{G}}(\Delta u, \Delta v) \) represents a higher similarity between the Snake and the obtained edge image. The pixel location that yields the largest \( S_{\hat{G}}(\Delta u, \Delta v) \) is considered the most likely moving target position. Consequently, the displacement

---

**Figure 6.** Illustrative example of contour matching. (a) Edge image. (b) Full search is performed to find the pixel location that yields the maximum similarity between the Snake and the obtained edge image.
vector associated with the moving target location can be expressed as:

\[(\Delta u, \Delta v) = \arg \max_{\Delta u, \Delta v \in W} S_G(\Delta u, \Delta v), \quad \text{if } S_G(\Delta u, \Delta v) \geq s_2, \quad (19)\]

where \(s_2\) is a predetermined threshold used to determine whether the moving target lie within the search window or not.

### 3.3. Integration of template matching and contour matching

Both the template and contour matching are employed in the proposed visual tracking algorithm. The moving target position is estimated using a weighted average of \(S_i\) and \(S_G\), where:

\[X = \arg \max_{\Delta u, \Delta v \in W} R(\Delta u, \Delta v).\]  

In (20):

\[R(\Delta u, \Delta v) = (r_1(t)\bar{S}_i(\Delta u, \Delta v) + r_2(t)\bar{S}_G(\Delta u, \Delta v)), \quad (21)\]

where \(R\) is the total similarity, \(\bar{S}_i\) and \(\bar{S}_G\) are the corresponding values of \(S_i\) and \(S_G\) after normalization, and \(r_1(t)\) and \(r_2(t)\) are the associated weightings for \(\bar{S}_i\) and \(\bar{S}_G\), respectively.

In (21):

\[
\begin{align*}
\bar{S}_i(\Delta u, \Delta v) &= \frac{S_i(\Delta u, \Delta v)}{m}, \\
\bar{S}_G(\Delta u, \Delta v) &= \frac{S_G(\Delta u, \Delta v)}{n},
\end{align*}
\]

where \(m\) and \(n\) represent the total number of pixels inside the template mask and the total number of control points of the Snake, respectively.

The values of \(r_1(t)\) and \(r_2(t)\) are adjusted using the self-organized approach [6]. The key is to automatically adjust the values of \(r_1(t)\) and \(r_2(t)\) according to the qualities of template and contour matching results. To do so, define the quality functions for template matching and contour matching as:

\[
q_i(t) = \text{ramp}(\bar{S}_i(X) - \langle \bar{S}_i(\Delta u, \Delta v) \rangle_{\Delta u, \Delta v \in W}),
\]

\[
q_G(t) = \text{ramp}(\bar{S}_G(X) - \langle \bar{S}_G(\Delta u, \Delta v) \rangle_{\Delta u, \Delta v \in W}),
\]

where \(\langle \cdot \rangle\) represents the average value and \(\text{ramp}(\cdot)\) represents the ramp function:

\[
\text{ramp}(x) = \begin{cases} 
0, & \text{if } x \leq 0 \\
x, & \text{if } x > 0.
\end{cases}
\]

If the value of the quality function for template matching is larger than that for contour matching, this indicates that template matching provides a better outcome, hence the weighting of template matching should be increased and vice versa.
Performing normalization on $q_i$ and $q_G$, one will have:

\[
\bar{q}_i = \frac{q_i}{q_i + q_G}, \\
\bar{q}_G = \frac{q_G}{q_i + q_G}.
\]  

(25)

Now $r_1(t)$ and $r_2(t)$ can be adjusted dynamically based on:

\[
\tau \dot{r}_1(t) = \bar{q}_i - r_1(t), \\
\tau \dot{r}_2(t) = \bar{q}_G - r_2(t),
\]  

(26)

where $\tau$ is the time constant.

The flowchart for the integration of template matching and contour matching is illustrated in Fig. 7.

### 4. VISUAL SERVO STRUCTURE AND g–h FILTER

To evaluate the performance of the proposed approach, a real-time pan–tilt dynamic visual tracking system developed in our lab is used as the test platform. The dynamic visual tracking system comprises of a CCD camera mounted on a pan–tilt unit which is powered by two AC servomotors, and a personal computer (with Intel Celeron 2.4 GHz CPU) that is used as the control kernel and also as the user/camera...
interface platform. It can be operated in two different modes, i.e. detection and dynamic tracking. Throughout all the experiments, the visual tracking system is assumed to be operated in the detection mode initially. Namely, the CCD camera is initially at rest. When the visual tracking system is turned on, the CCD camera starts to capture images and the adaptive background subtraction method [10] is used to perform motion detection. If a moving target is detected, a rectangular window that contains the detected moving target is selected as the target template. The visual tracking system will be switched to the dynamic tracking mode, where the proposed visual tracking algorithm will be employed to track the moving target. The servo control structure adopted in the dynamic tracking mode belongs to the category of image based and dynamic-look-and-move [1]. The block diagram of the servo control structure is illustrated in Fig. 8, where $x_t$ is the current target location; $f$ is the focal length, $Z$ is the depth of the target and $L$ is the distance between the lens center and the rotational center of the pan–tilt unit. The motion of the pan–tilt unit is controlled to track the moving target so that the target’s image will be locked in the center of the screen. The outer visual loop controller includes a P-type controller and a feedforward controller.

Most commercial image grabbers are only capable of providing a frame rate of 30 Hz, i.e., the sampling time for the visual loop is 33 ms. If the sampling period for the inner servo loop is set to 33 ms, the latency in the outer visual loop may

Figure 8. Block diagram of the servo control structure used in this study.
lead to serious deteriorations in tracking performance. In order to cope with this difficulty, linear interpolation on vision commands is performed to realize a single-rate ($T = 1 \text{ ms}$) servo control for the dynamic visual tracking system. Moreover, prediction of the visual feedback signal based on the g–h filter [23] is used to deal with the latency problem in the visual loop. The g–h filter shown in Fig. 8 consists of two parts—prediction and updating. Equation (27) is used to update the current target velocity and position, while (28) is used to predict the target velocity and position at the next time instant under the assumption that the target has a constant velocity motion:

$$\begin{align*}
\dot{\theta}_{n,n}^* &= \dot{\theta}_{n,n-1}^* + h \left( \frac{\theta_{tn} - \theta_{n,n-1}^*}{T} \right), \\
\theta_{n,n}^* &= \theta_{n,n-1}^* + g(\theta_{tn} - \theta_{n,n-1}^*), \\
\dot{\theta}_{n+1,n}^* &= \dot{\theta}_{n,n}^*, \\
\theta_{n+1,n}^* &= \theta_{n,n}^* + T \theta_{n+1,n}^*.
\end{align*}$$

Combining (27) and (28) will yield the mathematical expression for the g–h filter:

$$\begin{align*}
\dot{\theta}_{n+1,n}^* &= \dot{\theta}_{n,n-1}^* + \frac{h}{T} (\theta_{tn} - \theta_{n,n-1}^*), \\
\theta_{n+1,n}^* &= \theta_{n,n-1}^* + T \dot{\theta}_{n+1,n}^* + g(\theta_{tn} - \theta_{n,n-1}^*),
\end{align*}$$

where $\dot{\theta}_{n,n-1}^*$ and $\theta_{n,n-1}^*$ represent the target velocity and position updated at the $n$th sampling time, respectively. $\theta_{tn}$ is the measured target position, $T$ is the sampling interval, $g$ and $h$ are constant weightings for the g–h filter.

Note that, in practical implementations, the g–h filter described by (29) cannot be used directly due to the fact that the current target position cannot be measured directly. To cope with this difficulty, the following implementation procedures are adopted in this paper:

(i) Calculate the value of $\tilde{\theta}_n$ using the relation $\tilde{\theta}_n = \frac{\tilde{z}}{f(z+L)} u_n$.

(ii) Based on the value of $\tilde{\theta}_n$, compute $\theta_{tn} = \tilde{\theta}_n + \theta_c n$.

(iii) Substitute the value of $\theta_{tn}$ into (28) to calculate $\theta_{n+1,n}^*$.

(iv) Based on the value of $\theta_{n+1,n}^*$, compute $\tilde{\theta}_{n+1,n} = \theta_{n+1,n}^* - \theta_c n$.

(v) Calculate the value of $u_{n+1}^*$ using the relation $u_{n+1}^* = \frac{f(z+L)}{\tilde{z}} \tilde{\theta}_{n+1,n}^*$.

In addition, the values of the parameters used in all experiments are listed as follows: $k_1 = 15$, $s_1 = 0.3$, $\lambda = 0.6$, $\varepsilon = 15$, $k_2 = 80$, $s_2 = 0.3$, $\tau = 100$ and $f = 800$ pixels. Note that after image pyramid processing, $f$ becomes 200 pixels. On the other hand, the gain constant for the pan axis is 5.2, while the gain constant for the tilt axis is 2.7. The feedforward controller gains for the pan axis and the tilt axis are 35 and 10, respectively. The parameters for the g–h filter are: $g = 0.7$ and $h = 0.377$. 
5. EXPERIMENTAL RESULTS

Four experiments are conducted to evaluate the performance of the proposed approach.

5.1. Visual tracking without occlusion

In this experiment, the system is operated in the detection mode at first. It is assumed that the intended moving target, a ‘human head’, is not occluded throughout the experiment. The target template is obtained using the adaptive background subtraction method. After the target template is acquired, the system is switched to the dynamic tracking mode, i.e., the servomotors will be turned on and the pan-tilt unit is controlled to lock the image of the target in the center of the screen. In the dynamic tracking mode, two different visual tracking algorithms, i.e., the modified area-based matching (SDG-based matching + template updating with memory) and the proposed approach, are employed. Figure 9 illustrates the experimental results of the modified area-based matching method. The results show that the visual tracking system indeed can keep the image of the moving target around the center of the screen. Fig. 10 illustrates the experimental results of the proposed approach. The white dot contour in Fig. 10 is the Snake contour. From Fig. 10a–d, it is found that some parts of the Snake converge to an incorrect contour (neckline). The reason is that there is a strong edge content between the neckline and the head. However, the proposed visual tracking algorithm also contains the information provided by the template matching cue. Consequently, it tracks the target correctly even though the Snake converges to an incorrect contour.

Figure 9. Tracking sequence of the modified area-based matching (the white rectangle is the target template).
5.2. Visual tracking with occlusion

In this experiment, the human head is partially or completely occluded by an object. Figure 11 illustrates the experimental results of the modified area-based matching. In Fig. 11b and 11e, the human head is occluded completely. If the conventional area-based matching method is employed, a tracking failure may occur. In this experiment, template updating with memory is employed to deal with this problem. In addition, the SDG-based matching is adopted rather than the commonly used SSD. If the similarity between the occluding object and the template is smaller than
a predetermined threshold ($s_1$ in (13)), the visual tracking system will not perform tracking so that tracking failures can be prevented. The results in Fig. 11 show that the visual tracking system using the modified area-based approach can track the moving target correctly even though the moving target is occluded by an object.

Figure 12 illustrates the experimental results of the proposed method. In the experiment, the similarity index for the template matching is smaller than a
predetermined threshold due to the fact that the contour of the occluding object and
the contour of the human head are different. Hence, not only the target template will
not be updated, but also the updating process for the Snake contour will be halted
temporarily to prevent from tracking failures. For instance, in Fig. 12b and 12e,
even though the human head is occluded completely, the Snake contour will not be
updated. When the human head is occluded by some objects, both the similarity
indices for template matching and contour matching will decrease. If any one of
the similarity indices is smaller than the threshold, the tracking window will simply
remain at the original position until the moving target reappear in the scene.

5.3. Visual tracking for the case of the moving target blocked by an object with
similar intensity

In area-based matching, intensity is the only image content considered. If the
intensity of the occluding object is similar to that of the moving target, a tracking
failure likely will occur. In Fig. 13, the visual tracking system that utilizes modified
area-based matching originally tracks a person in striped clothes, where the target
template is the white rectangle. In Fig. 13c, there is a person in white clothes
walking through the scene. Since the skin colors of these two persons are similar,
when the person in white clothes occludes the person in striped clothes, the target
template will be gradually updated and eventually leads to a tracking failure as
shown in Fig. 13d–f.

Figure 14 shows the experimental results of the proposed approach. Again, there
is a person in white clothes walking through the scene. Even though similar intensity
diminishes the performance of template matching, the contour of the person and the

![Figure 13](image1.jpg)

(a) (b) (c)

(d) (e) (f)

**Figure 13.** Tracking sequence of the modified area-based matching approach under the case that the
moving target is occluded by an object with similar intensity.
Snake contour are not similar. As a result, the proposed approach can correctly track the moving target. Experimental results shown in Figs 13 and 14 suggest that the proposed approach is more robust to noise than the area-based matching.

5.4. Visual tracking with a g–h filter

In this experiment, the moving target (a picture of car) is attached to a linear motor that is controlled to perform a 0.5-Hz repetitive motion with stroke of 44 cm (Fig. 15). The distance between the rotational center of the pan–tilt unit and the linear motor is 150 cm. With visual feedback, the tracking system is controlled to lock the target’s image in the center of the screen.

A total of four dynamic visual tracking experiments are conducted with/without the g–h filter. In all four experiments, the proposed visual tracking approach is employed. In addition, the moving target performs a horizontal motion; hence, only the tracking performance in the pan axis will be discussed. The tracking error is defined as the distance from the center of the image plane to the center of the target image. Experimental results of dynamic visual tracking without the g–h filter are listed in Table 2. According to Table 2, it is found that there are 70% of frames with tracking errors less than 10 pixels, while there are 91% of frames with tracking errors less than 12 pixels. Experimental results of dynamic visual tracking with the g–h filter are listed in Table 3. According to Table 3, it is found that there are 84% of frames with tracking errors less than 10 pixels, while there are 97% of frames with tracking errors less than 12 pixels. Clearly, the tracking performance has been improved after adding the g–h filter to the visual loop.
Figure 15. Moving target (a picture of car) is attached to a linear motor.

Table 2.
Experimental results of visual tracking without the g–h filter

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Total frames</th>
<th>Error &lt;10 pixels (%)</th>
<th>Error &lt;12 pixels (%)</th>
<th>Average error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1382</td>
<td>69.03%</td>
<td>93.05%</td>
<td>8.4421</td>
</tr>
<tr>
<td>2</td>
<td>1467</td>
<td>71.23%</td>
<td>90.73%</td>
<td>8.4145</td>
</tr>
</tbody>
</table>

Table 3.
Experimental results of visual tracking with the g–h filter

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Total frames</th>
<th>Error &lt;10 pixels (%)</th>
<th>Error &lt;12 pixels (%)</th>
<th>Average error (pixel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1123</td>
<td>83.70%</td>
<td>97.77%</td>
<td>7.4862</td>
</tr>
<tr>
<td>2</td>
<td>1981</td>
<td>84.91%</td>
<td>97.32%</td>
<td>7.4074</td>
</tr>
</tbody>
</table>

6. CONCLUSIONS

This paper focuses on exploring the problems of dynamic visual tracking using active cameras. To improve the tracking performance, a visual tracking algorithm that combines modified area-based matching with Snake-based contour matching was developed, in which the moving target position is estimated based on a weighted average of area-based matching and contour matching similarity. In addition, to facilitate the use of contour matching, a new approach that can automatically generate the initial contour is also developed. Moreover, the contour matching is used in combination with the fast greedy algorithm so that it can be applied to the problem of dynamic visual tracking using active cameras. On the other hand, in order to cope with the latency problem that often occurred in the visual tracking system, a g–h filter is added to the visual loop to provide a prediction on the target location in the next frame so that tracking performances can be improved. Several experiments have been conducted to evaluate the performance of the proposed approach. Experimental results show that compared with the area-based matching
The proposed approach exhibits a better tracking ability. Experimental results also show that the visual tracking system with the g–h filter is superior to the visual tracking system without the g–h filter in terms of tracking performance.

Acknowledgements

The authors are grateful to the National Science Council of the Republic of China for supporting this research under grant NSC91-2213-E006-122. Special thanks are dedicated to Mr. C.-K. Wang for his assistance with this work.

REFERENCES


**ABOUT THE AUTHORS**

**Ming-Yang Cheng** was born in Taiwan, in 1963. He received the BS degree in Control Engineering from the National Chiao-Tung University, Taiwan, in 1986. He received the MS and PhD in Electrical Engineering from the University of Missouri-Columbia, USA, in 1991 and 1996, respectively. From 1997 to 2002, he held several teaching positions at the Kao Yuan Institute of Technology, Dayeh University, and the National Kaohsiung First University of Science and Technology, respectively. In 2002, he joined the Department of Electrical Engineering at the National Cheng Kung University, Taiwan, where he is currently an Associate Professor. His research interests include visual servoing, motion control, motor drive and biped locomotion.

**Mi-Ching Tsai** was born in Taiwan, in 1956. He received both the BS and MS degrees in Electronic Engineering from the National Taiwan Institute of Technology, in 1981 and 1983, respectively. He received his PhD from the Department of Engineering Science, Oxford University, in 1990, and became a full Professor in the Department of Mechanical Engineering at National Cheng Kung University, Taiwan, in 1996. He was a Visiting Professor at the Engineering Department (Control Group) of Cambridge University from 2003 to 2004. He is currently the Director of the NCKU Electrical Motor Technology Research Center. His research interests include robust control, servo control, motor design, motor control and applications of advanced control technologies using DSP. He is a Senior Member of the IEEE and also a Fellow of the IEE. In addition, he has served as an Associate Editor of the *IEEE/ASME Transactions on Mechatronics*.

**Chia-Yang Sun** was born in Taiwan, in 1980. He received the BS and MS degrees in Mechanical Engineering from the National Cheng Kung University, Taiwan, in 2004. He joined the National Center for High-Performance Computing in 2005, where he is currently an Assistant Researcher in the Visualization and Interactive Media Lab. His research interests include visual tracking, image processing and motion control.