CHAPTER 3

Pose Measurement Based on Vision Perception

Autonomous rendezvous and docking (ARVD) is necessary for planned space programs such as space attack and defense, on-orbital servicing [1,2], and other rendezvous and proximity operations. Recently, there have been some different solutions to realize measurement, such as microwave radar, laser radar, satellite navigation, and vision-based measurements. All of them have been used or tested for ARVD. The PRISMA mission has demonstrated a navigation technology based on both a passive optical system [3] and an active RF system [4]. The passive optical system has shown relatively precise reconstruction of the distance to target. It has also demonstrated the reconstruction of the "pose" of the target, meaning the reconstruction of its quaternions (based on predefined target geometry). The navigation concept of OLEV is to use ranging for absolute navigation and to hand over to relative navigation at the distance of a two kilometer. For relative navigation, a set of six rendezvous cameras (far, mid- and close-range, and redundant) are used. For DEOS, the sensor package consists of the following individual sensor arrangements: 2 LIDAR heads, 2 far range mono cameras, 1 mid-range stereo camera, 1 close range stereo camera, and 1 docking mono camera. From their reports, we can see the task is far from trivial though much research has worked for decades to address it.

Our system's design philosophy is dexterous, light, and economical, which also decides how to select the sensors for ARVD. After a comprehensive analysis, we built a visual perception system based on four charge-coupled device (CCD) cameras because they are becoming more commonly used in rendezvous missions for their relatively low cost, low mass, and energy requirement [5].

In the above mission scenario, it has been described that a Robot Platform begins to detect, track, and measure the noncooperative target based on its own binocular stereo system when it arrives at point S3 about 1 km away from the target. The acquired relative pose information helps the Robot Platform fly to the target accurately. After a 800 m

Tethered Space Robot
https://doi.org/10.1016/B978-0-12-812309-6.00003-8

approach, the Robot Platform arrives at point S4 about 200 m away from the target. Then, the Operational Robot will be launched out and fly toward the frontal target freely. In this phase, the Robot Platform has to measure the relative pose information between the noncooperative target satellite and the cooperative Operational Robot. The information will be directly used to determine whether the Operational Robot can approach the noncooperative target opportunely. Once their distance reduces to 20 m, the Robot Platform cannot supply their relative pose information because of factors such as occlusion of the viewing field. Immediately, the visual perception subsystem mounted on the Operational Robot has to measure the information autonomously. Furthermore, it has to detect the capture ROI to guide 3-fingers for precise manipulation. The involved core technologies contain cooperative object recognition, recognition of capture ROI, and accurate 3D pose estimation.

Considering that the shape of the target satellite's main body is usually cubical, polyhedral, or cylindrical, these satellites can naturally be photographed by optical sensors. When the target satellite is coupled with advanced image processing and estimating algorithms, it achieves cost-effective and accurate sensing capability to capture and track the target in time to obtain the full 6 degrees of freedom (DOF) relative pose information during the proximity operation.

3.1 MEASUREMENT SYSTEM SCHEME

The robotic sensing system is composed of different types of sensors that enable robots to have the preliminary ability of sensing similar to humans. Microwave radar, laser radar, satellite navigation, and vision-based measurements are currently used for relative navigation. The charge-coupled device (CCD) camera is becoming more commonly used for relative navigation in rendezvous and docking missions for its relatively low cost, low mass and energy requirement [6].

In this section, we will propose a visual perception system for the TSR's automatic rendezvous from 100 to 0.15 m. The core problem, which is tracking the whole contour of noncooperative moving targets in real-time, will be emphasized in Section 3.2.

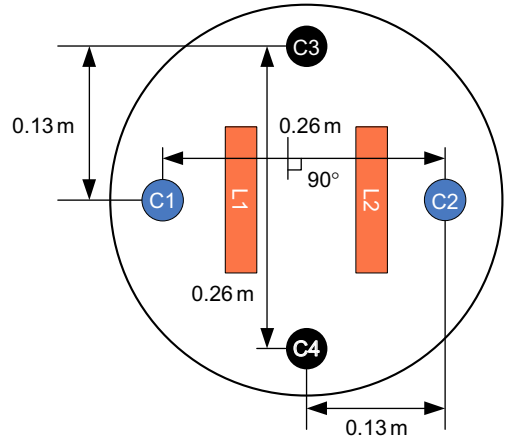


Fig. 3.1 Construction of the visual perception system for TSR.

In our proposed system, the visual perception system mounted on the operational robot is divided into three components, as shown in Fig. 3.1. The system involves four low-cost CCD cameras, the focal length of which is 12 mm and the image size is 704×576 pixels. Two of these cameras (C1, C2), called far-range detectors, are deployed lenses with 20° field of view (FOV) angles. The other two cameras (C3, C4), called close-range detectors, are deployed lenses with 40° FOV angles. The baselines of these collaborative detectors are the same at 0.26 m. Two light emitting diodes (LEDs) (L1, L2) are used for illumination compensation. Specifically, when the TSR is at a distance of 100–20 m away from the target during the approaching phase, two azimuth angles obtained via C1 or C2 are utilized to control the position deviations of two corresponding directions. When the distance is 20–0.2 m, the information of relative position and attitude obtained via binocular vision (C3 and C4) can be used to guide the TSR to move to the desired capture position of target.

Specifically, the baseline of (C3, C4) is limited to 0.26 m. The binocular stereo cameras are generally deployed with a parallel optical center. In order to lengthen the binocular cameras' measurement distance, C3 and C4 are designed to deploy with a 10° inline angle (Included angle).

Our designed visual perception system supplies different measurement information in different listed following,

Table 3.1 Information required during the approach

From 100 to 20 m	From 20 to 0.15 m
Object's azimuthal angle (By C1 or C2)	Object's azimuthal angle (By C3 or C3)
Object's estimated relative distance (By C1 and C2)	Object's relative distance (By C3 and C4)
Object's 2D contour information (By C1 or C2)	Object's relative attitude angle (By C3 and C4)
Object's least circumscribed rectangle (By C1 or C2)	Object's relative velocity (By filter)
Estimated centroid value of Object's main flat (By C1 or C2)	Object's relative angular velocity (By filter)
	Object's best grapple position (By C3 and C4)

Once the TSR is released, C1 and C2 are open to work. From Table 3.1, we can see that the far-range detectors (C1, C2) are used to acquire object's azimuthal angle from 100 to 20 m. And as a binocular system, they can supply the object's estimated relative distance coarsely. When the estimated relative distance reaches about 25 m, the synthesized system in the TSR activates C3 and C4 open to work. Once the object's relative distance is about 20 m measured by C3 and C4, C1 and C2 are shut down.

In addition, the target is noncooperative, which means that the detection, tracking, and pose measurement of the spacecraft is a challenging task in autonomous rendezvous and docking. 3D space points can be transformed by CCD sensors into 2D features in photographed images. Thus designing a visual perception algorithm is necessary to exploit the natural features extracted from the noncooperative target. Thus, from 100 to 20 m, image sequences are acquired by the far-range detectors (C1, C2) and used to detect, segment, and track the target and to realize the measurement of the range, angle of pitch, and orientation. From 20 to 0.15 m, the main body can be photographed by the close-range detectors (C3, C4). In addition, the stereo vision-based approach is applied to recognize the vertexes of the main body contour. The pose parameters with respect to the vision frame can then be measured. Fig. 3.2 shows the task flow of the visual perception system during the whole mission.

As shown in Fig. 3.2 from 100 to 20 m, the far-range detectors (C1, C2) with 20° FOV angles acquire image sequences to detect, segment, and track the target and to realize the measurement of range, angle of pitch, and

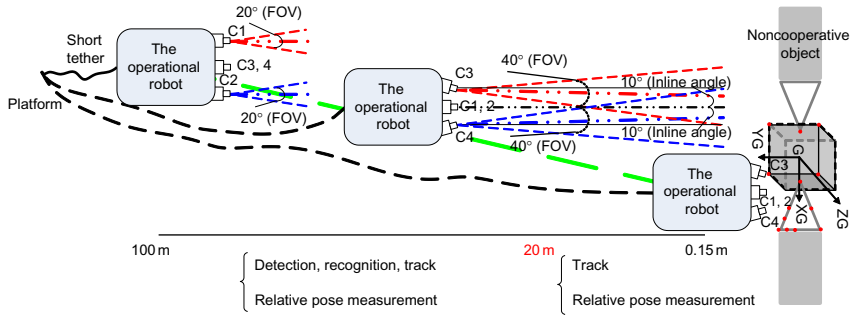


Fig. 3.2 Task flow of the visual perception system.

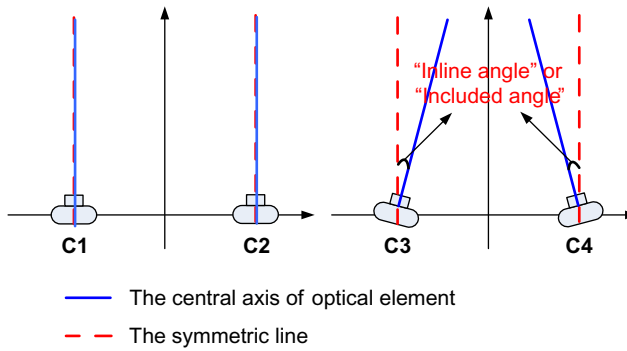


Fig. 3.3 An explanation of inline angle or included angle.

orientation. From 20 to 0.15 m, the main body can be photographed by the close-range detectors (C3, C4) with 40° FOV angles. The pose parameters corresponding to the vision frame can then be measured.

The 10° FOV mentioned above is "Inline angle" or "Included angle" from the central axis of optical element to symmetric line (shown in Fig. 3.3).

3.2 TARGET CONTOUR TRACKING

In this section, a novel monocular real-time robust feature tracking algorithm called MRRFT is proposed, which is based on the combination of the improved SURF, P-KLT matching and Greedy Snake algorithms. We first evaluate six state-of-the-art feature extraction algorithms and compare the SIFT with SURF in detail. Then we propose our MRRFT as follows:

(1) The algorithm establishes the target model using the features extracted by our improved SURF and the contour extracted by the Greedy Snake algorithm, after which it dynamically updates these features

according to our strategy to ensure a precise model during the sequence.

(2) The algorithm assumes that the displacement between frames is extremely small, eliminates mismatched points by utilizing a statistical method, and designs a discrete feature filter to achieve high tracking accuracy.

Finally, our MRRFT is tested on simulation data generated by Vega Prime and natural images collected from our library. The results show that the proposed algorithm works well when tracking the satellite model. The MRRFT is robust to global and local changes in the object's appearance attributed to pose, scale, and illumination variations. For comparison, we evaluated the GFTT + P-KLT, SIFT + P-KLT, and our method under the same conditions and found that MRRFT is more accurate than GFTT + P-KLT, and significantly faster and more accurate than SIFT + P-KLT.

3.2.1 Related Works

Current tracking algorithms [6a] can be commonly classified into three categories: point tracking, kernel tracking, and silhouette tracking [6b–11]. These three categories represent the target with various models and update the location of a target in the camera image sequence using different procedures.

Selecting an appropriate feature is an important step in tracking. The most desirable property of a visual feature is its uniqueness, which enables the objects to be easily distinguished in the feature space. In 1999, Lowe [12] presented scale-invariant feature transform (SIFT) for extracting distinctive invariant features from images that can be invariant to image scale and rotation. He approximated the Laplacian of Gaussian (LoG) by a Difference of Gaussians (DoG) filter in order to improve the speed,

In 2005, Bay [13] proposed a novel detector-descriptor scheme and coined the term SURF in a similar manner. The detector is based on the Hessian matrix. However, the detector employs a very basic approximation to the LoG by using box filter representations of the respective kernels, just as DoG is a very basic Laplacian-based detector. Experiments showed that SURF is three times faster than SIFT. However, the two methods exhibit good performance similarly.

Many researchers have recently proposed numerous distinguishing feature point detectors and descriptors, such as rotation invariant fast features (RIFF) [14], binary robust independent elementary features (BRIEF) [15], oriented fast and rotated BRIEF (ORB) [16], and LAZY. However,

selecting an appropriate algorithm that achieves extraction from noncooperative satellites that are not rich in texture is difficult. Therefore we initially conduct simulations to test the performance of the detectors.

The well-known KLT was proposed by Lucas and Kanade and fully developed by Tomasi and Kanade [17]. This algorithm was developed to track point features for image registration or stereo correspondence. It is widely used in object tracking. The KLT algorithm is a typical approach that uses the information between continuous image frames.

Mian [17a] presented a modified KLT algorithm for tracking one or more objects. The tracking algorithm is based on local features and continuously updates the features while tracking. Rabaud [18] developed a highly parallelized version of the KLT tracker that combined graph cuts and random sample consensus (RANSAC) to count crowd objects. Sugimura [19] proposed a human tracking method with trajectory-based clustering and a gait feature based on KLT corresponding points and Delaunay triangulation. Tsuduki [20,21] used mean-shift searching to track a point based on the information obtained by a SIFT. Moreover, Tsuduki achieved improved tracking performance relative to KLT because a SIFT feature is invariant to changes caused by rotation, scaling, and illumination. Ben [22] described and demonstrated a scalable real-time system for obtaining stable head images from pedestrians in high-definition surveillance videos. Gong [23] proposed a moving vehicle detection algorithm based on motion vectors that realizes the detection of moving objects against dynamic backgrounds using the KLT matching algorithm and K-medoids clustering algorithm.

Liu [24] proposed a new feature point tracking algorithm based on a combination of the SIFT and KLT matching algorithms, which can track a target when it changes in size and attitude stably and accurately. The new algorithm has stronger robustness and reliability and obtains more accurate target locations than the KLT tracking algorithm.

Although progress has been made in recent years, the robustness, stability, and precision of tracking a noncooperative satellite with a plain surface and simple structure in space remains a challenging problem. Aiming to overcome the disadvantages of the proposed methods, we developed a method called monocular real-time robust feature tracking algorithm (MRRFT) based on a combination of an improved SURF algorithm, a pyramid KLT matching algorithm, and the Greedy Snake algorithm. The disadvantages of the proposed method that we aim to overcome include the incapability of the good features to track (GFTT) feature detector to extract the corner accurately and precisely because of rapid relative movement.

Meanwhile, SIFT's extraction time is extremely long, and the method lacks adaptive update strategies in paper [24].

We introduce the SURF for target tracking and improve it to extract features more symmetrically and robustly. Compared with GFTT, SURF can overcome noise sensitivity and location errors. Combined with the P-KLT matching algorithm, the extracted features can be tracked in real-time while being invariant to image scale, rotation, and illumination. The acquired matching points are set as the initial contour points. Then, the Greedy Snake algorithm is used to segment the object to acquire precise target locations. Furthermore, we designed a discrete feature filter and adaptive feature updating the strategy to enhance robustness. A series of simulations and experiments demonstrate that the proposed tracking system can adapt to objective changes in attitude and size and stably track noncooperative satellites in real-time.

3.2.2 Feature Extraction

3.2.2.1 Simulation Comparisons

We examine SIFT, SURF, RIFF, BRIEF, LAZY, and ORB to select the appropriate extraction algorithm based on Eugene's experiments. And the results are similar to those of Eugene.

All quality tests employed similar procedures and were implemented in C++ using the OpenCV2.3.1. We used the same image database, which includes the general deformations such as scale changes, blur changes, and rotation. The transformation algorithm depends on a particular test. In the rotation test case, the source image rotates around its center at 360° with 30° steps. For scaling, the image is resized from 0.25× to 2.2× similar to the original. The blur test used a Gaussian blur, and the Gaussian kernel was resized from 1× to 19× and the step is 2×. The comparative diagram is shown in Fig. 3.4.

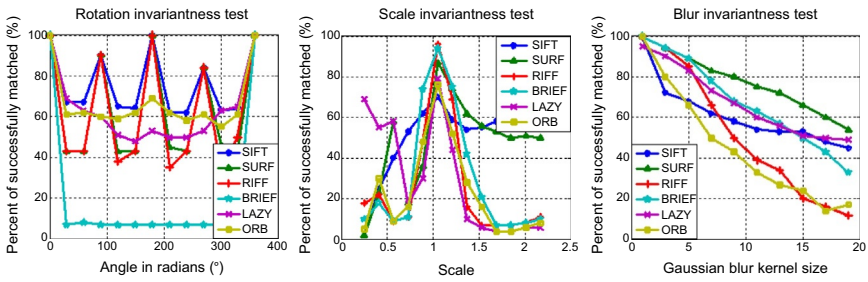


Fig. 3.4 Comparative diagram of features extraction invariance tests.

Fig. 3.4 shows that SIFT, ORB, LAZY, and SURF have good performance in the rotation test. SIFT and SURF show good performance in the scaling test, whereas SURF, BRIEF, LAZY, and SIFT show good performance in the blur test. Therefore, SIFT and SURF are more suitable for the aim of this part.

Then we compare SIFT and SURF in detail. We use the same image database (four images are selected from the standard dataset and two images are personally photographed), the sizes are all 512 pixels × 512 pixels. Fig. 3.5 shows that an average of 682.5 features per image are extracted by SURF compared with 490.5 features extracted by SIFT. The average detection time of SURF is 583 ms per image on average compared with 3042 ms for SIFT. We can then determine that the extraction time of SURF for one feature is 0.85 ms on average, whereas that of SIFT is 6.2 ms.

As we can see, SURF works rapidly and efficiently in most situations. If SURF is used to detect the target box, which comprises only a small block of the whole image, less time would be consumed. Thus, combined with P-KLT, the proposed algorithm can realize target detection and tracking in real-time.

3.2.2.2 Description of SURF

Considering that Gaussian filters are nonideal in any case and given Lowe's success with LoG approximations, SURF creates a "stack" without 2:1 down sampling for higher levels in the pyramid, thus resulting in images of the same resolution [25]. Given its use of integral images, which facilitate the computation of rectangular box filters in near constant time, SURF filters the stack using a box filter approximation of second-order Gaussian partial derivatives.

Given a point $x=(x, y)$ in an image I, the Hessian matrix $H(x, \delta)$ in x at scale δ is defined as follows:

$$H(x,\delta)=\begin{bmatrix} L_{xx}(x,\delta) & L_{xy}(x,\delta) \\ L_{xy}(x,\delta) & L_{yy}(x,\delta) \end{bmatrix} \tag{3.1}$$

where $L_{xx}(x, \delta)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2}g(\delta)$ with the image I in point x, similar to $L_{xy}(x, \delta)$ and $L_{yy}(x, \delta)$. The 9×9 box filters are approximations for the Gaussian second order derivatives with $\delta=1.2$ and represent our lowest scale.

L_{xx}, L_{xy}, L_{yy} can then be denoted as approximations by D_{xx}, D_{xy}, D_{yy}, respectively, to derive the approximated Hessian's determinant:

$$\Delta\left(H_{\text{approx}}\right)=D_{xx}D_{yy}-\left(0.9D_{xy}\right)^2 \tag{3.2}$$

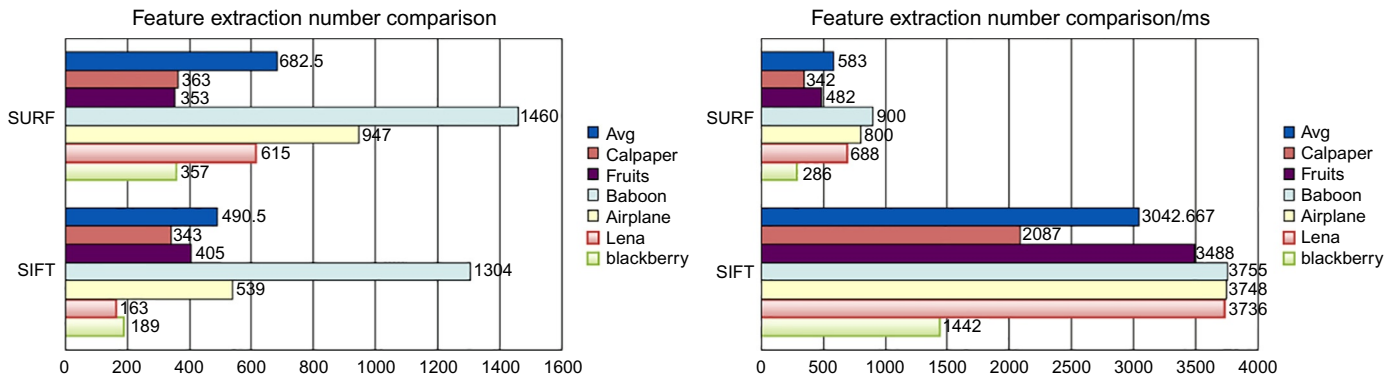


Fig. 3.5 Comparative diagram of the SIFT and SURF.

The determinant in this study is referred to the blob response at location $x = (x, y, \delta)$. The search for the local maximum of this function over both space and scale yields the interest points for an image.

The scale space is analyzed by upscaling the filter size rather than iteratively reducing the image size in SURF. The output of the aforementioned 9×9 filter is considered the initial scale layer. The following layers are obtained by filtering the image with gradually bigger masks, considering the discrete nature of integral images and the specific structure of our filters.

The responses are then thresholded, such that all values below the predetermined threshold are removed. After thresholding, a nonmaximal suppression is performed to identify a set of candidate points. The final step in localizing the points involves the interpolation of the nearby data to identify the location in both space and scale to subpixel accuracy.

3.2.2.3 Improved SURF

The original SURF extracts a significant number of features that are not usually well-distributed in one image. This characteristic affects the computational speed and results in the complicated tracking of the consecutive frame. Therefore, taking steps to overcome these disadvantages is necessary. We proposed several modifications in this study.

(1) Re-selecting the nonmaximum suppression neighborhood

We applied nonmaximum suppression in a $5 \times 5 \times 5$ neighborhood to localize interest points in the image and over scales. To accomplish this task, each pixel in the scale-space is compared with its 74 neighbors, which comprises the 24 points in the native scale and the 25 points in each of the scales above and below. Fig. 3.6 illustrates the nonmaximal suppression step. At this stage, we can identify a set of interest points that are less than those of the original step but are more robust and stable.

(2) Avoiding response to the cluster

Based on Ref. [23], we improve an adaptive discrete feature filter corresponding to the detecting block, which ensures that the extracted key points are well-distributed in the image.

(a) Considering time consumption and tracking accuracy, we preset the feature number threshold n as 20 in our research. Once the number of extracted features exceeds n, the filter is initiated.

(b) Determination of the main key point.

Given that the approximated Hessian's determinant is referred to as the blob response, we can set the corner with the maximum

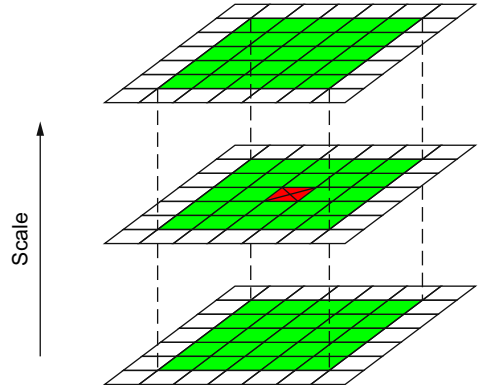


Fig. 3.6 Comparative diagram of the extreme points.

blob response as the first main key point. Similarly, we can determine the second main key point.

(c) Determination of the distance threshold (*dis*).

Assuming that the detection block area is $H \times W$, we set the initial threshold $dis = 5\% * L$, where $L = \sqrt{H^2 + W^2}$ is the diagonal length of the rectangular block. The candidate corners that are distributed in the circle, the center of which is the first main key point with radius *dis*, are then removed. Thereafter, the filter is again activated if the number of features exceeds n, and distance threshold *dis* increases by $1\% * L$. The process continues until the number of features is less than n.

3.2.3 Feature Matching Algorithm

After extracting several appropriate robust features, their exact location and corner response are acquired. If we want to match the features between adjacent frames using the Euclidean distance, the descriptors must first be obtained. Evidently, this is not suitable for our real-time requirement.

As the camera moves, the patterns of image intensities change in a complex manner. These changes can often be described as feature motion in an image sequence. We assume that the motion during image movement agrees with the affine model. This procedure enables the features to be tracked by the KLT algorithm.

Compared with this matching method, KLT is an algorithm that is based on optimal estimation. This algorithm searches the image patch in the

neighborhood region of each feature point to match the reference image patch. The displacement $d = \left[d_x, d_y\right]^T$ between the images is estimated by minimizing the sum of square difference (SSD). The KLT algorithm tracks feature points when the displacement and brightness variation between frames is minimal. Given that the computational cost is much smaller, we adopt KLT in our algorithm for real-time tracking.

3.2.3.1 Improved P-KLT Algorithm

The classical KLT algorithm can be used to track points based on the assumption that the displacement and brightness variation between frames is minimal. Specifically, the motion between consecutive frames must be small and coherent.

However, for most normal video cameras running from 25 to 30 Hz, large and noncoherent motions are common. In practice, we prefer a large window to catch large motions. However, a large window often breaks the coherent motion assumption, which is a natural trade-off between local accuracy and robustness. Thus the classical KLT algorithm becomes unsuitable.

A solution for this problem is a pyramidal implementation of the classical KLT algorithm [26], as shown in Fig. 3.7. We can initially track over larger spatial scales using an image pyramid and then refine the initial motion velocity assumptions by working our way down the levels of the image pyramid until we arrive at the raw image pixels.

The recommended coarse-to-fine feature tracking is first used to solve for optical flow at the top layer. The resulting motion estimations are then employed as the starting point for the next layer down. In this manner, we

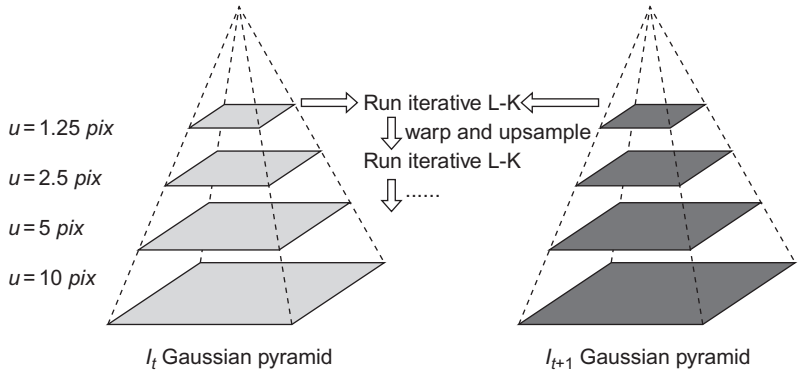


Fig. 3.7 Multiscale flow estimation.

continue going down the pyramid until we reach the lowest level. Thus we minimize the violations of our motion assumptions to track faster and longer motions. This elaborate function is known as the P-KLT.

3.2.3.2 Rejecting the Outliers

Illumination changes and high moving speed result in mismatching. After P-KLT matching, we consider all the coarse matching points' Δx (difference of x coordinate) as statistic and then determine whether the normal distribution is followed. According to the principle of normal distribution, the distributed probability of Δx is as follows:

$$\begin{cases} P(\mu-\sigma<\Delta x<\mu+\sigma)=68.26\% \\ P(\mu-2\sigma<\Delta x<\mu+2\sigma)=95.44\% \\ P(\mu-3\sigma<\Delta x<\mu+3\sigma)=99.74\% \end{cases} \tag{3.3}$$

If $\mu+3\sigma$ is selected as the rejecting threshold, almost all the coarse matching points are reserved and the purpose of rejecting the outliers fails. Similarly, if $\mu+\sigma$ is selected as the threshold, almost 1/3 of the coarse matching points are dropped. This result would hinder the fine matching points from following the uniform distribution, thus affecting tracking accuracy. Therefore $\mu+2\sigma$ is selected as the rejecting threshold. Most of the inconsistent matches are rejected and 95.44% of the coarse matching points are reserved as fine matches.

3.2.4 Precise Location and Adaptive Strategy

3.2.4.1 Precise Location of Object

When the location of well-matched features are determined, such points can be used to locate and determine the exact shape of the object to realize stable and robust tracking. As mentioned in Ref. [24], the Greedy Snake algorithm satisfies our requirement. Thus we attempt to combine the algorithm proposed by Williams [27] with our tracking system.

In MRRFT, the finely matched features extracted from the surface of the object are obtained after rejecting the outliers. Calculating all the points' least circumscribed rectangle, which can be regarded as the simple contour of the object, becomes easy. Thus we set this rectangle's four peaks as the initial contour used in the Greedy Snake algorithm. The main parameters in the Greedy Snake algorithm are preset as follows: $\alpha=0.4$, $\beta=0.03$, and $\gamma=0.36$.

The Gaussian blur and a median filter are applied before defining the initial points because the Greedy Snake algorithm is sensitive to both the Gaussian and Salt and Pepper noise.

Discrete Point Filter

In practice, false points exist during detection and tracking regardless of how well the rejection strategy for outliers works. This result can be attributed to several unknown errors possibly caused by complex illumination and an unintelligent detector or tracker.

When an error occurs, not all points are located in the surface of the object, and several discrete points are observed, the existence of these points will cause several mistakes. Moreover, a similar situation exists in the contour points obtained by the Greedy Snake algorithm's segmentation. Therefore a discrete point filter must be designed.

To deal with these false points, several strategies can be adopted, such as algorithms based on textural characteristic, color characteristics, and distance rules. Considering the requirement of accuracy and real-time results, we used the distance rule in our filter.

We define all the points to filter as $Pt_1, Pt_2, \ldots, Pt_n$, and their 2D coordinates are defined as: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Their geometric center (x_a, y_a) can easily be calculated as:

$$x_a = \frac{x_1 + x_2 + \cdots + x_n}{n}, \quad y_a = \frac{y_1 + y_2 + \cdots + y_n}{n} \tag{3.4}$$

For each $n-1$ point of all the n points, we calculate their average Euclidean distance to the geometric center dis_a.

Then, the Euclidean distance between the rest point (x_r, y_r) and the geometric center dis_r are calculated.

As regards the core, we continue to compare dis_r with dis_a. If the inequality $dis_r > R^* dis_a$ (R is the distance threshold, $R=2$ in this chapter) holds, (x_r, y_r) is replaced by (x_a, y_a). Otherwise, the next judgment works until $n-1$ times are completed.

Adaptive Features Updating Strategy

If features are tracked over longer image sequences, their appearance can undergo larger changes because of pose, scale, and illumination changes. Complex or similar backgrounds will lead to the same problem. Thus we have to determine whether to continue matching against the originally detected patch or to re-sample each subsequent frame at the matching

location. From Ref. [28], the former strategy is known to be prone to failure because the original patch can undergo appearance changes such as foreshortening. The latter runs the risk of the feature drifting from its original location to several other image locations. Thus we design an adaptive features updating strategy that combines active and passive updating.

(1) Active updating

For each N frame, we re-sample the image at the interesting area using the improved SURF to initialize the features to be tracked ($N=10$ in this chapter). The number N can be changed according to the frequency, size of the frames, and the object's relative speed.

(2) Passive updating

Normally, the noncooperative object can be stably and robustly tracked by the improved SURF combined with P-KLT. However, the complex situation when tracking a noncooperative satellite in space makes the use of the active updating strategy insufficient. Thus we also design a passive updating strategy to overcome abnormal tracking. We set up three criteria. Once each of these criteria holds, the passive strategy works to update the features.

(a) Significant changes in the object area

When optical zoom occurs or the distance between the TSR and the object rapidly changes, the area of the object in a 2D image evidently changes. These characteristics often result in tracking failure because of scaling change. Therefore when the object areas, which are calculated by the features in two consecutive frames, change by more than 15%, feature re-sampling is initiated.

(b) Significant changes in the location of the object's centroid

If the location of the object's centroid from all the features in every frame can be found, the Euclidean distance of the centroid to the image's upper left corner, the coordinate of which is (0, 0), can be calculated. Then, we compare the Euclidean distances between two consecutive frames. If the change reaches 10%, feature re-sampling is initiated.

(c) Loss of part of the tracked points

The number of features is counted in every frame. If the change between two consecutive frames reaches 30%, feature re-sampling is initiated.

The proposed algorithm records the tracking box, including size and center, for each frame. When the feature updating strategy is

effective, feature re-sampling is initiated in a new box. Considering accuracy and robustness, the center of the new box is set to be the same as that in the previous normal frame. Meanwhile, the sides of the new box are $1.5\times$ longer than those of the previous box. Thus the area is $2.25\times$ bigger than the old one, which enables the improved SURF to extract more robust features under complicated conditions.

The task flow of our proposed algorithm for the TSR is shown in Fig. 3.8.

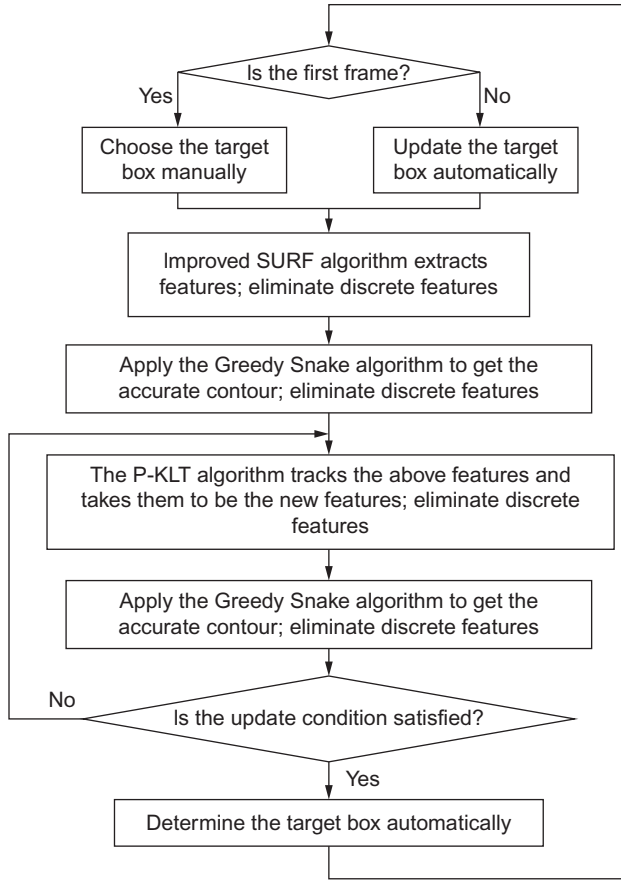


Fig. 3.8 Flow diagram of the target tracking system.

3.2.5 Results, Limitations and Future Works

3.2.5.1 Experiments Condition

Our MRRFT was implemented in VS2005 IDE using the OpenCV 2.0 libraries (Intel 2012) to achieve real-time tracking. Experimental results were obtained using a PC with 2.0GHz Xeon 5130 and 2G RAM.

We used different methods to evaluate MRRFT performance. First, synthetic data were used. Computer image and graph technology was applied to generate the emulated images of the cameras. The 3D models, orbital environment, TSR models, and the noncooperative target were created using Vega Prime.

The main body of the noncooperative satellite is assumed to be a $200 \times 200 \times 250\ mm^3$ cube. In addition, the span of the solar panels is 560 mm. The parameters of the camera used in this study are set with a focal length of 12 mm with 40° FOV angles and an image size of 704×576 pixels. In this simulation, we designed several challenges from the tracking point of view. The challenges were as follows: camera movement and optical zoom; high speed and agility of the satellite; satellite pose variation in pitch, yaw, and roll; occlusions attributed to our preset's strong light spot; specular reflections from the satellite surface and solar array; and background clutter caused by bright stars (Fig. 3.9).

Furthermore, we designed several semiphysical experiments to simulate the real scene in our lab, as shown in Fig. 3.10. A satellite model is used, the main body of which is a $70 \times 70 \times 60\ mm^3$ cube, and the span of the solar

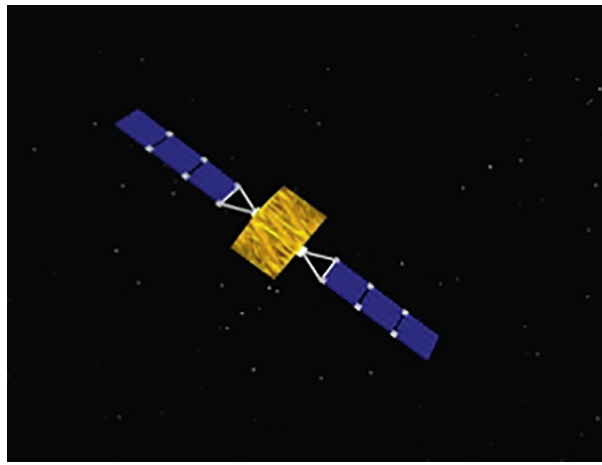


Fig. 3.9 Satellite model generated in Vega Prime.

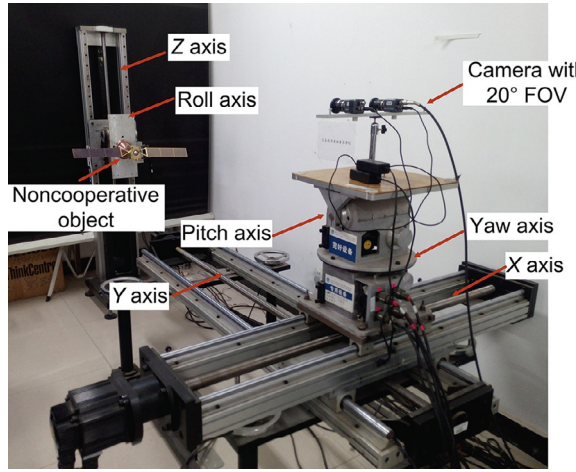


Fig. 3.10 6 DOF motion platform and model.

panels is 30 mm. We have a 6 DOF motion platform comprising a 2 DOF slipway, 2 DOF (pitch and yaw) revolving table, and 2 DOF lifting and roll platforms.

In the experiments, one camera (similar to that described previously) is used to record videos. The 6 DOF and lights can be separately or jointly controlled. We designed different experiments to test our MRRFT in different kinds of scenes. The detailed results are analyzed in the following section.

3.2.5.2 Results

In this section, the performance of the proposed algorithm is evaluated for several types of image sequences. The proposed algorithm is generic and can be used for tracking any object of interest. Moreover, the algorithm can be used to track multiple objects by defining multiple clusters of features, one for each object.

In the experiments, we present results for tracking a single satellite. Given that object detection is outside the scope of this study, the satellite was manually identified to initialize the tracker. Once initialized, the tracking algorithm automatically tracked the satellite in the subsequent frames. All the frames shown in the figures in this chapter were cropped to a fixed size of 320×240 pixels (from 704×576 pixels) to make the satellites more visible. The scale of the aircraft in the cropped frames was relatively small, which increases the difficulty of the tracking scenario.

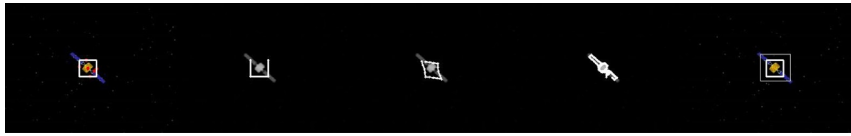


Fig. 3.11 Determination of the target precise location. (A) Features extracted from SURF; (B) initial contours; (C) intermediate step; (D) final result; (E) a comparison of the initial and final contours.

Fig. 3.11 shows the process by which the MRRFT precisely locates the target position. The features extracted from SURF are indicated by red points in (a). The white rectangle stands for the points' least circumscribed rectangle. (b) We set this rectangle's four peaks as the initial contours used in the Greedy Snake algorithm. Then, the Greedy Snake algorithm is employed, and the intermediate step of the Greedy Snake algorithm is shown in (c). The white contour surrounding the satellite in (d) is the final result. (e) shows a comparison of the initial and final contours. The rectangle composed of thinner line segments is the precise location result, whereas the thicker one is the initial contour. Therefore we can conclude that the MRRFT works efficiently.

Fig. 3.12 shows a consecutive tracking sequence when the distance filter works. In frame 255, the tracker incorrectly detects point 1 when the satellite flies across the jamming light spot at the center of screen. Then the distance filter eliminates point 1. The MRRFT works well in the subsequent frames.

Fig. 3.13 shows the process of passive updating in a tracking sequence comprising five consecutive frames. Frame 362 is a normal frame. In frames

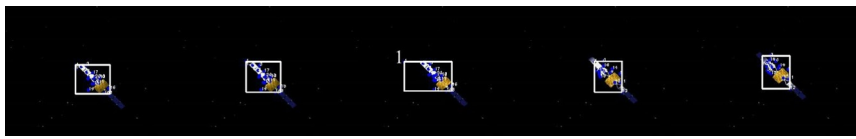


Fig. 3.12 Schematic diagram of the distance filter.

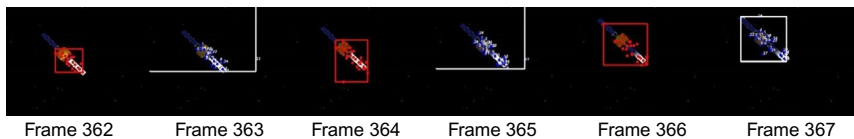


Fig. 3.13 Schematic diagram of the features' passive update.

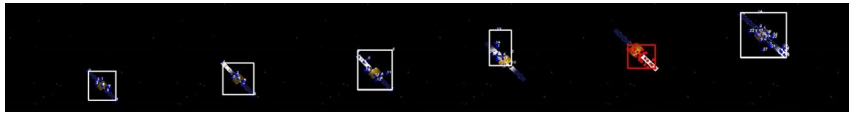


Fig. 3.14 Schematic diagram of tracking in the presence of occlusion.

363 and 365, the high relative speed causes the tracker to fail. Thus the passive updating strategy is employed. In frames 364 and 366, our MRRFT extracts features in a new rectangle, the area of which is 2.25 × bigger than the old one in frames 362 and 364. We consider this rectangle as the new tracking box. The tracker works efficiently in the subsequent frames.

Fig. 3.14 demonstrates the performance of MRRFT when the tracking target is partly occluded by the strong light spot that is preset at the center of screen. We preset a ray light source 20 m away from the satellite, which lights the center point-blank. In frame 341, the target is tracked well in deep space. In frame 348, the top left panel is exposed in the jamming spot. In frame 356, most parts of the satellite are illuminated by the spot. However, our MRRFT could still accurately and automatically tracks the target in the whole sequence despite a few errors caused by strong noise.

Quantitative Comparisons

Fig. 3.15 is the comparative diagram of the tracking satellite models of GFTT + P-KLT (M1), SIFT + P-KLT (M2), and our MRRFT (M3). The white rectangle drawn in the image indicates the precise location acquired by each algorithm. Pose, scaling, illumination, and background changes were present. We ran the algorithms with the same initial target position. From Fig. 3.15A–C, we can find that M1 does not work well in frames 363 and 661, whereas M2 does not work well in frames 331 and 373. However, our MRRFT tracks the noncooperative satellite in the whole sequence more accurately and stably because the improved SURF extracts more robust and well-distributed features when combined with the Greedy Snake algorithm.

We evaluated the aforementioned algorithms using the center location error, as well as the overlapping rate. The results are shown in Tables 3.2 and 3.3. Overall, the proposed tracker performed well against the other state-of-the-art algorithms.

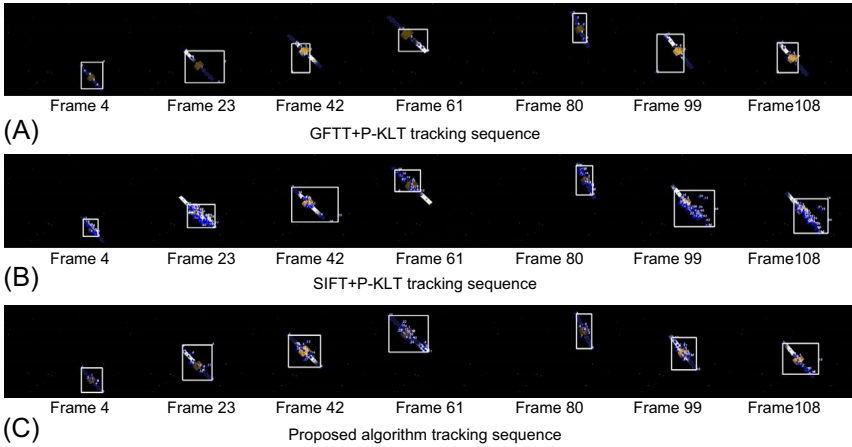


Fig. 3.15 Comparison of tracking satellite models among GFTT + P-KLT, SIFT + P-KLT and proposed algorithm.

Table 3.2 shows the location information of the satellite's geometric center calculated by each algorithm, which is represented by the center of the rectangular box. The standard real coordinates were manually searched by determining the center of their minimum enclosing rectangles. The results show that the average errors of M1, M2, and M3 were 15.0, 15.7, and 5.9 pixels. Our MRRFT located the center more accurately, which can be explained by several factors. The features were well-distributed after filtering, which made the initial contour capable of surrounding most parts of the satellite. Furthermore, the Greedy Snake algorithm extracted an accurate shape because of the precise initial contour.

Table 3.3 shows the area information of the satellite's minimum enclosing rectangle calculated by each algorithm, which is represented by the center of the rectangular box. The standard real coordinates were manually searched by finding the center of their minimum enclosing rectangle. We tested these algorithms by using the same video sequence. A comparison of the seven frames is made in this study. The results show that the contour area error per frame of M1, M2, and M3 is 33.8%, 39.9%, and 12.6%, respectively. Equally, our MRRFT has higher definition because of the same reasons.

Table 3.2 Location information of the satellite's geometric center

Frame number	Real location	M1 tracking	M2 tracking	M3 tracking	M1 error	M2 error	M3 error
8	(229.5,192.5)	(231.5,186.5)	(225,187.5)	(229.5,191.5)	6.3	6.7	1.0
18	(191.5,161.5)	(208,164.5)	(194,157.5)	(186,147)	16.8	4.7	15.5
28	(151.5,123.5)	(142,142)	(170.5,128)	(146,118.5)	20.8	19.5	7.1
38	(103.5,79)	(118,94.5)	(92.5,67.5)	(98.5,73.5)	21.2	15.9	7.4
48	(238,67)	(237,63)	(233.5,66)	(238,67.5)	4.1	4.6	0.5
58	(177.5,122.5)	(156,128.5)	(200.5,138)	(179.5,125)	22.3	27.7	4.0
68	(159,139.5)	(145.5,140)	(184,157.5)	(165,139)	13.5	30.8	6.0

Table 3.3 Location information of the satellite's minimum enclosing rectangle

Frame number	Real area	M1 tracking	M2 tracking	M3 tracking	M1 error	M2 error	M3 error
8	4221	3933	1710	3355	6.8%	59.5%	20.5%
18	6241	8466	4248	6688	35.7%	31.9%	7.2%
28	7265	3648	10,890	6804	49.8%	49.9%	6.3%
38	9870	4322	3685	9579	56.2%	62.7%	2.9%
48	4700	2664	3182	3480	43.3%	32.3%	25.9%
58	7007	6930	9870	5330	1.1%	40.9%	23.9%
68	7476	4180	7656	7332	44.1%	2.4%	1.9%

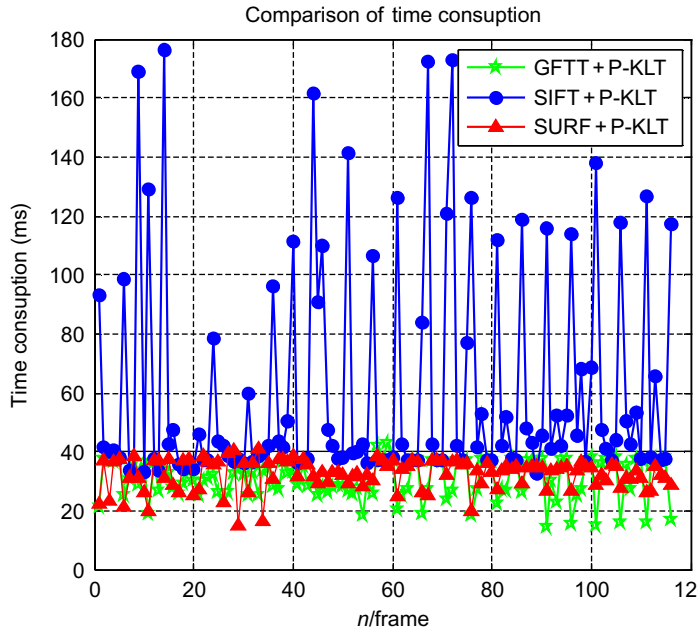


Fig. 3.16 Comparison of time consumption.

In the experiments, we recorded the time consumptions of the whole process in each frame. All the methods used P-KLT matching. The average time consumption of M1, M2, and M3 can be calculated as 30.7, 60.7, and 32.5 ms, respectively. The statistics of 116 consecutive frames is shown in Fig. 3.16. From the figure, the average time consumption of all the frames is approximately 34 ms. The blue points, which are higher than average, refer to the time consumption of the SIFT feature extraction. The green pentagrams and the red triangles, which are lower than average, represent the time consumption of GFTT and SURF feature extraction. Thus, our MRRFT can be used in real-time tracking situations.

Qualitative Analysis

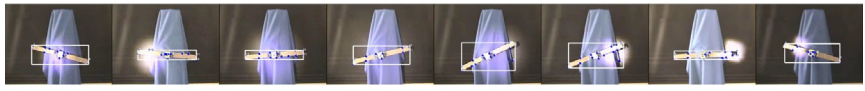
The proposed algorithm was tested in semiphysical experiments. Fig. 3.17 shows six tracking sequences obtained in different scenes:

Fig. 3.17A Tracking sequence with glimmer: A dusky spot lamp-house was used. The edge of the model was too dusky for feature extraction. The light was slightly brighter at the center. Thus, the MRRFT worked well.

Fig. 3.17B Tracking results with dramatic illumination and pose changes: The bright spot in the figure was a strong noise caused by illumination from an electric torch. We can directly conclude that our MRRFT

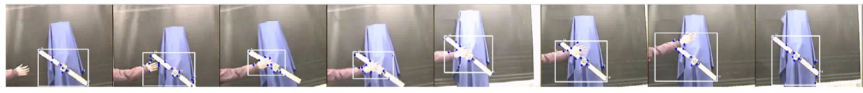
Frame 401 | Frame 402 | Frame 403 | Frame 404 | Frame 405 | Frame 406 | Frame 407

(A) Tracking results with glimmer



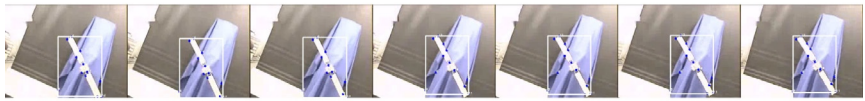
Frame 2551 | Frame 2591 | Frame 2631 | Frame 2671 | Frame 2711 | Frame 2751 | Frame 2791 | Frame 2831

(B) Tracking results with dramatic illumination change and pose changes



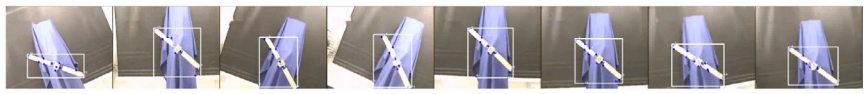
Frame 778 | Frame 783 | Frame 808 | Frame 825 | Frame 879 | Frame 920 | Frame 951 | Frame 999

(C) Tracking results with severe occlusions



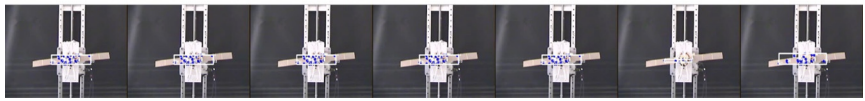
Frame 297 | Frame 298 | Frame 299 | Frame 300 | Frame 301 | Frame 302 | Frame 303

(D) Tracking results with motion blur



Frame 231 | Frame 253 | Frame 275 | Frame 297 | Frame 319 | Frame 473 | Frame 561 | Frame 643

(E) Tracking results with in-plane and out-of-plane rotations



Frame 31 | Frame 71 | Frame 111 | Frame 151 | Frame 191 | Frame 231 | Frame 271 | Frame 311

(F) Tracking results with complex backgrounds

Fig. 3.17 Tracking results in six typical challenging video sequences.

worked well even with pose changes. Furthermore, strong noise was observed.

Fig. 3.17C Tracking results with severe occlusions: We can see that several disturbed features caused by the occlusion do not cause MRRFT to fail. MRRFT can also stably tracked the robust features. Once the occlusion disappeared, MRRFT again worked efficiently.

Fig. 3.17D Tracking results with motion blur: Fast motion result in motion blur in consecutive frames, which is difficult to account for in object tracking. Although the true target was blurred, the improved SURF could extract sufficient robust features despite small errors.

Fig. 3.17E Tracking results with in-plane and out-of-plane rotations: The MRRFT performed well throughout the sequence with translational and similarity transformations.

Fig. 3.17F Tracking results with complex backgrounds: Despite using the Greedy Snake algorithm to determine the exact contour, the MRRFT did not perform well in this sequence. This characteristic was challenging as the background was cluttered because the holistic representations inevitably include background pixels that could be considered as part of the foreground object through straightforward update schemes.

In our experiments, the features extracted outside the object surface and points not covering the entire object do not frequently appear because of the adoption of the Greedy Snake algorithm. A rich set of experiments, including comparative analysis, confirmed our approach. However, when the object scale was extremely small, the improved SURF could not extract a sufficient number of features. Then, the satellite model could not accurately describe the features and the tracker fails. Another source of error was the high relative speed between the satellite and the operational robot, especially when the camera moved opposite to the direction of the satellite.

These issues can be addressed by introducing a dynamic state model into our MRRFT, which can better predict the location of the object in the subsequent frame. This method can help avoid the shifting of the tracker to a background object by preventing sudden changes in the state model.

Any single tracking algorithm cannot deal with all challenges simultaneously. In the future, we plan to use a combination of tracking algorithms, such as particle filter and camshift, which complement each other. We expect that this method can possible achieve more robustness to these challenging factors.

3.3 DETECTION OF ROI

In this section, we discuss the detection of circular ROI. It can be used for recognition of the common parts mounted in debris.

A Guiding Operational Robot that is used to capture the circular bolts, docking ring or motor injector, is expected to lock the combination system for refine manipulation. Only with prior knowledge can we use it to recognize that the target has circular modules since it is noncooperative. To circumvent the aforementioned limitations of the existing methods, we propose a fast and robust circular object detection method.

3.3.1 Arc Support Region

Desolneux et al. [28] presented an algorithm for straight segment detection. The two key points of their approach are the use of gradient orientation and a new framework to deal with parameter settings. And in Ref. [28b], they introduced neuroperception science to determine the gradient threshold, which improved the detection process of aligned line segments based on image gradient.

Based on their works, and the description in Refs. [29,30], for a given image $I(x, y)$, the gradient of pixel (x, y) can be calculated by the following equation:

$$g_x(x, y) = \frac{I(x+1, y) + I(x+1, y+1) - I(x, y) - I(x, y+1)}{2}$$
$$g_y(x, y) = \frac{I(x, y+1) + I(x+1, y+1) - I(x, y) - I(x+1, y)}{2} \quad (3.5)$$

The gradient direction of (x, y) is:

$$dir(x, y) = arctan\left(-\frac{g_x(x, y)}{g_y(x, y)}\right) \quad (3.6)$$

The gradient magnitude value of (x, y) is:

$$G(x, y) = \sqrt{g_x(x, y)^2 + g_y(x, y)^2} \quad (3.7)$$

A line segment in Fig. 3.18, is defined as the line-support region, namely a straight region whose points share roughly the same image gradient angle. Such line segments are roughly oriented along the average level-line direction equal to the line segment angle up to a certain tolerance τ.

$$\tau = |dir(x_1, y_1) - dir(x_2, y_2)| < 2\pi/n \quad (3.8)$$

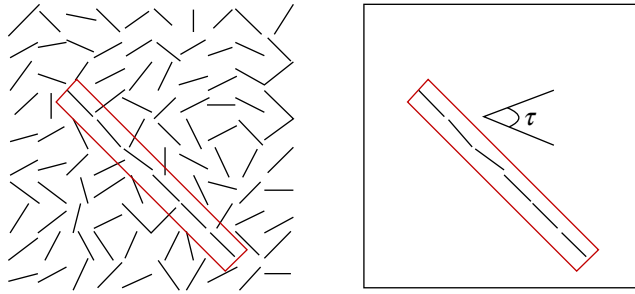


Fig. 3.18 Schematic diagram of gradient direction.

τ is the angle tolerance used in the search for line support regions. According to Refs. [28b,31], a small value is more restrictive, leading to an over-partition of line segments. A large value results in large regions and in the merging of unrelated ones. In this section, we concentrate on detecting ASRs in image. According to numerical experiments, when $\tau = 60°$, it works well in the search for different ASRs.

ρ is a threshold on the gradient magnitude: Pixels with small gradient are not considered. The reason is to cope with the quantization of the image intensity values.

The criterion we use to set ρ is to leave out points where the angle error is larger than the angle tolerance. Let I denote the image and I' the quantized one. Then:

$$\begin{aligned} I' &= I + c \\ \nabla I' &= \nabla I + \nabla c \end{aligned} \tag{3.9}$$

where c is the quantization noise. In Ref. [6], it is shown that the error of the image gradient angle can be bound by

$$|\nabla angle| = \arcsin\left(\frac{|\nabla n|}{|\nabla I'|}\right) \leq \arcsin\left(\frac{q}{|\nabla I'|}\right) \tag{3.10}$$

where q is a bound to $|\triangle c|$. Imposing that $|\triangle angle| < \tau$ one obtains:

$$\rho = \nabla I' = \frac{q}{\sin \tau} \tag{3.11}$$

For example, on images quantized to grey values in 0, 1, 2, …, 255, $|\triangle c|$ can be bounded by $q = 2$ (the worst case is when adjacent points gets errors of +1 and −1). We set τ to 60° and the threshold to the gradient magnitude is $\rho = 69.3$.

In order to partition the image into ASRs by grouping connected pixels that share the same gradient angle up to τ, a region growing algorithm is applied. Each region starts with just one pixel and the region angle set to the level-line angle at that pixel (orthogonal to the gradient angle). Then, the pixels adjacent to the region are tested; the ones with level-line orientation equal to the region angle up to a certain precision are added to the region. At each iteration, the region angle is updated to a pseudomean level-line orientation of the region's pixels. The process is repeated until no new points can be added.

Seed pixels with larger gradient magnitudes are tested first as they are more likely to belong to straight edges. A pseudo-ordering is used here to reduce computational complexity. When a pixel is added to a region, it is marked and never visited again. This key property makes the algorithm greedy and therefore linear.

After region growing, several ideal ASRs are acquired while some short line or curve segments are extracted in mistake. Their existence will affect the accuracy of circle parameters. So the number of points contained in each region is counted and compared with the threshold P. P is a threshold on the number of aligned points. The regions, whose number of aligned points is smaller than P, are deleted.

3.3.2 Estimation of Circle Parameters

For an ASR, we can determine its corresponding circle equation and solve the circular parameters (a, b, r) by using three noncollinear coordinates on it, such as two endpoints (x_1, y_1),(x_2, y_2) and midpoint (x_3, y_3).

Then the three points are used as a group to compute the circle equation:

$$\begin{aligned}(x_1 - a)^2 + (y_1 - b)^2 &= r^2 \\ (x_2 - a)^2 + (y_2 - b)^2 &= r^2 \\ (x_3 - a)^2 + (y_3 - b)^2 &= r^2\end{aligned} \tag{3.12}$$

The value of (a, b, r) can be calculated when the actual values are added to these equations.

It is worth mentioning that, each ASR obtained above has many inner points without a parallel thinning process. Hence, the value (a, b, r) cannot be used as the ideal parameter. However, the coarse value can be used for fine estimation.

An SFA can be determined by (a, b, r), of which the center is (a, b) and the side length is $r/2$. In general, the ideal center of a circle is located in an SFA.

As a salutary reference, Ref. [32] described a method for circle detection using a histogram based accumulator space. When using the method combined with the defined SFA, we can acquire accurate circular parameters with less computational complexity and time consumption.

In detail, the SFA is defined as $Q(r/2, r/2)$, and then the Euclidean distances $Dis(i, j)$ between every coordinates on each ASR and each coordinate of its ASR's corresponding SFA are computed and recorded in a three dimensional accumulator.

$$Dis(i,j) = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{3.13}$$

where $Dis(i, j)$ is a vector with the distances between any pixel (x_i, y_i) on each ASR and coordinate (x_j, y_j) of its ASR's corresponding SFA.

Then, a histogram is used to count the frequency of the distances that participates in the accumulator. For each ASR found in the image, the same process is followed. Assuming that there is a real circle in the image, it is detected and described as several ASRs. The distances between each point on the ASR and the true center (x_c, y_c) can be counted. The highest frequency of distances is the radius r_c, definitely. So the parameters of each circle are acquired after a frequency count.

3.4 VISUAL SERVOING AND POSE MEASUREMENT

3.4.1 Theory of Calculating Azimuth Angles

In order to analyze more easily, we draw the pinhole camera model in Fig. 3.19. $OwXwYwZw$ is the world coordinate system (WCS), $OcXcYcZc$ is the camera coordinate system (CCS), $o'uv$ is the pixel coordinate system (PCS), and oxy is the retinal coordinate system (RCS).

Where o' is the origin of the PCS and o is the origin of the RCS. o, which at the intersection of the image plane and the optical axis, is referred to as the principal point.

The principle point is usually not equivalent to the center of the imager. In fact, the center of the chip is usually not on the optical axis. So two new parameters, u_0 and v_0, are introduced to model a possible displacement (away from the optical axis) of the center of coordinates on the projection screen.

Define the physical size of each pixel in axis x and y as d_x, d_y. Then a relatively simple model in which a point Q in RCS, whose coordinates are (x, y), is projected onto the screen at some pixel location given by (u, v) in accordance with the following equation:

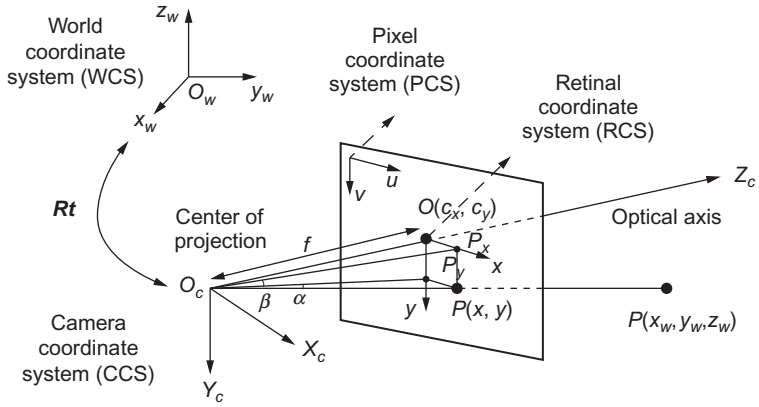


Fig. 3.19 Schematic diagram of image sensor's imaging model.

$$\begin{cases} u = \dfrac{x}{d_x} + u_0 \\ v = \dfrac{y}{d_y} + v_0 \end{cases} \tag{3.14}$$

Given a point $P(x_w, y_w, z_w)$ of the noncooperative satellite in the WCS, it can be projected onto the screen at some pixel location given by $P(x, y)$ in the RCS. Define the line of sight O_cP, and O_dpx is the projection vector in $O_cX_cZ_c$ plane of the line of sight O_cP. *Define* α *is the* angle of position, which is the angle between O_dpx and O_cZ_c. O_dpy is the projection vector in $O_cY_cZ_c$ plane of the line of sight O_cP. *Define* β is the angle of site, which is the angle between O_dpy and O_cP.

$$\begin{cases} \alpha = \arctan \dfrac{(u - u_0)d_x}{f} \\ \beta = \arctan \dfrac{(v - v_0)d_y}{f} \end{cases} \tag{3.15}$$

Given the relation that maps the point $P(x_w, y_w, z_w)$ in the physical world to the point on the projection screen with coordinates $P(u, v)$, we can calculate α, β by Eq. (3.15) according to the research described in Ref. [33]. Where d_x,d_y can be calculated according to the size of CCD and picture resolution. They are also can be acquired from the calibration results.

3.4.2 Improved Template Matching

In contrast with the other common correlation based similarity methods namely Sum of Squared Difference (SSD), Normalized Cross Correlation (NCC) and Sum of Hamming Distances (SHD), SAD is simple, more accurate and less complex. While, sometimes it is not accurate if two image windows have almost the same grey distribution. In order to improve this algorithm we give Normalized SAD (NSAD) in the following equation:

$$d(A, B) = \sum_x \sum_y \frac{|(A(x, y) - A_m) - (B(x, y) - B_m)|}{\max(A(x, y), B(x, y))} \tag{3.16}$$

where, $A(x, y)$, $B(x, y)$ are the value in the position (x, y) in template image and the subsearched image. A_m, B_m are the mean value of template image and the subsearched image.

The best matching position is the position where the value $d(A, B)$ is minimum. According to the matching criterion function, NSAD has the translation invariance, but the rotation invariance, which restricts its application in some complex circumstance, cannot be guaranteed.

Given FAST algorithm's success with hollow annulus [34], we create a series of concentric rings as shown in Fig. 3.20. Since circularity possess the quality of central symmetry, the proposed ring-shape matching criterion can make the method have the rotation invariance.

It deserves to be mentioned that solid circularity shape matching criterion has the same effect. However, it is not only apt to bring about an accumulated error when there is inconstant environment, but also to lose some detailed information of the image because of the sum of the difference

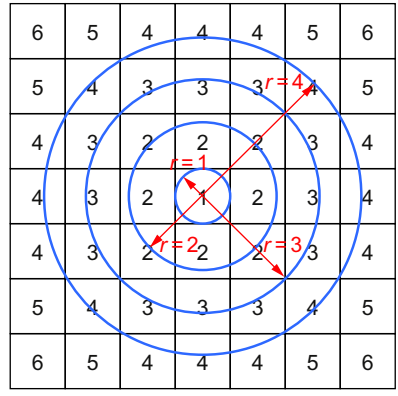


Fig. 3.20 Schematic diagram of distributed pixels assembled into hollow annulus.

values. In this way, some mistake matching results will be obtained, seen in Ref. [35].

As a result, a 35 pixel × 35 pixel square template image can be regard as composed of 18 concentric complete and incomplete rings with $r = 1, 2, \ldots, 18$. In the same way, we can deal with N pixel × N pixel square template image.

Therefore e_{oi} is the sum of the grey difference values of the ith sub concentric ring:

$$e_{oi} = \sum_{j=1}^{N_i} \frac{\left|\left(A_j(x, y) - A_{im}\right) - \left(B_j(x, y) - B_{im}\right)\right|}{\max\left(A_j(x, y), B_j(x, y)\right)} \tag{3.17}$$

And the matching function can be defined as follows:

$$M_o = \sum_{i=1}^{18} e_{oi} \tag{3.18}$$

where $A_j(x, y)$ and $B_j(x, y)$ are the grey value of the jth coordinate in the ith ring of the template image and the subsearched image. A_{im} and B_{im} represent the mean grey value of the ith ring.

N_i is the total number of all the pixels contained in the ith ring, while M_o is defined as the grey value matching function of 35 pixel × 35 pixel square image.

It can be analyzed that results also lose some detailed information of the image because of the sum of the difference values. As is known, the image gradient informs the change of grey values between adjacent pixels; therefore, it is meaningful to detect the edge of structure in the image.

Therefore we introduce the gradient information to add some detailed information of the image in matching its function since it is meaningful to increase pixel uniqueness.

For pixel (x, y) in image I, the gradient $g(x, y)$ is defined as:

$$g(x, y) = [I(x+1, y) - I(x-1, y)]*i + [I(x, y+1) - I(x, y-1)]*j \tag{3.19}$$

where, I(x, y) is the grey value of pixel (x, y), i, j is the direction of the Cartesian coordinate. Its module $m_g(x, y)$ can be calculated as:

$$m_g(x, y) = \sqrt{g(x, y, 1)^2 + g(x, y, 2)^2} \tag{3.20}$$

where, $g(x, y, 1)$ and $g(x, y, 2)$ are the gradient value of pixel (x, y) in direction i and j.

Then we can define that:

$$e_{gi} = \sum_{j=1}^{N_i} \frac{\left|\left(m_{jga}(x, y) - m_{igam}\right) - \left(m_{jgb}(x, y) - m_{igbm}\right)\right|}{\max\left(m_{jga}(x, y), \left(m_{jgb}(x, y)\right)\right)} \tag{3.21}$$

$$M_g = \sum_{i=1}^{18} e_{gi} \tag{3.22}$$

where, e_{gi} is the sum of the gradient difference values of the ith sub concentric ring, $m_{jga}(x, y)$ and $m_{jgb}(x, y)$ are the gradient modules of the jth coordinate in the ith ring of the template image and the subsearched image. m_{igam} and m_{igbm} represent the mean gradient module of the ith ring. M_g is defined as the gradient module matching function of 35 pixel × 35 pixel square image.

Finally, we construct the matching criterion as follows:

$$M_h = W_1 \times M_o + W_2 \times M_g \tag{3.23}$$

W_1 and W_2 are the weight coefficient, which can be decided according to different circumstances. We can conclude that the synthesized criterion function M_h has the rotation invariance just like M_o and M_g. Meanwhile, the matching accuracy rate can be raised. So it can be used to recognize targets in complex background.

3.4.3 Least Square Integrated Predictor

When tracking a moving target, it is meaningful to predict the future position using the past information, which can decrease time consumption and improve occlusion invariance. Considering our online application, the least square integrated predictor is raised and applied here.

Assume the central coordinate of the target is $(x(t), y(t))$ and the relative movement between our Operational Robot and the target is a uniform straight-line motion. The best linear estimation of $x(t)$ is,

$$X_l(t) = a_0 + a_1 t \tag{3.24}$$

Which can be rewritten as $X_l = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}$

$X_l(t_i)$ $(i = 1, 2, \ldots, N)$ is the measured value in different moments. Then the difference between the estimation and measured value is:

$$\varepsilon_i = (a_0 + a_1 t_i) - X_l(t_i) \tag{3.25}$$

The N moments residual sum of the squares is calculated:

$$\sum_{i=1}^{N} \varepsilon_i^2 = \sum_{i=1}^{N} [(a_0 + a_1 t_i) - X_l(t_i)]^2 \tag{3.26}$$

We can get the best approximation when Eq. (3.24) gains the least value using the least square method:

$$\begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \dfrac{\sum_{i=1}^{N} t_i^2 \sum_{i=1}^{N} X_l(t_i) - \sum_{i=1}^{N} t_i \sum_{i=1}^{N} X_l(t_i) t_i}{D} \\ \dfrac{-\sum_{i=1}^{N} t_i \sum_{i=1}^{N} X_l(t_i) + N \sum_{i=1}^{N} X_l(t_i) t_i}{D} \end{bmatrix} \tag{3.27}$$

$$D = N \sum_{i=1}^{N} t_i^2 - \left(\sum_{i=1}^{N} t \right)^2 \tag{3.28}$$

Eq. (3.28) is just the general solution of the N moments optimum linear prediction under the least error condition.

Similarly, assume the relative movement is a uniform acceleration curvilinear motion. The best square estimation of $x(t)$ is:

$$X_q = a_0 + a_1 t + a_1 t^2 \tag{3.29}$$

Equally, we can get the general solution a_0, a_1, a_2 using the least square method.

Generally, the relative motion is composed of linear motion and curvilinear motion. Since linear approximation can quickly reflect the target's state, while square approximation has a smoothing filter effect on the target's trajectory, we construct a least square integrated predictor as shown in Eq. (3.30):

$$X(t) = W_4 \times X_l(t + 1/t) + (1 - W_4) \times X_q(t + 1/t) \tag{3.30}$$

The weight coefficient W_4 can be decided by the prediction errors of the linear predictor and square predictor. Such integrated predictors possess the ability of automatic corrective measure, moreover they can fit a nonuniform growing curve. In this section, $W_4 = 0.8$ makes the experiment results fine.

Tests indicate that prediction accuracy declines when N grows after a threshold. So we use a 3 moments linear predictor and a 5 moments square predictor:

$$X(k+1) = (4x(k) + x(k-1) - 2x(k-2))/3 \quad \text{and} \quad X(k+1) = (9x(k) - 4x(k-2) - 3x(k-3) + 3x(k-4))/5.$$

3.4.4 Updating Strategy of Dynamic Template

From 100 to 20 m, the shape and size of a noncooperative satellite change quickly, resulting from the pose variation in a complex and sometimes changing environment. The difference between template image and target image then gradually becomes much larger. Once the difference is accumulated to a threshold value, the original template image becomes invalid. Hence, we have to update the template dynamically in order to maintain recognition and track stability.

The matched point is the geometrical center of the matching sub image in this section. Firstly, we set up two updating criteria.

(a) The coordinate error between target's predicted position and matched position $\mathrm{DIS} = \sqrt{(x-x')^2 + (y-y')^2}$, where (x, y) represents the real coordinate of matched target in current frame, (x', y') means the predicted coordinate of target in current frame.

(b) The similarity degree between the template image and the matched target sub image-$S = 1/M_h$

Thus we design an adaptive template updating strategy, in each matched frame:

(a) If $D > D_t$, it means position tracking is not in normal operation due to occlusion or sudden change. In regard to track stability, we continue to use the template while the predicted coordinate is temporarily adopted as the matched center until the tracking becomes fine.

(b) If $D < D_t$ and $S < S_t$, it means position tracking is fine, but the tracking is going to fail in case the template image is used continually. Updating the template image should be considered immediately following the designed formula:

$$TM_t = W_3 \times TM_{t-1} + (1 - W_3)P_t \tag{3.31}$$

where, TM_t is the template image in the current frame while TM_{t-1} is in the prior frame. And P_t represents the matched sub image in the prior frame. It is worth mentioning that D_t, S_t can be changed according to the object's appearance and environment, etc.

3.4.5 Visual Servoing Controller

Define α is the noncooperative satellite's angle of position in the Operational Robot's FOV, β is the angle of site. We design a visual servoing controller using PD controller with a presetting dead zone, which is similar to the work of [36].

Then we design the control law as follows:

$$\begin{cases} F_{y_r} = -\left(K_{p\alpha} + K_{d\alpha}s\right)\alpha & \text{while } |\alpha| \geq \alpha_0 \\ F_{z_r} = -\left(K_{p\beta} + K_{d\beta}s\right)\beta & \text{while } |\beta| \geq \beta_0 \end{cases} \tag{3.32}$$

where, F_{yr}, F_{zr} are the control forces in direction y and z. $K_{p\alpha}$, $K_{p\beta}$ are the proportion coefficients. $K_{d\alpha}$, $K_{d\beta}$ are the damping coefficients. α_0, β_0 are the dead zone of our designed controller.

As described in Fig. 3.21, α, β actually reflect the position deviation in direction y_r and z_r. As a result, we can control the position deviations by controlling the azimuth angles under a certain threshold range.

The axes of the target's orbit coordinate $OtXtYtZt$ are oriented as follows: the x-axis is in the orbital plane in the local horizontal direction, the y-axis is along the orbital normal and the z-axis is collinear with a line that extends from the center of the Earth to the centroid of the space platform and completes a right-handed triad.

During the approaching process, when the deviations of α, β are restricted under a certain threshold, the position deviations will gradually decrease since the distance between the Operational Robot shortens.

Three factors should be considered when selecting proper α_0, β_0. First, they should satisfy the requirement of position control. Second, it should be

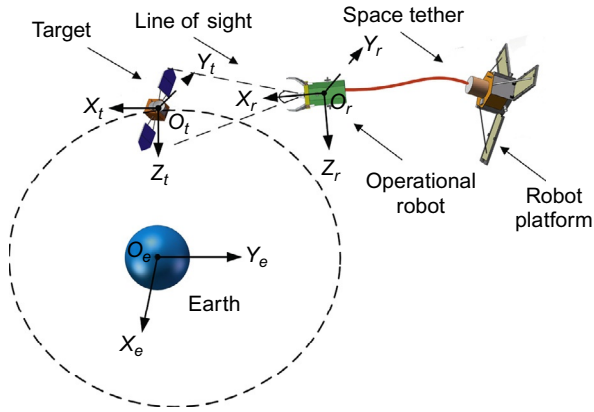


Fig. 3.21 Schematic diagram of TSR's approaching using azimuth angles.

avoided to control frequently in order to save the mass carrier. Furthermore, the monocular camera's FOV is worth considering.

3.4.6 Experimental Validation

3.4.6.1 Experimental Set-up

A servoing control system is set up based on monocular vision to test our algorithm. It is composed of a control subsystem, a visual perception subsystem, an intelligent motion platform, and a target simulator. The visual perception subsystem uses a camera named HaiLong M200A mounted on a 1/3.2 inch (4:3) CMOS sensor and a fixed focal Computer M1614 with 23.4° FOV, 16 mm focal length. The image resolution is set as 1280 pixel × 960 pixel. After transformation, we get the physical length of the rectangle CMOS as 6.35 mm × 4.762 mm. Then the key parameters can be calculated $d_x = 6.35/1280$, $d_y = 4.762/960$.

The tests are implemented in VS2010 IDE using OpenCV 2.4.3 libraries. Experimental results are obtained using a PC with 2.0GHz Xeon 5130 and 2G RAM.

3.4.6.2 Design of Experiments

We designed several semiphysical experiments to simulate the real scene in our lab, as shown in Fig. 3.10. A Chang'e-IIsatellite model is used, the main body of which is a $70 \times 70 \times 60$ mm^3 cube. We have a 6 DOF motion platform composed of a 2 DOF slipway, 2 DOF (pitch and yaw) revolving table, and 2 DOF lifting and roll platforms. The 6 DOF can be controlled separately or jointly by PEWIN32RO installed in PC, including their amplitudes, velocity, and acceleration. Lights systems, background simulation, and any other environmental factors can also be separately or jointly controlled.

We designed a curve approach experiment to test our algorithm. In experiments, we design some challenging scenes as from the tracking point of view, such as camera movement and optical zoom, high speed and agility of satellite, satellite pose variation in pitch, yaw and roll, occlusions due to our preset strong light spot, specular reflections from the satellite surface and solar array, and background clutter caused by bright stars.

3.4.6.3 Results and Discussions

Qualitative Analysis

The template image used in our research is 60 pixel × 60 pixel as shown in Fig. 3.22A.

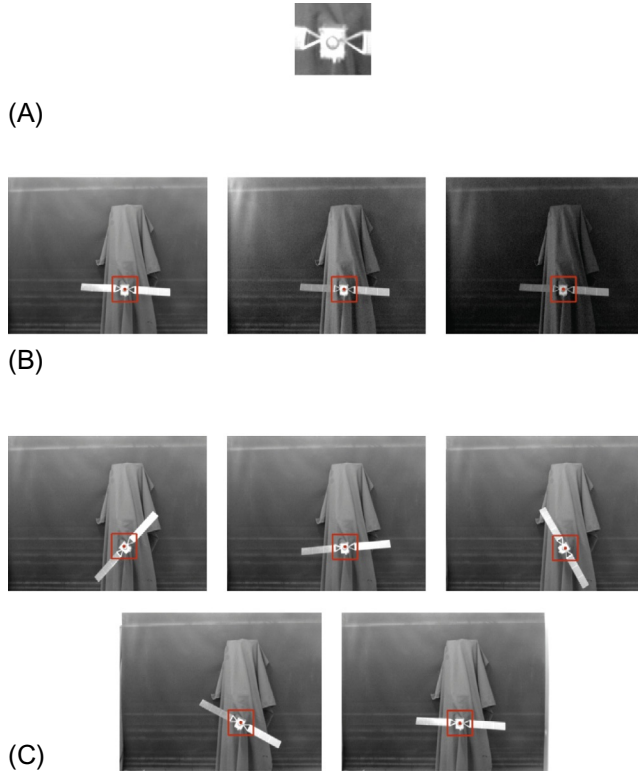


Fig. 3.22 Qualitative tests of the proposed template matching algorithm. (A) A template image; (B) matching results under different light conditions; (C) images matching results with different rotation angles.

Fig. 3.22B shows the matching results under different light conditions and Fig. 3.22C are images matching results with different rotation angles. It can be concluded that our algorithm is robust to brightness and rotation.

Quantitative Comparisons

In order to test our method deeply, it was applied in the TSR's monocular visual servoing scheme and tested in semiphysical experiments. We recorded all the found centroid, which could be used for calculating azimuth angles.

Fig. 3.23 shows the tracking sequences in the first 717 frames. It can be seen that the template image is matched well respectively. Fig. 3.24 shows the measurements of coordinate x and y in the first 717 frames.

We also analyze the performance of the synthesis predictor. From Fig. 3.25, accurate errors data are drawn in detail. The max absolute error of the x coordinate is 9.5 pixel and the mean absolute error is 0.75 pixel.

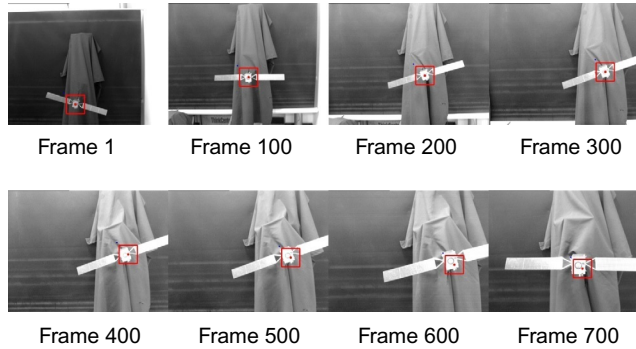


Fig. 3.23 Tracking results of our algorithm in the first 717 frames.

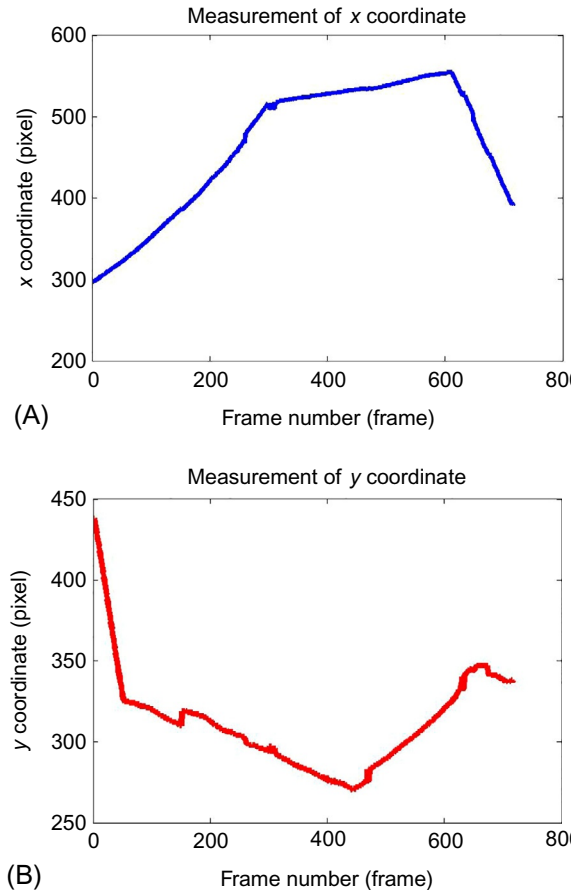


Fig. 3.24 Measurements of coordinate *x* and *y* in the first 717 frames. (A) Measurement of *x* coordinate. (B) Measurement of *y* coordinate.

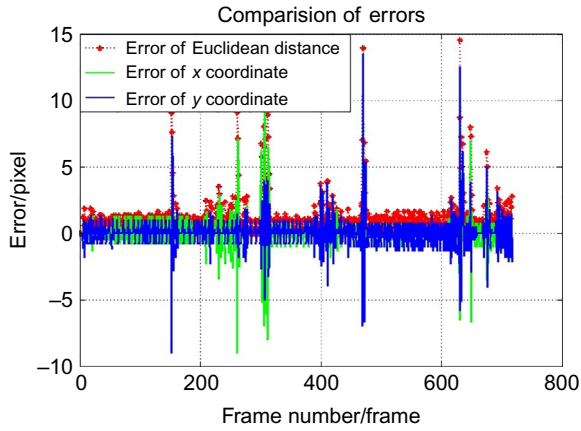


Fig. 3.25 Comparison of prediction error per frame of the least square synthesis predictor.

The max absolute error of the y coordinate is 13.5 pixel and the mean absolute error is 0.84 pixel. The max absolute error of the Euclidean distance is 14.57 pixel and the mean absolute error is 1.28 pixel. It can be concluded that our designed predictor has good performance in accuracy.

Based on the data, we can also draw the real motion path and the predicted one of the Operational Robot in the approaching process. In Fig. 3.26, the red line marked with triangle is real motion path while the blue one is the predicted one. We can intuitively find that the predicted one highly coincides with the real one. Actually, once the predictor gives the possible position, the small template image only has to search for a

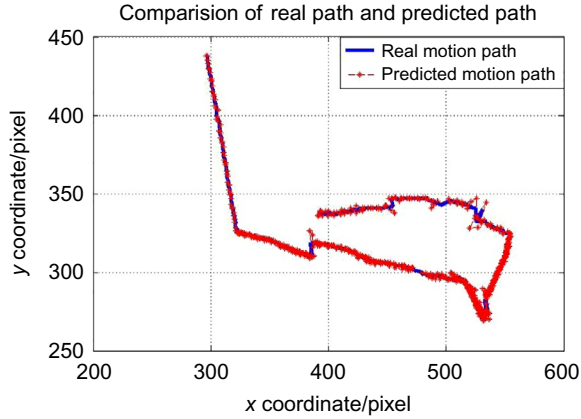


Fig. 3.26 Comparison of path generated by our algorithm and real path.

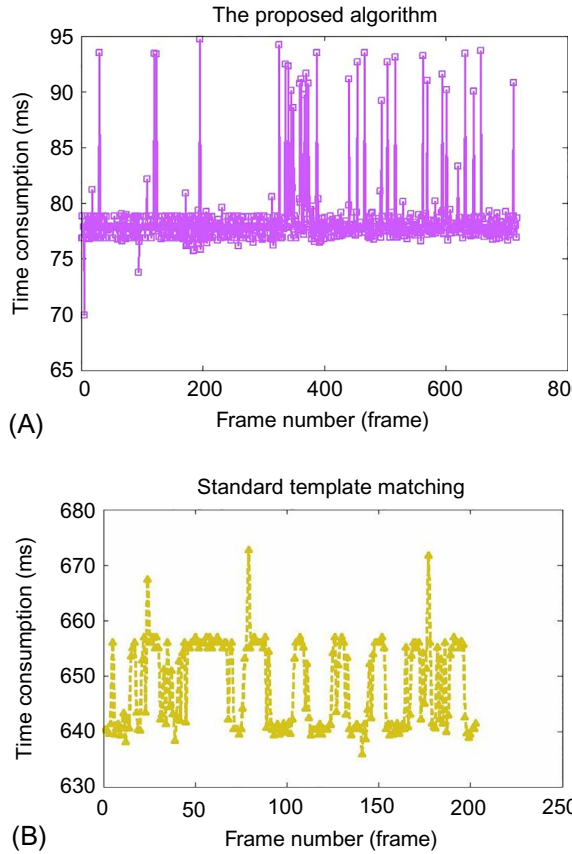


Fig. 3.27 Comparison of time consumption per frame between two algorithms. (A) Time consumption of the proposed method. (B) Time consumption of the standard template matching.

matching scene within a 50 pixel × 50 pixel window centered on this point. As a result, it can decrease sophisticated computation and improve matching accuracy (Fig. 3.27).

As contrast, we also used standard template matching to track the small target and recorded the time consumptions of the whole process in each frame. (A) shows the consumption of our proposed method (M1) and (B) shows the consumption of standard template method (M2). It is worth mentioning that M2 failed since frame 204, so the record in (B) stopped from that. However, our method worked well from the beginning. The average time consumption of M1 and M2 can be calculated as 78.42 and 648.16 ms.

REFERENCES

[1] K. Landzettel, C. Preusche, A. Albu-Schaffer, et al., Robotic on-orbit servicing-DLR's experience and perspective, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, IEEE, 2006, pp. 4587–4594.

[2] B. Liang, C. Li, L. Xue, et al., A Chinese small intelligent space robotic system for on-orbit servicing, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2006, IEEE, 2006, pp. 4602–4607.

[3] J.L. Jørgensen, M. Benn, VBS—The Optical Rendezvous and Docking Sensor for PRISMA, *DTU Space*, Elektrovej Bld. 327, DK—2800 Kgs Lyngby, Denmark.

[4] M. Delpech, P.Y. Guidotti, S. Djalal, T. Grelier, J. Harr, RF based navigation for prisma and other formation flying missions in Earth orbit, Adv. Astronaut. Sci. 135 (2010) 1533–1551.

[5] W. Wang, G.M. Xie, Online high-precision probabilistic localization of robotic fish using visual and inertial cues, IEEE Trans. Ind. Electron. 62 (2) (2015) 1113–1124.

[6] C. Lyn, G. Mooney, D. Bush, P. Jasiobedzki, D. King, Computer vision systems for robotic servicing of the Hubble Space Telescope, in: AIAA SPACE 2007 Conference & Exposition, 2007. (a) A. Yilmaz, O. Javed, M. Shah, Object tracking: a survey, ACM Comput. Surv. 38 (2006) 13. (b) B.D. Lucas, T. Kanade, An iterative image registration technique with an application to stereo vision, in: IJCAI, 1981, pp. 674–679.

[7] J. Shi, C. Tomasi, Good features to track, in: Proceedings CVPR'94, 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600.

[8] J. Fleet, Measurement of image velocity, (1992); (a)A. Makhmalbaf, M.-W. Park, J. Yang, I. Brilakis, P.A. Vela, 2D vision tracking methods' performance comparison for 3D tracking of construction resources, in: Construction Research Congress, American Society of Civil Engineers, 2010, pp. 459–469.

[9] H. Wang, Z.-H. Li, Z. Zhang, J.-Y. Tan, Video-based traffic parameter extraction with an improved vehicle tracking algorithm, in: CICTP 2012, 2012, pp. 856–866.

[10] M.-W. Park, I. Brilakis, Construction worker detection in video frames for initializing vision trackers, Autom. Constr. 28 (2012) 15–25.

[11] C.-Z. Xiong, Y.-G. Pang, Z.-X. Li, Y.-L. Liu, Y.-H. Li, Vehicle tracking from videos based on mean shift algorithm, in: ICCTP 2009@ Critical Issues in Transportation Systems Planning, Development, and Management, 2009, pp. 1–8.

[12] G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2004) 91–110.

[13] H. Bay, T. Tuytelaars, L. Van Gool, Surf: speeded up robust features, in: Computer Vision—ECCV 2006, Springer, Berlin, 2006, pp. 404–417.

[14] G. Takacs, V. Chandrasekhar, S. Tsai, D. Chen, R. Grzeszczuk, B. Girod, Unified real-time tracking and recognition with rotation-invariant fast features, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, 2010, pp. 934–941.

[15] M. Calonder, V. Lepetit, C. Strecha, P. Fua, Brief: binary robust independent elementary features, in: Computer Vision—ECCV 2010, Springer, Berlin, 2010, pp. 778–792.

[16] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, in: IEEE International Conference on Computer Vision (ICCV), 2011, 2011, pp. 2564–2571.

[17] C. Tomasi, T. Kanade, Detection and Tracking of Point Features, School of Computer Science, Carnegie Mellon Univ, Pittsburgh, 1991; (a)A.S. Mian, Realtime visual tracking of aircrafts, in: Digital Image Computing: Techniques and Applications (DICTA), 2008, 2008, pp. 351–356.

[18] V. Rabaud, S. Belongie, Counting crowded moving objects, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2006, 2006, pp. 705–711.

[19] D. Sugimura, K. Kitani, T. Okabe, Y. Sato, A. Sugimoto, Tracking people in crowds based on clustering feature trajectories using gait features and local appearances, in: Proc. of MIRU, 2009, pp. 135–142.
[20] Y. Tsuduki, H. Fujiyoshi, T. Kanade, Mean shift-based point feature tracking using SIFT, J. Inf. Process. Soc. 49 (2007) 35–45.
[21] Y. Tsuduki, H. Fujiyoshi, A method for visualizing pedestrian traffic flow using sift feature point tracking, in: Advances in Image and Video Technology, Springer, Berlin, 2009, pp. 25–36.
[22] B. Benfold, I. Reid, Stable multi-target tracking in real-time surveillance video, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011, 2011, pp. 3457–3464.
[23] J. Gong, F. Liu, C. Song, J. Cui, Z. Li, Research on the moving vehicle detection algorithm based on the motion vector, in: Instrumentation, Measurement, Circuits and Systems, Springer, Berlin, 2012, pp. 41–49.
[24] Y. Liu, J. Wang, P. Li, A feature point tracking method based on the combination of SIFT algorithm and KLT matching algorithm, J. Astronaut. 32 (2011) 1618–1627.
[25] L. Juan, O. Gwun, A comparison of sift, pca-sift and surf, Int. J. Image Process. 3 (2009) 143–152.
[26] G. Bradski, A. Kaehler, Learning OpenCV: Computer Vision With the OpenCV Library, O'Reilly Media, Inc., Farnham, 2008
[27] J. Williams, M. Shah, A fast algorithm for active contours and curvature estimation, CVGIP: Image Underst. 55 (1992) 14–26.
[28] R. Szeliski, Computer Vision: Algorithms and Applications, Springer, London, 2010; (a)A. Desolneux, L. Moisan, J.M. Morel, Meaningful alignments, Int. J. Comput. Vis., 40 (1) (2000) 7–23; (b)A. Desolneux, L. Moisan, J.M. Morel, Computational gestalts and perception thresholds, J. Physiol. Paris, 97 (2) (2003) 311–324.
[29] V. Pătrăucean, P. Gurdjos, R.G. Von Gioi, A parameterless line segment and elliptical arc detector with enhanced ellipse fitting, in: Computer Vision–ECCV 2012, Springer, Berlin, Heidelberg, 2012, pp. 572–585.
[30] X.H. Qin, R. Ma, W.P. Fu, et al., A line segments detection algorithm based on grad, Acta Photon. Sin. 41 (2) (2012) 205–209.
[31] Y.F. Yuan, P.M. Zhu, N. Zhao, et al., Automatic identification of circular mare craters based on mathematical morphology, Sci. Sin. Phys. Mech. Astron. 43 (3) (2013) 324–332.
[32] S. Basalamah, Histogram based circle detection, Int. J. Comput. Sci. Netw. Secur. 12 (8) (2012) 40–43.
[33] D.X. Zeng, X.R. Dong, R. Li, Optical measurement for spacecraft relative angle, J. Cent. South Univ. (Sci. Technol.) 38 (2007) 1155–1158.
[34] E. Rosten, R. Porter, T. Drummond, Faster and better: a machine learning approach to corner detection, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 105–119.
[35] P. Huang, J. Cai, Z. Meng, Z. Hu, D. Wang, Novel method of monocular real-time feature point tracking for tethered space robots, J. Aerosp. Eng. 27 (6) (2013) 04014039.
[36] W. Xu, B. Liang, C. Li, Y. Xu, Autonomous rendezvous and robotic capturing of noncooperative target in space, Robotica 28 (2010) 705–718.