This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

A Direct 3D Object Tracking Method Based on Dynamic Textured Model Rendering and Extended Dense Feature Fields

Leisheng Zhong, Ming Lu, and Li Zhang

***Abstract*—We propose a novel method for robust 6-dof pose tracking of rigid objects from monocular images. In our method, 3D object tracking is achieved by directly aligning video frames to dynamic templates rendered from a textured 3D object model. Unlike previous methods which usually utilize a small number of discrete templates to align with video frames, we employ online textured model rendering to create dynamic templates in continuous pose space according to the previously estimated object pose. In this way, pose estimator could be easily converged to the optimal state. Besides, the rendered template also helps to detect the occlusion area by comparing it with the current frame, making our method highly robust to partial occlusions. The performance of our method is further improved by introducing a generic representation of dense images features, which we call Extended Dense Feature Fields (EDFF). Different kinds of pixel-level image features can be added to the EDFF and be optimized simultaneously in a unified Gauss-Newton optimization scheme. Attributing to dynamic templates from textured model rendering and complementary features in EDFF, our method is able to deal with poor-textured and specular objects, as well as lighting variation and heavy occlusions. While our method is quite simple and straight-forward, it achieves competitive or even superior results compared to the state-of-art on challenging datasets.**

***Index Terms*—3D object tracking, dynamic templates, textured model rendering, direct image alignment, dense image features, occlusion detection.**

I. Introduction

ROBUSTLY tracking the 6-dof pose of an object is an essential problem in 3D computer vision [1]–[4]. It is also critical for the development of robotic manipulation and augmented reality applications [5]–[7]. Existing model-based 3D object tracking methods typically rely on a simple triangle mesh and a small set of template images of the object [7]–[9], as shown in Fig. 1(b). These methods are prone to fail when the object pose changes in a wide range through the video sequence. The problem lies in the difficulty to build a complete template set that accounts for all possible poses in 6-dimensional pose space, which leads to tracking failure when a proper template for current pose is not available. Another category of methods choose to align consecutive video frames over a triangle mesh model [10], as shown in Fig. 1(c). These methods benefit from the temporal consistence between successive frames. However, they sometimes tend to fail due to background interference. That is, besides the foreground pixels, the background pixels around the object are also very similar in neighboring video frames, so they would gradually drag the tracking result to background area and finally lead to large pose drifts. On the other hand, robustly tracking the pose of a partially occluded object is still a difficult problem. Some previous works try to solve this problem by employing robust dense features [8] or sparse keypoints [11]. These strategies are effective in general cases, but still fail in heavily occluded situations.

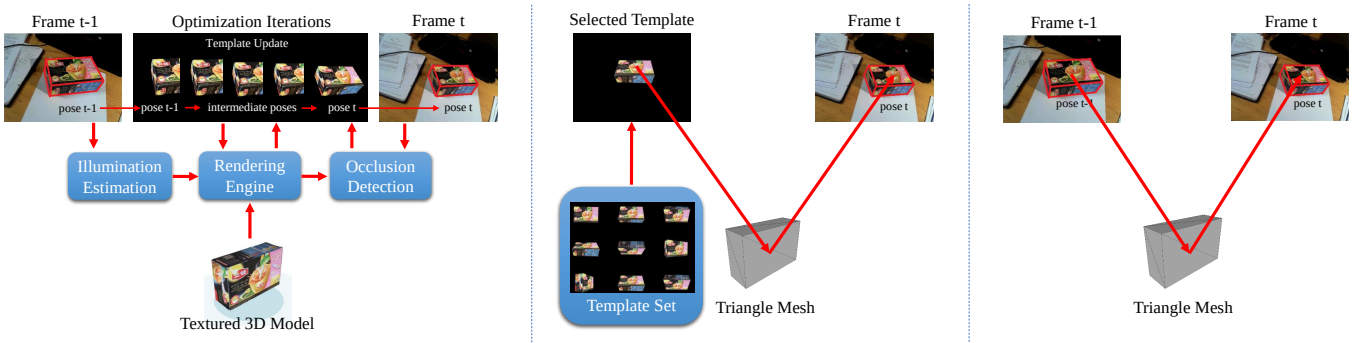
Therefore, we introduce a dynamic template-based method to resolve these problems. An overview of the proposed method is shown in Fig. 1(a). Instead of building a static template set with fixed number of template images offline, we create dynamic templates in continuous 6-dof pose space online by rendering a pre-built textured 3D model of the object. It is thus no longer a problem to find a suitable template for each video frame. Considering temporal consistency, object model with the latest tracked pose is rendered as template for tracking the current frame. The template is also dynamically updated with each intermediate pose output during optimization iterations, until final convergence. Note that this template updating strategy is not possible in previous methods which only adopt wireframe models. On the other hand, the rendered template is always background-free and non-occluded, which acts as a natural segmentation of the object and eliminates the influence from background and occluded pixels. We employ an efficient occlusion detection module by comparing the rendered template with the current video frame. We also add an illumination estimation module in our method. We render the tracked object with the estimated illumination, aiming to get a realistic rendering to best resemble the current frame. Experiments show that using dynamically rendered templates to align with current image is much more robust and stable, thus yielding significant improvements in tracking performance.

The general framework of our 3D pose tracking algorithm is similar to those of previous works [8], [12]. We use direct image alignment[1] [14] to iteratively optimize the pose parameters. The direct alignment process is presented in Fig. 2. Most previous works only use a single type of image

Leisheng Zhong, Ming Lu and Li Zhang are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China. E-mails: {zls13,lu-m13}@mails.tsinghua.edu.cn, chinazhangli@mail.tsinghua.edu.cn.

[1]In some of the previous works, 'direct image alignment' is called 'dense image alignment'. Also, 'direct method' (for 3D tracking) is sometimes called 'dense method'. They all belong to the same kind of methods based on Lucas-Kanade algorithm. According to [13], 'direct' is a more appropriate term. So we use 'direct' throughout this paper.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology



(a) Align with Dynamic Templates (our method) **(b) Align with Discrete Templates** **(c) Align Consecutive Frames**

Fig. 1. Comparison of the proposed method with previous direct tracking methods. (a) We propose to align the current frame with dynamically rendered templates from a pre-built textured 3D object model. (b) Some previous works try to align the current frame with a template selected from a small set of discrete templates. (c) Other previous works align consecutive video frames over a triangle mesh model.

feature for alignment, among which the most commonly used one is image intensity [10], [12], [15]. Crivellaro *et al.* [8] introduce gradient-based Descriptor Fields in place of pixel intensities to improve the alignment robustness. In our experiments, we find it even more robust to combine different dense features together. So we build on [8] to formulate a generic representation of dense image features. We refer to this extension as Extended Dense Feature Fields (EDFF). EDFF is constructed using different combinations of pixel-level image features. Experiments show that different kinds of image features tend to be complementary to each other and an EDFF involving multiple features often leads to more robust pose tracking results. The proposed EDFF also makes it possible to handle monocular object tracking and RGB-D object tracking in a unified manner, because depth information can be trivially added to the EDFF in our framework. Moreover, with careful selection of features in EDFF, our method is able to deal with various difficult tracking situations, such as tracking poor-textured objects or tracking in environments with evident illumination changes. Some of the tracking targets and environments in our experiments are shown in Fig. 3.

At the core of our method is the rendering of textured 3D models as dynamic templates. In discrete template case, each 2D template is just a partial representation of the 3D object. The missing information makes direct image alignment harder in both initialization and optimization procedures. We take advantage of a pre-built textured 3D model, which can be seen as a full representation of the object, to render out dynamic templates online. The benefits lie in several aspects: Firstly, a good initial template for the current frame is automatically acquired by rendering the object with the latest tracking result. This initial template is almost always in the region of convergence for aligning with the current frame. Besides, dynamic template updating during optimization also results in easier and faster convergence. Secondly, the rendered template is a natural segmentation of the object from background, which constrains the pixels involved in alignment optimization and helps to avoid drifts caused by background interference. Thirdly, with the full information from the textured model, it is easy to detect object occlusions and discard the occluded pixels before pose optimization. To the best of our knowledge, we are the first to apply dynamic textured model rendering to a direct 3D object tracking method with explicit occlusion detection and illumination estimation strategies.

The main contributions of this paper are:

1. Applying dynamic textured model rendering to direct 3D object tracking. This technique enables online template creation and updating, helps to deal with background clutters and partial occlusions, thus remarkably improves tracking performance in challenging situations.

2. Constructing a generic representation of dense image features for direct image alignment. With this representation, complementary features are combined in a unified pose optimization framework to further enhance tracking robustness.

3. Employing a simple yet efficient occlusion detection process in the tracking pipeline. This step eliminates the influence from occluded pixels and significantly improves the tracking results on heavily occluded video sequences.

II. Related Work

A vast variety of 3D object tracking methods have been proposed in the past several decades. Lepetit *et al.* [1] give a comprehensive and detailed survey on early monocular model-based object tracking algorithms. According to the survey, early methods try to use feature points or strong edges to handle this task. But feature point-based methods are limited to well-textured objects and edge-based methods perform badly with background clutter or partial occlusions. In the following, we only focus on some recent works that are closely related to our method.

Template-based direct image alignment approaches have become very popular in recent years [8], [14], [16], [17]. Most of them are built on the famous direct image alignment framework: Lucas-Kanade algorithm [18]. L-K algorithm is originally designed for parameter estimation of 2D image transformations, but is extended to 3D object tracking by employing a 3D warping function [19] with a known 3D object model. Crivellaro *et al.* [8] propose a gradient-based dense descriptor to improve the alignment performance with poor-textured and specular objects. Although this method is very robust owing to the well-designed Descriptor Fields, it still gets

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

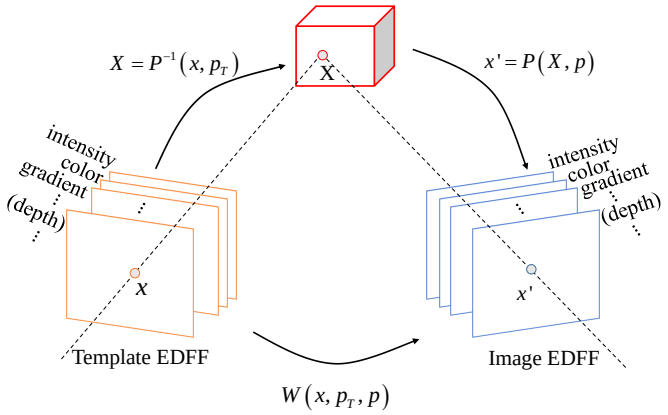


Fig. 2. Extended Dense Feature Fields (EDFF) for direct image alignment. We propose a generic representation for direct image alignment applied to 3D object tracking. Arbitrary pixel-level features, such as intensity, color, gradient and depth, can be freely added to the EDFF. These features are then efficiently processed at the same time to give out the optimal pose estimate. The 3D warping function $W(x, p_T, p)$ is a transfer function that performs back-projection and re-projection over the 3D model to find corresponding pixels between the template and image.

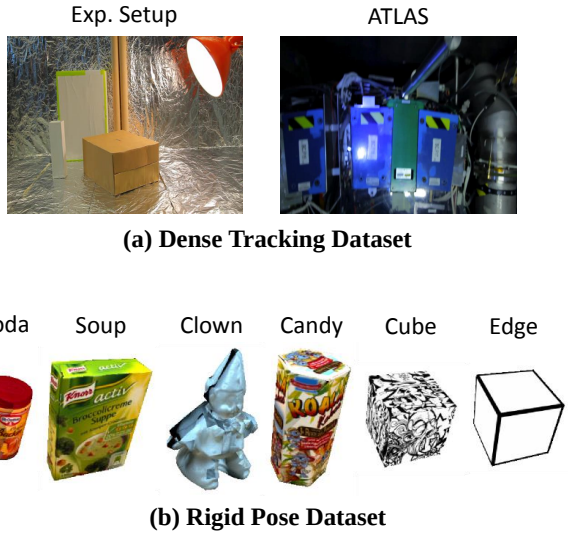


Fig. 3. Illustration of the tracked objects in our experiments. Our method can successfully track the 6-dof pose of both well-textured and poor-textured (e.g. clown and edge) objects, under difficult situations such as lighting variation (e.g. Exp.Setup) and specularity (e.g. ATLAS). The objects are taken from Dense Tracking Dataset [8] and Rigid Pose Dataset [11] respectively.

a relatively low success rate when tracking objects with fast and large-scale movements, such as in video ATLAS#2 [8]. We extend their work by employing dynamic template rendering through pose optimization, and generalizing the gradient-based Descriptor Fields to multi-feature-based Dense Feature Fields. We compare our results with [8] in Section IV. Caron *et al.* [20] propose a model-based visual pose estimation and camera localization method. They also render textured models to compare with video sequences, but they mainly focus on city-scale scene models for vehicle localization, so that 3D object tracking in complex situations is not explicitly addressed. A recent work [10] employs a constrained objective function to model intensity variations by using the surface normal of the object under Lambertian assumption. Inspired by them, we add an illumination estimation module in our method, which can be seen as a decoupled version of their strategy. We use linear combinations of the first nine spherical harmonics (SH) to model complex lighting conditions [21]. But we choose not to couple this factor into the final objective function. Instead, we first estimate the illumination of the video frame and then render the object under the estimated illumination as template. This ensures the illumination consistency between the rendered template and the video frame, while keeping the final objective function simple and unchanged.

A different class of approaches use 3D level set functions to maximize the discrimination between statistical foreground and background appearance models [22], [23]. In these region-based methods, PWP3D [22] is the most famous, and is still among the state-of-art. These methods use different methodology with ours. They take advantage of image statistics and fit them in a probabilistic framework, while we directly utilize dense image features. Although region-based methods have been successful in 3D texture-less object tracking, they are not guaranteed with good results in complex scenes [10], [11].

Pauwels *et al.* [4], [11] propose a stereo-based 3D object tracking method that combines dense stereo cues, optical flow, AR flow and sparse keypoint features together, and provides a synthetic dataset for evaluation. Because of the rich features they adopt in their method, it performs much better than region-based and particle-filter-based methods [22], [24]. Although our method is monocular-based and only uses dense image features, it obtains competitive results compared to [11] in noise-free and noisy sequences of the provided dataset. After applying occlusion detection, we even get superior results in the occluded sequences, which proves the efficiency of our method in handling heavy occlusions. More details of the experiments are discussed in Section IV.

In this paper, we assume an initial object pose is available and focus on the frame-to-frame pose tracking problem. 3D object detection methods (such as [25]) can be combined with our method for initialization.

III. Proposed Method

In this section, we present our 3D object tracking pipeline. As shown in Fig. 1(a), we track a new frame by aligning it to dynamically rendered object templates. As a pre-processing step, we first estimate the illumination of the previous frame. Next, an appropriate template for the current frame is acquired by rendering the textured 3D model with the latest tracking result and the estimated illumination. After that, we detect the occlusion area by comparing the current frame with the newly rendered template and discard the occluded pixels. Finally, we iteratively optimize the pose parameters in the direct image alignment framework with the proposed Extended Dense Feature Fields. During each iteration, the template is dynamically updated according to intermediate pose outputs to accelerate convergence of pose optimization.

The proposed 3D object tracking pipeline is summarized in Algorithm 1. We divide our tracking pipeline into five components: (A) Direct Image Alignment for 3D Object

Tracking, (B) Dynamic Template Rendering, (C) Extended Dense Feature Fields, (D) Illumination Estimation, and (E) Occlusion Detection. Among them, (A) is the general framework for direct 3D object tracking methods, and we present it here for completeness. (B), (C), (D) and (E) are newly introduced components in our method. More specifically, (B) and (C) are the main contributions of the proposed method, which distinguish our method from the previous works. (D) and (E) are very important components to help improving the tracking performance of our method in challenging situations.

In the following, we discuss each component of the above tracking pipeline in detail.

A. Direct Image Alignment for 3D Object Tracking

We start by introducing the general framework of direct 3D object tracking. Similar to previous works, we use the direct image alignment framework [14] as our basic method. The original Lucas-Kanade algorithm [18] aims to minimize the sum of squared error (SSD) between intensities of template image T and input image I:

$$E(p) = \sum_x \|Int(I, W(x,p)) - Int(T,x)\|^2 \qquad (1)$$

here Int refers to pixel intensities. $W(x,p)$ is a 2D warping function which transfers template point x to image point x' through a 2D transformation (such as affine transformation or homography) with parameter p.

In order to apply this method to 3D object tracking, we use a 3D warping function $W(x, p_T, p)$ [19] to align current image with template through a 3D object model. This function transfers template point x to image point $x' = W(x, p_T, p)$ by back-projecting x to the 3D model in pose p_T and then re-projecting it to image plane in pose p. The 3D warping process is shown in Fig. 2. Crivellaro *et al.* [8] suggest to use gradient-based Descriptor Fields instead of image intensities in (1). We further extend this concept and propose a unified optimization scheme: Extended Dense Feature Fields (EDFF). The final error function is:

$$E(p) = \sum_x \|F(I, W(x, p_T, p)) - F(T,x)\|^2 \qquad (2)$$

where $p = [t_x, t_y, t_z, \theta_x, \theta_y, \theta_z]$ is the vector representation of 6-dof object pose for the current frame I, and p_T is the 6-dof object pose for template T. F denotes the proposed EDFF, which will be detailed in Section III-C.

B. Dynamic Template Rendering

Some of the previous direct 3D tracking approaches rely on a small number of discrete templates [8]. These discrete templates usually compose of a set of template images and the ground-truth poses corresponding to each image. When a new frame comes, they first try to select the most similar template from the template collection and then perform direct image alignment between the selected template and the new frame. These methods tend to fail when the current object pose is far from any of the template poses or the template is not properly selected. We find these circumstances frequently happen in our experiments. This problem is more critical in object tracking tasks than camera tracking tasks because the tracked object usually occupies only a small region of the image, making it more difficult to find a template which is similar enough for convergence of pose optimization. The problem can be partly solved by constructing a larger template set, but growing number of templates would greatly increase the computation in template selection.

To overcome these problems, we use a pre-built textured 3D model to dynamically create template images which is always supposed to be similar to the current frame. This approach is not new in object tracking researches. It has been used in sparse feature point-based tracking [26], model-based tracking with mutual information [20], and also in [11] as a complementation for optical flow to recover from motion drifts. We find it also very beneficial to apply textured model rendering to direct 3D object tracking methods. In the following, we present the details of 3D textured model creation, dynamic template rendering and online template updating in our method.

1) 3D Textured Model Creation: The main difference of creating 3D models between the proposed dynamic template-based method and the discrete template-based method [8] is that [8] only creates wireframe models, but we need to create textured models. There are several different ways to create a textured 3D object model. For example, for simple objects such as boxes, we can first create a wireframe model using 3D modeling/editing softwares [27], [28]. Then we can use the *texture mapping* tools in these softwares to map a texture image to the wireframe model and obtain a textured 3D model. For more complex objects, we can use another type of 3D modeling softwares [29], [30] to directly generate textured 3D models from a set of pictures, as suggested by [4].

2) Dynamic Template Rendering: With a textured 3D model, template images of the object in different poses can be easily created using a rendering engine such as the OpenGL-based renderer [31]. We render an initial template for frame k using the latest tracked pose from frame $k-1$, which guarantees the similarity between the template and the image. In order to render realistic template images from a 3D object model, the OpenGL-based rendering pipeline includes a *Modelview Transform* and a *Projective Transform* for all of the vertices [22]. Firstly, a Modelview Transform is applied to transform the vertex coordinates from the model coordinate frame to the camera coordinate frame:

$$X_c = T_m X_m = \begin{bmatrix} R & t \\ 0_{1\times 3} & 1 \end{bmatrix} X_m \qquad (3)$$

where T_m is the Modelview Transform matrix, in which the rotation matrix R and the translation vector t are derived from the latest tracked pose of the object. X_m and X_c are the vertex coordinates in model coordinate frame and in camera coordinate frame respectively.

A Projective Transform is then applied to project the vertices to the virtual image plane. Specifically, we render the model using the exact intrinsic parameters of the same camera that captures the video. After that, the rendering engine determines the pixel colors on the rendered image according to the

texture coordinates of the corresponding vertices in the pre-built textured 3D model. Also, the depth values of each pixel is acquired by the rendering engine through *ray tracing*. The 3D location with respect to each pixel can be calculated from its depth value via back projection, which is then used in the 3D warping function in (2).

3) Online Template Updating: During pose optimization iterations, the template is also dynamically updated according to each intermediate pose estimate to further increase convergence speed. Since we use Gauss-Newton method to iteratively optimize the pose parameters, a pose update is obtained after each iteration. Thanks to the textured 3D model utilized in our method, the template can also be updated after each iteration. The new template (after each iteration) is rendered in the same way as the initial template (of each frame), except that we use the updated pose parameters to perform Modelview Transform instead of using the latest tracked pose from the previous frame.

Experiments demonstrate that the online template updating strategy could effectively increase convergence speed and reduce the number of iterations required for pose optimization. Although this strategy needs extra computation for re-rendering the object model after each iteration, the decrease of iteration numbers makes a good compensation. Moreover, unlike the discrete template-based methods, the proposed method does not require a template selection step for each frame. In the template selection step, they have to compute the Normalized Cross-Correlation (NCC) between the current frame and each template from the template set, and choose the template with the largest NCC score [8]. This step could also be computationally expensive, especially when using a large template set. Overall, the increased processing time of our method compared to [8] is acceptable, as detailed in Section IV-E.

The implementation of dynamic template rendering in our method is detailed in Algorithm 1. Experiments show greatly improved convergence rate with the help of dynamic template rendering, especially in challenging videos with large and fast movements such as ATLAS#2 in Dense Tracking Dataset [8], which will be presented in Section IV. Dynamic template rendering also benefits the occlusion detection step applied in our method, which will be detailed in Section III-E.

C. Extended Dense Feature Fields

Dense gradient-based Descriptor Fields proposed by [8] has been proved more robust than image intensities in direct image alignment algorithms. We borrow the idea of Descriptor Fields, and further extend it to combinations of pixel-level image features, which we call Extended Dense Feature Fields (EDFF). As shown in Fig. 2, different kinds of pixel-level features can be freely added to or discarded from EDFF to best adapt to the physical properties of the current tracked object. We can stack up any type of features we want, and the unified Gauss-Newton optimization step will automatically combine them together to give out the optimized pose estimate. For example, with intensity, color and gradient in EDFF, we can process monocular object tracking. When depth information is available, we can simply add depth in EDFF, thus we have an RGB-D object tracking implementation. In this way, we can handle RGB and RGB-D based 3D object tracking in a unified manner.

Suppose we would like to stack up d dimensions (channels) of features in EDFF, then $E(p)$ is the weighted sum of SSD between each dimension of template and image features:

$$E\left(p\right)=\sum_{i=1:d}w_i\left(\sum_x\left\|f_i\left(I,W\left(x,p_T,p\right)\right)-f_i\left(T,x\right)\right\|^2\right) \quad (4)$$

Here, the weight w_i adjusts the influence of different features, and is taken as $w_i=\frac{1}{\sigma_i^2}$ in our experiments. σ_i^2 is the variance of the i-th feature f_i. (4) is equivalent to (2) where $F=\left[\sqrt{w_1}f_1,\sqrt{w_2}f_2,...,\sqrt{w_d}f_d\right]^T$.

We use standard Gauss-Newton method to iteratively minimize the error function $E(p)$. At each iteration, we optimize the increment δp by:

$$E\left(\delta p\right)= \sum_{i=1:d}w_i\left(\sum_x\left\|f_i\left(I,W\left(x,p_T,p\right)\right)+J_i\left(x,p\right)\delta p-f_i\left(T,x\right)\right\|^2\right) \quad (5)$$

Here, $J_i\,(i=1,...,d)$ are the jacobians. This weighted least squares problem can be efficiently solved from a single stacked normal equation:

$$J^T\mathbf{W}J\delta p=J^T\mathbf{W}\left(F\left(I,W\left(x,p_T,p\right)\right)-F\left(T,x\right)\right) \quad (6)$$

where $J\in R^{Nd\times m}$ (N denotes the number of pixels involved in optimization, d denotes the dimension of dense features, m denotes the number of pose parameters) is the stacked jacobian, $Res=F\left(I,W\left(x,p_T,p\right)\right)-F\left(T,x\right)\in R^{Nd\times 1}$ is the stacked residuals. $\mathbf{W}$ is the weight matrix. Since we only need to solve one linear system per iteration, the computation cost of our algorithm scales well with respect to the dimension of EDFF.

A large variety of features could be used in our unified dense feature fields, such as image intensities, colors, intensity gradient, chrominance gradient, depth information, or other pixel-level features. The components of EDFF should be chosen according to the characteristics of the object and environment. Generally speaking, intensity and color are sensitive to complex illumination changes, especially for specular objects. In these situations, gradient-based dense features would be a better choice because they are relatively insensitive to illumination changes and specularity [8]. But it would be beneficial to add color or intensity in EDFF when the lighting condition of the environment is stable, especially for extremely poorly textured objects that show nearly no gradient details. Depth information would improve the tracking results in most cases.

D. Illumination Estimation

In order to deal with complex lighting variations, we add an illumination estimation step before rendering out templates for

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

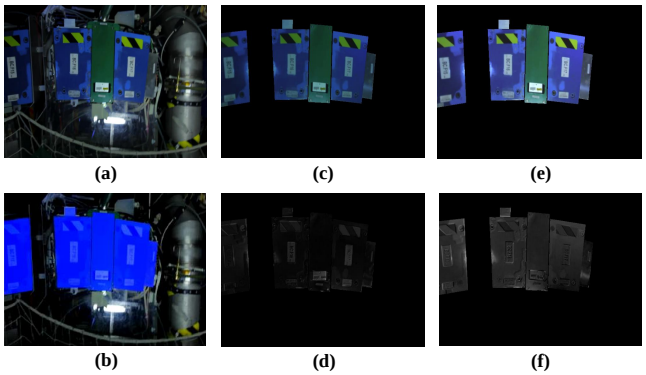


Fig. 4. Illumination Estimation. (a) A sample frame from ATLAS#2 in Dense Tracking Dataset [8]. (b) Object mask acquired from tracking result. Only pixels inside the mask are involved in illumination estimation. (c, d) Template re-lighted by the estimated illumination (of the previous frame) and the error image compared to the current frame. (e, f) Rendered template without re-lighting and the corresponding error image. After employing the estimated lighting parameters to the rendering engine, the rendered template is much more similar to the video frame.

the current frame. Although the lighting condition may change all along the video sequence, we assume that neighboring frames usually share similar illumination. So we first estimate the illumination of the previous frame according to the tracked foreground region, and then use the estimated illumination to re-light the textured 3D model and create templates for tracking the current frame. According to [21], a nine-dimensional linear subspace spanned by the zero, first and second order spherical harmonics (SH) accounts for at least 98 percent of the variability in the Lambertian reflectance functions. So we use the first nine spherical harmonics basis functions [32] to represent illumination. The RGB image I can be formulated as:

$$I(i,j) = \rho(i,j) \sum_{k=0}^{8} l_k H_k(\mathbf{n}(i,j)) \tag{7}$$

where $H_k(\mathbf{n})$ are the SH basis functions, l_k are the SH coefficients, $\rho(i,j)$ are the albedos corresponding to each pixel, $\mathbf{n}(i,j) = (n_x, n_y, n_z)$ are the surface normals.

Although $\mathbf{n}(i,j)$ can be directly calculated from the 3D model (rendered with the latest tracked pose), simultaneously solving for the SH coefficients l_k and the albedo map $\rho(i,j)$ in (7) is still an underconstrained problem. Therefore, we follow the previous works [32], [33] and solve for l_k and $\rho(i,j)$ in two steps.

Firstly, we temporarily set the albedo map to an uniform map, i.e., $\rho(i,j) = 1$. In this way, we can solve for the SH coefficients l_k by minimizing the following error function:

$$E_L = \sum_{i,j} \left\| I(i,j) - \sum_{k=0}^{8} l_k H_k(\mathbf{n}(i,j)) \right\| \tag{8}$$

This is a standard least squares problem. The SH coefficients can be calculated by solving a linear system. We calculate the lighting in separate RGB channels and obtain a 27 dimensional SH vector to represent the illumination, which is then used to re-light the 3D model using the rendering engine.

Fig. 5. Occlusion Detection. (a) A heavily occluded frame from Rigid Pose Dataset [11]. The tracked box is largely occluded by the teddy bear. (b) The template rendered with tracked pose from last frame. The rendered template is always clear and non-occluded, which contributes to a simple but efficient occlusion detection strategy. (c) The detected occlusion area. Pixels inside the area are discarded from pose tracking optimization.

Secondly, with the calculated SH coefficients l_k, we can go back to solve for the albedo $\rho(i,j)$. We obtain a dense albedo map by dividing the RGB image by the lighting term for each pixel:

$$\rho(i,j) = \frac{I(i,j)}{\sum_{k=0}^{8} l_k H_k(\mathbf{n}(i,j))} \tag{9}$$

Applying the above two steps to the rendered template, we obtain the albedo map of the rendered template T (denoted by $\rho^T(i,j)$). Applying the first step to the current frame, we obtain the lighting parameters of the current frame I (denoted by l_k^I). Then we re-light the template using the estimated lighting parameters of the current frame:

$$T^{relight}(i,j) = \rho^T(i,j) \sum_{k=0}^{8} l_k^I H_k(\mathbf{n}(i,j)) \tag{10}$$

We demonstrate the illumination estimation step in Fig. 4. The illumination estimation step makes our textured model rendering more realistic. It is now possible to create template that has very similar appearance with the video frame, which makes the following steps (such as occlusion detection and direct image alignment) more robust and stable.

E. Occlusion Detection

Occlusion handling is one of the most difficult problems in 3D tracking. Some previous works address this problem by using sparse keypoints [11], or employing robust dense features [8]. These strategies can handle certain types of occlusions, but would fail in more difficult situations. In this paper, we present a new realistic textured model rendering based strategy for handling partial occlusions. The proposed strategy could accurately detect the occlusion area by comparing the occluded image with the rendered occlusion-free template. As a result, we are able to tackle the problem of partial occlusion more effectively than previous works.

In our method, an occlusion detection step is performed before pose optimization. The occlusion area is detected as follows:

Firstly, we render the 3D model with the latest tracked pose to get an occlusion-free template, as shown in Fig. 5(b). Directly comparing the template with the new frame

is not appropriate due to the object motion between successive frames. So we apply a Gaussian filter to both the template and the new frame to compensate for the motion. The parameters of Gaussian filtering depend on the average motion in neighboring frames. We use an 11×11 Gaussian kernel with standard deviation of $\sigma = 5$ throughout our experiments. After that the smoothed template and smoothed new frame are compared to determine the occlusion area. In order to further alleviate the distraction from pixels near occlusion edges, we slightly dilate the detected occlusion area to get an expanded estimation of occluded pixels. We find the dilation operation critical in the improvement of tracking performance. The reason is that leaving out a small number of non-occluded pixels near occlusion edges does not affect the tracking performance due to the robustness of direct tracking method, but reserving them would dramatically influence the alignment because of the strong gradients they contribute to. The above process is formulated as:

$$Occl = [|G^\sigma * I - G^\sigma * T| - \beta]^+ \oplus Strel \qquad (11)$$

where $Occl$ refers to the detected occlusion area, G^σ is the Gaussian smoothing kernel, β is the detection threshold, $[\cdot]^+$ operator is defined as:

$$[x]^+ = \begin{cases} 1, & if\, x > 0 \\ 0, & otherwise \end{cases}$$

$\oplus$ is the image dilating operator, and $Strel$ is the structural element used to dilate the detected occlusion area. In our experiments, $Strel$ is set to a disk-shaped structural element with a radius of 5 pixels.

After the detection of the occlusion area (as shown in Fig. 5(c)), we discard those occluded pixels and use the rest valid pixels to track the new frame. Evaluation of our method on heavily occluded video sequences are presented in Section IV.

Algorithm 1 3D Object Tracking Pipeline

Input: Textured Model M, previous frame I_{k-1}, current frame I_k, previous pose p_{k-1}
Output: Current pose p_k

Render M in pose p_{k-1} to get the foreground mask of I_{k-1};
Estimate the illumination of I_{k-1} based on (8), using only pixels inside the foreground mask;
Re-render M in pose p_{k-1} with the estimated illumination to create the initial template T_0, read out the 3D coordinates of each foreground pixel from the renderer;
Calculate the template EDFF $F(T_0, x)$ and the image EDFF $F(I_k, x)$;
Detect the occlusion area based on (11), discard the occluded pixels;
Set $p_k = p_{k-1}$, $p_T = p_{k-1}$, $T = T_0$;
for a number of iterations **do**
  Calculate the stacked jacobian J and stacked residual $Res = F(I, W(x, p_T, p_k)) - F(T, x)$ in (6), solve for the pose increment δp;
  Update current pose $p_k = p_k + \delta p$;
  if converged **then**
    Output p_k;
    Break;
  else
    Update template pose $p_T = p_k$;
    Update template T by rendering M in pose p_T, read out the 3D coordinates of each foreground pixel from the renderer, re-calculate the template EDFF $F(T, x)$;
  end if
end for

IV. Experiments

A. Datasets and Implementation Details

We evaluate the proposed method on Dense Tracking Dataset [8] and Rigid Pose Dataset [11]. These two datasets provide both synthetic and real-world video sequences under challenging conditions, and are ideal for testing the robustness of our method. An illustration of the objects and environments in these two datasets are shown in Fig. 3.

Firstly, we evaluate our dynamic template-based method on Dense Tracking Dataset to compare with the discrete template-based method [8]. The effectiveness of our illumination estimation module is also demonstrated.

Secondly, we compare the overall performance of our method with 5 state-of-art methods on Rigid Pose Dataset. We also evaluate our method with and without the occlusion detection module, as well as using different dense features in EDFF.

To increase the basin of convergence, we employ a multi-scale optimization scheme by smoothing the image with Gaussian kernels. We use 4 levels, with $\sigma = [8, 4, 2, 0]$ ($\sigma = 0$ means the original image). The kernel size of each level is set to $2\sigma+1$. Pose optimization is started at the coarsest level, and the estimated pose is used as initial pose for the next level. The maximum number of iterations for each level is set to 10. We stop the iteration if the difference of average alignment error between two iterations is below 1e-4.

B. Results on Dense Tracking Dataset

The Dense Tracking Dataset [8] provides real-world video sequences of poor-textured objects in two challenging environments, including strong moving light source, bright specularities and motion blur caused by fast movement (as shown in Fig. 3(a)). Thanks to the discriminative gradient-based Descriptor Fields, [8] obtains very good tracking results using only 1 and 24 discrete templates for two different environments respectively. But they still get a relatively low tracking success rate on ATLAS#2, which includes fast and wide-range motions. The problem lies in the limitation of using discrete templates, as discussed in previous sections. By employing dynamic template rendering, we further improve the tracking success rate on this dataset.

Since only wireframe models are used in [8], the authors provide only wireframe models of the objects in Dense Tracking Dataset. So we create textured object models using a 3D

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

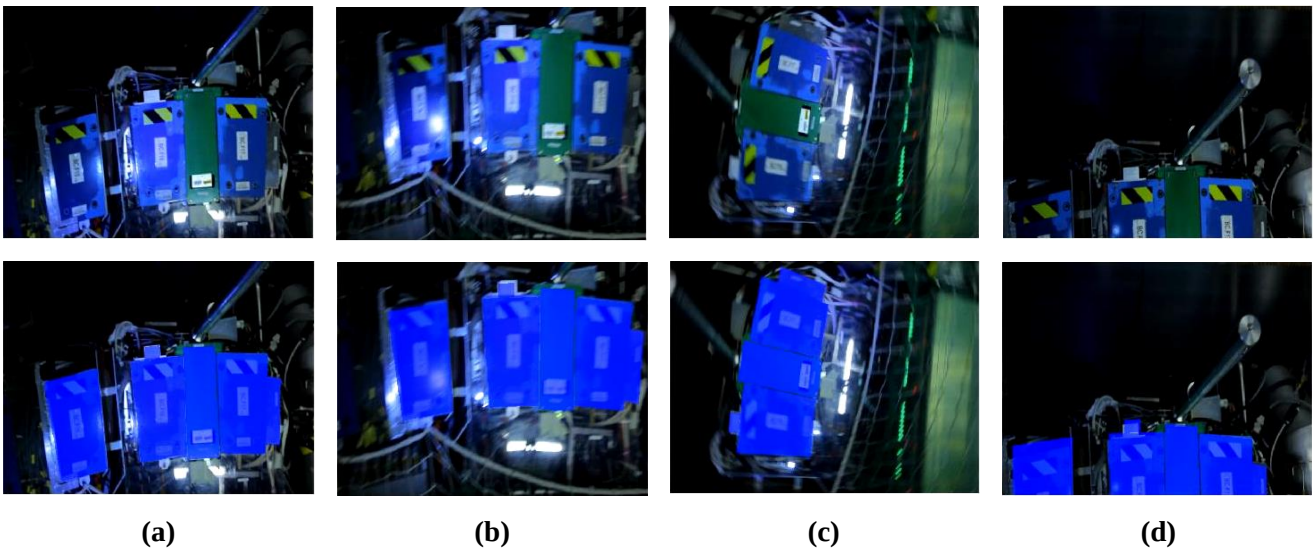


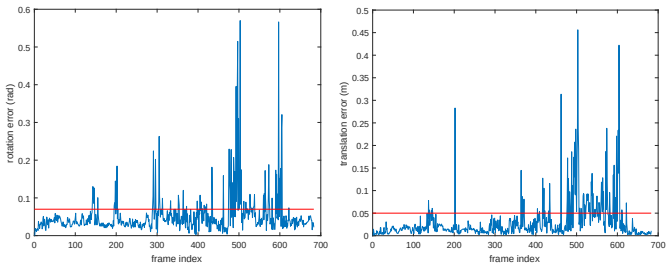
Fig. 6. Sample frames from video ATLAS#2 in Dense Tracking Dataset [8] and the corresponding tracking results. First row: Sample frames. Second row: Tracking results superimposed on the original images. Our method can successfully track the object with (a) illumination changes, (b) specularity, (c) large scale translation and rotation, and (d) partial occlusion.

model editing software named *AC3D* [28]. For each model, we first choose a proper image from the video sequences as texture image, in which the object is clear and non-occluded. Then we align the texture image with the 3D model using the UV mapping tool in *AC3D*. The resulting textured 3D model is used in the textured model rendering process in our tracking pipeline.

Here we use the same evaluation criteria as in [8]. We evaluate the methods by reporting the percentage of frames that are successfully tracked. To decide whether a frame is successfully tracked, we compute a rotation error R_{err} and a translation error t_{err}. The rotation error is defined as the L-2 distance between the vector representation of the estimated rotation and the ground truth (in radians), $R_{err} = \left\|(\theta_x, \theta_y, \theta_z)^T - (\theta_{x-true}, \theta_{y-true}, \theta_{z-true})^T\right\|$. The translation error is defined as the L-2 distance between the estimated translation and the ground truth (in meters), $t_{err} = \left\|(t_x, t_y, t_z)^T - (t_{x-true}, t_{y-true}, t_{z-true})^T\right\|$. A frame is successfully tracked if the rotation error and translation error are both smaller than the thresholds. For fair comparison, we use the same thresholds for rotation error ($\varepsilon_{rot} = 0.07\ radians$) and translation error ($\varepsilon_{transl} = 0.05\ meters$) as in [8], and we only use gradient in our EDFF.

Table I shows the evaluation results of our method compared to [8]. Our method generally gets a higher tracking success rate compared to the methods using 1^{st}-order Descriptor Fields and 1^{st}- and 2^{nd}-order Descriptor Fields, especially on ATLAS#2. We also evaluate our method with and without the illumination estimation step. Improved results are obtained with the help of the illumination estimation module. The results also demonstrate the superior performance of gradient over intensity and color when tracking specular objects in the environment with complex illumination changes.

The rotation and translation errors of our method over the ATLAS#2 sequence are shown in Fig. 7. Over 85% of the total frames are successfully tracked by our method on this most challenging sequence, which greatly outperforms discrete template-based methods. Some sample frames from ATLAS#2 and the corresponding tracking results are illustrated in Fig. 6.



(a) (b)

Fig. 7. Pose estimation errors of our method over the ATLAS#2 sequence. (a) Rotation errors (in radians). (b) Translation errors (in meters). The red horizontal lines are the thresholds used to decide whether the frames are successfully tracked.

We clarify that [8] mainly focuses on the efficiency of Descriptor Fields. They show that the gradient-based Descriptor Fields remains discriminative even after smoothing by large Gaussian kernels, thus significantly broadening the region of convergence of the alignment optimization. So they use very few discrete templates to illustrate the advantage of Descriptor Fields. If more templates were used in their method, the results would be better. But as discussed in Section III-B, constructing a larger template set would add to the difficulty in template selection, which also affects the overall performance of the algorithm.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

TABLE I
TRACKING SUCCESS RATE (IN %) ON DENSE TRACKING DATASET

	Exp#1	Exp#2	ATLAS#1	ATLAS#2
Intensity	42.1	22.2	88.6	22.5
Color	51.0	34.3	91.8	24.6
1^{st}-order Descriptor Fields	98.4	97.5	100	39.4
1^{st}- and 2^{nd}-order Descriptor Fields	92.8	97.8	100	33.4
Ours without illumination estimation	100	98.6	100	83.3
Ours with illumination estimation	**100**	**99.7**	**100**	**85.1**

TABLE II
TRACKING SUCCESS RATE (IN %) OF OUR METHOD USING DIFFERENT EDFFS ON ORIGINAL SEQUENCES OF RIGID POSE DATASET

	channels	soda	soup	clown	candy	cube	edge	average
Intensity	1	76	90	94	90	98	94	90.3
Color	3	94	95	96	93	97	93	94.7
Gradient Orientation	1	80	91	95	89	93	91	89.8
Gradient Fields	4	92	92	98	90	96	92	93.3
Lab Gradient Fields	12	90	**98**	98	**96**	92	93	94.5
Color+Gradient Fields	7	**94**	96	**98**	94	**98**	**95**	**95.8**
Depth	1	83	80	83	81	78	78	80.5
Depth+Color	4	95	94	97	95	**99**	**96**	96.0
Depth+Gradient Fields	5	**96**	94	**99**	**96**	98	95	**96.3**

C. Results on Rigid Pose Dataset

The Rigid Pose Dataset [11] provides synthetic video sequences of 6 different objects (as shown in Fig. 3(b)) under a variety of realistic conditions. The provided videos compose of original (noise-free) sequences, noisy sequences and occluded sequences, and are featured with wide-range movements, background and object variability, added noise, and heavy occlusions. Textured 3D object models are provided in this dataset, so we directly use them in our tracking pipeline. Since [11] is a stereo-based method, stereo video pairs (from left and right cameras) are provided. We only use the left video sequences to evaluate our method on original, noisy and occluded conditions.

We first evaluate the performance of different EDFFs. Then, we compare the results of our method with [11] and the other four methods: a state-of-art region-based 3D tracker, PWP3D [22]; a particle-filter-based tracker provided by BLORT [24]; a re-implemented version of the gradient-based direct tracker [8]; and a method based on direct alignment between consecutive video frames, which we refer to as Consecutive method. The Consecutive method can be seen as an approximate implementation of [10], except that the surface normal constraint for intensity variation is not considered (which will not affect the performance in this synthetic dataset). We use the same evaluation criteria as in [11]. We measure the tracking success rate throughout the entire sequence. When tracking is lost, the tracker is automatically reset to the ground truth.

1) Evaluation of Different EDFFs: We first evaluate the performance of different dense image features and their combinations used in our Extended Dense Feature Fields (EDFF). As discussed in Section III-C, different kinds of dense image features can be added to the EDFF and be optimized in a unified Gauss-Newton solver. The complementarity of different features would result in more accurate pose estimate. Here we consider nine different kinds of dense image features and their combinations: (1) Image intensity; (2) Color (RGB values); (3) Gradient Orientation; (4) Gradient Fields (here we refer Gradient Fields to gradient-based Descriptor Fields [8] for simplicity.); (5) Stacked Gradient Fields of Lab color space; (6) Color + Gradient Fields; (7) Depth; (8) Depth + Color; (9) Depth + Gradient Fields. Because we use a synthetic dataset for evaluation, depth information can be acquired by rendering the objects with ground-truth poses. But we clarify that real depth measurements from depth sensors can also be applied to our algorithm (after a simple pre-filtering). We evaluate the above dense features on the original sequences of Rigid Pose Dataset and the results are shown is Table II.

The results suggest that image gradient performs generally better than image intensity, but also show the benefit of using color information. We find in our experiments that color is a discriminative feature and is very helpful in distinguishing the object from cluttered background. The object and cluttered background may share similar gradient around the object silhouette, or they may happen to have similar colors, but these two situations rarely occur at the same time. So it would be complementary to combine color and gradient information together. We simply stack them up in the unified EDFF, and get a better average tracking success rate than using color or gradient alone. To further test the capability of the unified EDFF, we add synthetic depth measurement to it, and obtain superior results compared to EDFFs without depth. This proves that the proposed method can be extended to a RGB-D object tracking method by simply adding depth information to the Extended Dense Feature Fields. Here we only test some of the most commonly used dense image features. Some more complex image features can also be used in the EDFF, as long as they assign some sort of measurements to each pixel.

Note that in contrast to the results on Dense Tracking Dataset, in which gradient performs much better than intensity and color, the results here indicate that the performance of intensity, color and gradient are comparable on Rigid Pose Dataset. Particularly, color even performs slightly better than gradient. The reason is that unlike the videos in Dense Tracking Dataset, the videos in Rigid Pose Dataset do not contain

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

TABLE III
TRACKING SUCCESS RATE (IN %) ON ORIGINAL AND NOISY SEQUENCES OF RIGID POSE DATASET

	soda		soup		clown		candy		cube		edge		average
	orig	noisy	orig	noisy	orig	noisy	orig	noisy	orig	noisy	orig	noisy	
Sparse-and-Dense	**99**	**97**	**98**	**99**	**100**	**98**	**100**	**100**	**100**	**100**	**98**	**98**	**98.9**
PWP3D	84	84	96	96	96	89	84	84	84	74	85	84	86.7
BLORT	76	65	77	66	88	82	77	76	93	94	72	91	79.8
Descriptor Fields	92	85	92	93	98	93	90	88	96	95	92	94	92.3
Consecutive	90	88	95	95	93	92	93	**93**	95	95	95	**95**	93.3
Ours	**94**	**90**	**96**	**96**	**98**	**95**	**94**	91	**98**	**96**	**95**	91	**94.5**

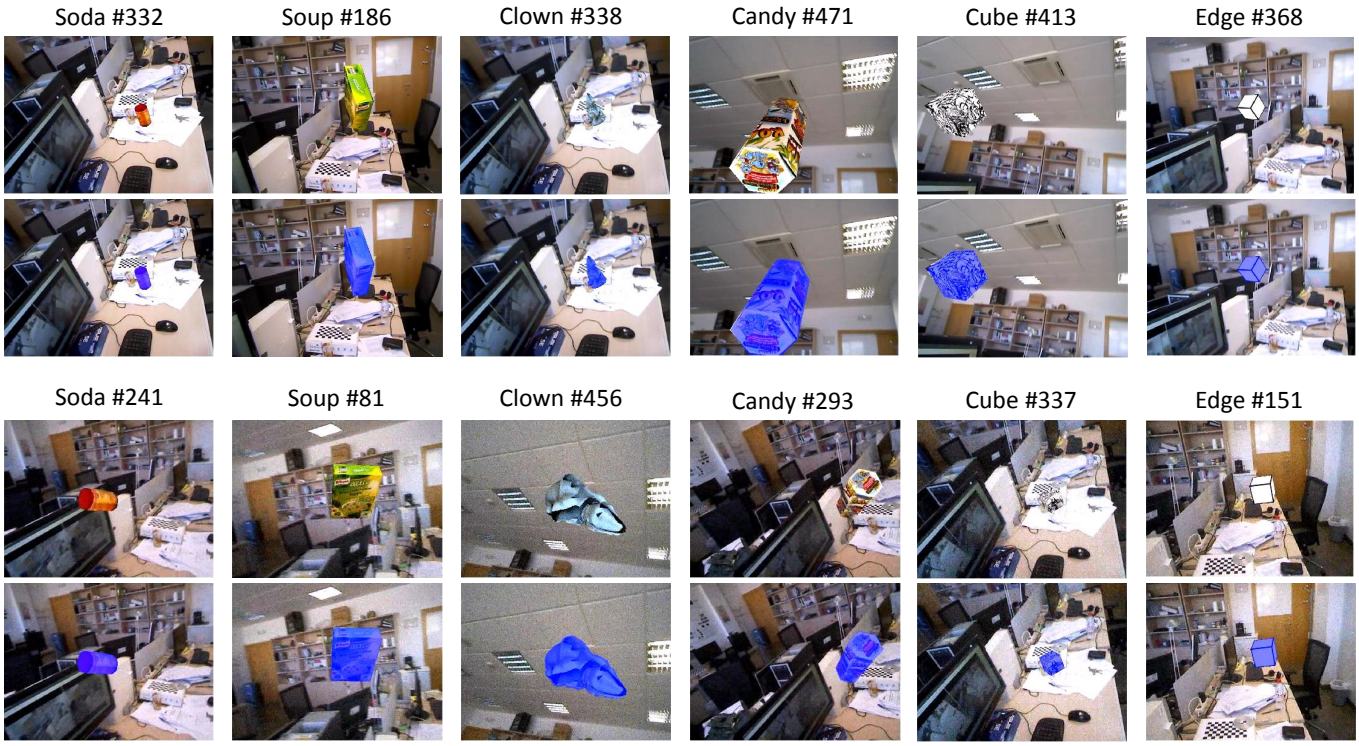


Fig. 8. Sample frames from original and noisy sequences of 6 objects in Rigid Pose Dataset [11] and the corresponding tracking results. First and second row: Sample frames and tracking results of original sequences. Third and fourth row: Sample frames and tracking results of noisy sequences. Tracking results are superimposed on the selected frames. (#number = frame index.) Our method is capable of tracking objects with wide-range rotations and translations, in cluttered background, and with added noise.

complex illumination changes, and the objects are not made of specular materials, so that intensity and color remain stable and are reliable features for pose tracking. This is especially true when tracking extremely poorly textured objects, such as the case of the "edge" sequence, in which intensity and color both perform slightly better than gradient.

Since we mainly focus on monocular-based 3D object tracking in this paper, only color and gradient information are included in our Extended Dense Feature Fields in the following evaluation.

2) Original and Noisy Sequences: We evaluate the overall performance of our method on original and noisy sequences of Rigid Pose Dataset, and compare our results with 5 state-of-art methods. Table III summarizes the evaluation results. Sparse-and-dense method in [11] obtains an almost perfect result, as reported in the first row of Table III. They combine dense motion and stereo cues with sparse keypoint correspondences, together with feeding back information from the model (AR flow). Although our method is monocular-based and only utilizes dense visual cues, it still obtains a competitive result compared to [11]. Our method also outperforms PWP3D [22], BLORT [24], Descriptor Fields [8] and Consecutive method [10]. Consecutive method seems to be less affected by the added noise, because it aligns noisy image to noisy template (i.e., the previous frame). But our method still obtains a superior overall performance compared to Consecutive method, because Consecutive method sometimes suffers from cluttered background interference, as discussed in previous sections. In order to compare the performance of the proposed Extended Dense Feature Fields with Descriptor Fields [8], here we use color+gradient in our Extended Dense Feature Fields (EDFF), while Descriptor Fields method [8] relies only on gradient. The other settings are the same for these two methods. The results show that the proposed EDFF outperforms Descriptor Fields under the same settings. The added color information makes our EDFF a more robust dense feature representation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2017.2731519, IEEE Transactions on Circuits and Systems for Video Technology

TABLE IV
TRACKING SUCCESS RATE (IN %) ON OCCLUDED SEQUENCES OF RIGID POSE DATASET

	soda	soup	clown	candy	cube	edge	average
Sparse-and-Dense	68	80	77	81	76	57	73.2
PWP3D	44	44	44	39	38	39	41.3
BLORT	54	63	76	64	76	68	66.8
Consecutive	66	76	68	81	71	65	71.2
Descriptor Fields without occlusion detection	50	54	48	66	53	40	51.8
Descriptor Fields with occlusion detection	74	84	81	76	80	67	77.0
Ours without occlusion detection	52	60	48	70	55	43	54.7
Ours with occlusion detection	**76**	**87**	**84**	**81**	**81**	**68**	**79.5**

Fig. 9. Sample frames from occluded sequences of 6 objects in Rigid Pose Dataset [11] and the corresponding tracking results. First row: Sample frames. Second row: Tracking results superimposed on the original images. (#number = frame index.) With the aid of an occlusion detection step, our method is able to handle heavily occluded situations.

compared to Descriptor Fields.

Some of the video frames and the corresponding tracking results are illustrated in Fig. 8. The tracking results are superimposed on the video frames to demonstrate the tracking accuracy. The proposed method precisely tracks both well-textured and poor-textured objects, with large scale movement and added noise in the sequences.

3) Sequences with Heavy Occlusions: We evaluate our method on the occluded sequences of Rigid Pose Dataset to demonstrate the effectiveness of the occlusion detection module. The results are summarized in Table IV. The last two rows show the results of our method with and without the occlusion detection step. The tracking success rate is greatly improved with the help of the occlusion detection module. To compare the performance of the proposed Extended Dense Feature Fields with Descriptor Fields under occlusion, we also evaluate the Descriptor Fields method [8] with and without adding our occlusion detection module. In both cases, the proposed EDFF (color+gradient are used) outperforms Descriptor Fields under the same settings, which further demonstrates the effectiveness of combining complementary features in our EDFF.

Moreover, our method even outperforms the stereo-based sparse-and-dense method in [11] which utilizes much richer information. In sparse-and-dense method [11], the key component to deal with occlusion is sparse keypoint detection and matching. Although keypoint-based methods are generally robust to partial occlusions, they still require sufficient texture to extract enough number of local keypoints. This condition is often violated by the heavy occlusion in the videos, as a large proportion of the tracked object is occluded in some frames, leaving only a small region for keypoint extraction. As opposed to sparse (indirect) methods, direct methods exploit dense image information, and are more robust in these heavily occluded situations. We find that direct methods are able to successfully track the object with very small non-occluded image regions. But this advantage would be weakened by the occluded pixels, if they were not correctly detected and discarded. The efficient occlusion detection module described in Section III-E enables our method to robustly detect the occlusion area. Experimental results show that our method outperforms the other 5 state-of-art monocular and stereo-based methods in dealing with partial occlusions. Some of the indicative frames in occluded sequences and the corresponding tracking results are presented in Fig. 9.

4) Evaluation Using Only Half of the Frames: In order to further demonstrate the ability of our method in dealing with large movements, we manually increase motion by dropping every second frame and using only half of the frames (i.e., frame 1,3,5,...) for pose tracking. In this way, the average inter-frame motion is approximately doubled, which makes it more difficult to successfully track the object. The experiments are conducted on original and noisy sequences of Rigid Pose Dataset. We compare our method with the 3 most competitive monocular-based methods: PWP3D [22], Descriptor Fields [8], and Consecutive method [10].

The results are depicted in Table V. Our method outperforms the 3 state-of-art monocular-based methods and the improvements are more significant than that of using all of the video frames (compared to the results in Table III). The evaluation results demonstrate that our method is more

TABLE V
TRACKING SUCCESS RATE (IN %) ON ORIGINAL AND NOISY SEQUENCES OF RIGID POSE DATASET (USING ONLY HALF OF THE FRAMES)

	soda		soup		clown		candy		cube		edge		average
	orig	noisy	orig	noisy	orig	noisy	orig	noisy	orig	noisy	orig	noisy	
PWP3D	72	67	80	77	84	83	78	75	71	71	70	68	74.7
Descriptor Fields	71	70	73	73	82	82	80	77	82	76	86	85	78.1
Consecutive	77	74	85	84	80	80	77	75	78	77	89	82	79.8
Ours	**89**	**87**	**87**	**86**	**88**	**86**	**85**	**83**	**85**	**84**	**89**	**87**	**86.3**

robust to large inter-frame movements than the previous works, owing to the dynamic template creation and updating strategy which increases the convergence rate for pose estimation (as discussed in Section III-B).

D. Discussion

In this subsection, we give a brief discussion about the above experimental results on the two datasets. Since the newly introduced components in our method are specially designed for dealing with challenging situations, the improvements are most significant on challenging sequences with illumination changes, specularity, heavy occlusions and large inter-frame movements.

Firstly, our method greatly outperforms the previous work [8] on ATLAS#2, which is the most difficult video sequence in Dense Tracking Dataset. Despite the illumination changes, specularity and fast movements, our method improves the tracking success rate from 39.4% to 85.1%. For the other 3 videos in Dense Tracking Dataset, our method shows a minor improvement, because they are relatively easier. Secondly, the results on occluded sequences of Rigid Pose Dataset are significantly improved with the help of the proposed occlusion detection module. We improve the tracking success rate of monocular-based methods from 71.2% (Consecutive method [10] in Table IV) to 79.5%. Our method even outperforms the state-of-art stereo-based method [11] with 6.3% improvements. Thirdly, when evaluating using only half of the frames on original and noisy sequences of Rigid Pose Dataset, the improvements of our method are much more remarkable. We improve the tracking success rate from 79.8% (Consecutive method [10] in Table V) to 86.3%, which proves the advantage of our method in dealing with large and fast movements.

To conclude, our method shows outstanding results for the most challenging situations, and also obtains minor improvements for relatively easier cases.

As presented in Section III, there are 4 newly introduced components in our tracking pipeline: Dynamic Template Rendering, Extended Dense Feature Fields (EDFF), Illumination Estimation, and Occlusion Detection. Each component contributes in dealing with one or more challenging factors. The illumination estimation module is designed for dealing with illumination changes, and the occlusion detection module for dealing with partial occlusions. The dynamic template rendering strategy helps to deal with both large movements and partial occlusions, while the proposed EDFF is a robust dense feature representation which would be helpful for all of the challenging factors. The proposed method obtains good performance in two challenging datasets thanks to these newly introduced components, and the most important contributors are different according to the main challenging factors of different video sequences. Moreover, the 4 components are closely related to each other and they work together to achieve the best performance. For example, without dynamic template rendering, the occlusion detection step is not possible. Specifically speaking, the dynamic template rendering strategy and the illumination estimation module play the main role on Dense Tracking Dataset, in which the main challenging factors are illumination changes and fast motions. For the occluded sequences of Rigid Pose Dataset, our occlusion detection module plays the main role. For original and noisy sequences of Rigid Pose Dataset, dynamic template rendering and EDFF are the main contributors.

E. Processing Time

In this subsection, we discuss about the computational complexity of the proposed algorithm, and present detailed processing times for each step. For all of the experiments a commodity desktop computer with Intel i7 quad core CPU @4.0 GHz and NVIDIA GeForce GTX970 GPU are used. In our implementation, GPU is only used for rendering purposes, and all the other computations are performed on the CPU. We run the proposed dynamic template-based method and our implementation of the discrete template-based method [8] under the same settings. Both methods need a pre-computation step, in which we need to build a textured 3D object model and [8] needs to build a discrete template set. The time for pre-computation is comparable for the two methods. For the subsequent tracking pipeline, the average run time is summarized in Table VI.

Compared to the discrete template-based method [8], our method requires relatively more computation. The extra computational complexity mainly comes from the dynamic template rendering, dense feature calculation, pose optimization, illumination estimation and occlusion detection steps. When using discrete templates, 3D model rendering is only performed once per frame. But in our method, the template is dynamically updated after each pose optimization iteration. So we need to re-render the 3D model per iteration with the updated pose parameters, which results in more computation for the rendering engine. Thanks to the modern graphic hardware, 3D model rendering can be processed very fast. A 3D model with moderate complexity (such as the models in Rigid Pose Dataset) can be rendered by our graphic card in 1$\sim$2 ms. So the added computation from model rendering is acceptable. The computational complexity of dense feature calculation and pose optimization is $O(d \times N)$, where d is the number of feature channels and N is the number of pixels. We use 7 channels (color+gradient fields) in our EDFF for

TABLE VI
PROCESSING TIMES PER FRAME (IN MS)

	Processing Step	Discrete Templates [8]	Dynamic Templates (proposed)
A (per frame)	Template Selection	31.3	-
	Rendering	1.7	1.7×2
	Illumination Estimation	-	6.1
	Occlusion Detection	-	3.2
	Dense Feature Calculation	2.4	3.5
B (per iteration)	Pose Optimization	3.3	5.7
	Dynamic Rendering	-	1.7
	Template Dense Feature Re-Calculation	-	1.6
Average Number of Iterations n_{iter}		17.1	12.6
Total Time per Frame $A + B \times n_{iter}$		91.8	129.6

best tracking performance, while [8] only uses 4 channels (gradient fields only). Moreover, we need to re-calculate the dense features for the updated template after each iteration. So there is also a moderate increase of run time for our method in these two steps. The extra run time for illumination estimation and occlusion detection steps are also acceptable because they only need to be performed once per frame.

Although our method requires more computation in the above steps, it has some advantages over [8]. Firstly, using dynamic templates leads to faster convergence in pose optimization, which reduces the average number of iterations. Secondly, our tracking pipeline does not include a template selection step, which is required in [8]. The computational complexity of template selection is closely related to the number of discrete templates (10 in our evaluation). Overall, our method requires approximately 40% more time to process a single frame compared to [8], with the benefit of more robust tracking performance.

V. CONCLUSIONS

We have presented an efficient direct 3D object tracking method based on textured model rendering. With the newly introduced rendering technique, dynamic templates are created and updated online to register with video frames, which results in much more robust tracking performance compared to previous approaches. We have also integrated an illumination estimation module and an occlusion detection module into the tracking pipeline, which makes our method capable of handling lighting variation and heavy occlusion. Moreover, we have generalized the representation of dense image features for direct image alignment, and obtained improved results by combining complementary features. With only monocular information, our method achieved competitive or even superior results compared to the state-of-art stereo-based method, and also outperformed the other state-of-art monocular-based methods on two challenging datasets.

This paper mainly focuses on the 6-dof pose tracking of rigid objects, but some of the components in our tracking pipeline are quite general, such as the illumination estimation and occlusion detection modules. Besides, 3D textured model rendering and Extended Dense Feature Fields are also very generic techniques. Therefore, these components could potentially be used to deal with other related problems, such as non-rigid object tracking.

ACKNOWLEDGMENT

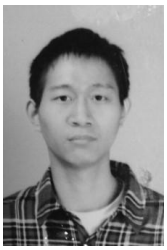
This work is partly supported by the National Natural Science Foundation of China under Grant Nos. 61132007 and U1533132.

REFERENCES

[1] V. Lepetit and P. Fua, *Monocular model-based 3D tracking of rigid objects*. Now Publishers Inc, 2005.
[2] I. Gordon and D. G. Lowe, "What and where: 3d object recognition with accurate pose," in *Toward category-level object recognition*. Springer, 2006, pp. 67–82.
[3] Q. Wang, W. Zhang, X. Tang, and H.-Y. Shum, "Real-time bayesian 3-d pose tracking," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 12, pp. 1533–1541, 2006.
[4] K. Pauwels, L. Rubio, and E. Ros, "Real-time pose detection and tracking of hundreds of objects," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 12, pp. 2200–2214, 2016.
[5] C. Choi and H. I. Christensen, "Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2010, pp. 4048–4055.
[6] J. P. Lima, F. Simões, L. Figueiredo, and J. Kelner, "Model based markerless 3d tracking applied to augmented reality," *Journal on 3D Interactive Systems*, vol. 1, 2010.
[7] Y. Park, V. Lepetit, and W. Woo, "Multiple 3d object tracking for augmented reality," in *Proceedings of the IEEE/ACM International Symposium on Mixed and Augmented Reality*, 2008, pp. 117–120.
[8] A. Crivellaro and V. Lepetit, "Robust 3d tracking with descriptor fields," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3414–3421.
[9] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A dataset and evaluation methodology for template-based tracking algorithms," in *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality*, 2009, pp. 145–151.
[10] B.-K. Seo and H. Wuest, "A direct method for robust model-based 3d object tracking from a monocular rgb image," in *Proceedings of the European Conference on Computer Vision Workshop*, 2016, pp. 551–562.
[11] K. Pauwels, L. Rubio, J. Diaz, and E. Ros, "Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2347–2354.
[12] Z.-W. Chen, C.-C. Chiang, and Z.-T. Hsieh, "Extending 3d lucas–kanade tracking with adaptive templates for head pose estimation," *Machine Vision and Applications*, vol. 21, no. 6, pp. 889–903, 2010.
[13] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *arXiv preprint arXiv:1607.02565*, 2016.
[14] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, no. 3, pp. 221–255, 2004.
[15] H. S. Hong and M. J. Chung, "3d pose and camera parameter tracking algorithm based on lucas-kanade image alignment algorithm," in *Proceedings of the International Conference on Control, Automation and Systems*, 2007, pp. 548–551.

[16] S. Benhimane and E. Malis, "Real-time image-based tracking of planes using efficient second-order minimization," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2004, pp. 943–948.
[17] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Proceedings of the European Conference on Computer Vision*, 2014, pp. 834–849.
[18] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision." in *Proceedings of the International Joint Conference on Artificial Intelligence*, vol. 81, no. 1, 1981, pp. 674–679.
[19] A. Crivellaro, P. Fua, and V. Lepetit, "Dense methods for image alignment with an application to 3d tracking," Tech. Rep., 2014.
[20] G. Caron, A. Dame, and E. Marchand, "Direct model based visual tracking and pose estimation using mutual information," *Image and Vision Computing*, vol. 32, no. 1, pp. 54–63, 2014.
[21] R. Basri and D. W. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.
[22] V. A. Prisacariu and I. D. Reid, "Pwp3d: Real-time segmentation and tracking of 3d objects," *International Journal of Computer Vision*, vol. 98, no. 3, pp. 335–354, 2012.
[23] C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert, "Region-based pose tracking with occlusions using 3d models," *Machine Vision and Applications*, vol. 23, no. 3, pp. 557–577, 2012.
[24] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze, "Blort-the blocks world robotic vision toolbox," in *IEEE International Conference on Robotics and Automation*, 2010.
[25] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes," in *Proceedings of the International Conference on Computer Vision*, 2011, pp. 858–865.
[26] M. Lourakis and X. Zabulis, "Model-based pose estimation for rigid objects," in *Proceedings of the International Conference on Computer Vision Systems*, 2013, pp. 83–92.
[27] Blender Foundation, "Blender," https://www.blender.org/, 2014.
[28] Inivis, "AC3D," http://www.inivis.com/, 2016.
[29] Autodesk, "Autodesk Remake," http://remake.autodesk.com, 2015.
[30] ——, "123D Catch," http://www.123dapp.com/catch/, 2014.
[31] "Openscenegraph," http://www.openscenegraph.org.
[32] C. Wu, M. Zollhöfer, M. Nießner, M. Stamminger, S. Izadi, and C. Theobalt, "Real-time shading-based refinement for consumer depth cameras," *ACM Transactions on Graphics*, vol. 33, p. 3, 2014.
[33] R. Or-El, G. Rosman, A. Wetzler, R. Kimmel, and A. M. Bruckstein, "Rgbd-fusion: Real-time high precision depth recovery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5407–5416.

Leisheng Zhong received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China in 2013. He is currently pursuing the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China. His research interests include 3D computer vision, vision for robotics and machine learning.



Ming Lu is currently a Ph.D. candidate in Tsinghua University. He is supervised by Prof. Li Zhang. His research interests include computer vision and computer graphics. Ming is developing the 3d face capture system for Intel Labs China which is widely used in face edit, face beautification, 3d avatar etc. He is also working on computer vision tasks like optical flow, object detection, semantic segmentation, neural style transfer.

Li Zhang received the B.S., M.S., and Ph.D. degrees in signal and information processing from Tsinghua University, Beijing, China, in 1987, 1992, and 2008, respectively. In 1992, he joined the faculty of the Department of Electronic Engineering at Tsinghua University, Beijing, China, where he is currently a professor. His research interests include image processing, computer vision, pattern recognition, and computer graphics.