SRT3D: A Sparse Region-Based 3D Object Tracking Approach for the Real World

Manuel Stoiber · Martin Pfanne · Klaus H. Strobl · Rudolph Triebel · Alin Albu-Schäffer

Abstract Region-based methods have become increasingly popular for model-based, monocular 3D tracking of texture-less objects in cluttered scenes. However, while they achieve state-of-the-art results, most methods are computationally expensive, requiring significant resources to run in real-time. In the following, we build on our previous work and develop *SRT3D*, a sparse region-based approach to 3D object tracking that bridges this gap in efficiency. Our method considers image information sparsely along so-called correspondence lines that model the probability of the object's contour location. We thereby improve on the current state of the art and introduce smoothed step functions that consider a defined global and local uncertainty. For the resulting probabilistic formulation, a thorough analysis is provided. Finally, we use a pre-rendered sparse viewpoint model to create a joint posterior probability for the object pose. The function is maximized using second-order Newton optimization with Tikhonov regularization. During the pose estimation, we differ-

entiate between global and local optimization, using a novel approximation for the first-order derivative employed in the Newton method. In multiple experiments, we demonstrate that the resulting algorithm improves the current state of the art both in terms of runtime and quality, performing particularly well for noisy and cluttered images encountered in the real world.

Keywords Region-based · 3D object tracking · Pose estimation · Sparse · Real-time

M. Stoiber
German Aerospace Center, 82234 Wessling, Germany
Technical University of Munich, 80333 Munich, Germany
E-mail: manuel.stoiber@dlr.de

M. Pfanne
German Aerospace Center, 82234 Wessling, Germany
E-mail: martin.pfanne@dlr.de

K. H. Strobl
German Aerospace Center, 82234 Wessling, Germany
E-mail: klaus.strobl@dlr.de

R. Triebel
German Aerospace Center, 82234 Wessling, Germany
Technical University of Munich, 80333 Munich, Germany
E-mail: rudolph.triebel@dlr.de

A. Albu-Schäffer
German Aerospace Center, 82234 Wessling, Germany
Technical University of Munich, 80333 Munich, Germany
E-mail: alin.albu-schaeffer@dlr.de

1 Introduction

Tracking a rigid object in 3D space and predicting its six degrees of freedom (6DoF) pose is an essential task in computer vision. Its application ranges from augmented reality, where the location of objects is needed to superimpose digital information, to robotics, where the object pose is required for robust manipulation in unstructured environments. Given consecutive image frames, the goal of 3D object tracking is to estimate both the rotation and translation of a known object relative to the camera. In contrast to object detection, tracking continuously provides information, which, for example, allows robots to react to unpredicted changes in the environment using visual servoing. While the problem has been thoroughly studied, many challenges such as partial occlusions, appearance changes, motion blur, background clutter, and real-time requirements still exist. In this section, we first provide an overview of common techniques. This is followed by a survey of related work on region-based methods. Finally, we introduce our approach and summarize the contributions to the current state of the art.

1.1 3D Object Tracking

In the past, many approaches to 3D object tracking have been proposed. Based on surveys (Lepetit and Fua, 2005; Yilmaz et al., 2006), as well as on recent developments, techniques can be differentiated by their use of key-points, explicit edges, direct optimization, deep learning, depth information, and image regions. Key-point features such as *SIFT* (Lowe, 2004), *ORB* (Rublee et al., 2011), or *BRISK* (Leutenegger et al., 2011) have been widely used for 3D object tracking (Wagner et al., 2010; Vacchetti et al., 2004), with more recent developments like *LIFT* (Yi et al., 2016) and *SuperGlue* (Sarlin et al., 2020) introducing deep learning at various stages. Explicit edges provide an additional source of information that is used by many approaches (Huang et al., 2020; Bugaev et al., 2018; Seo et al., 2014; Comport et al., 2006; Drummond and Cipolla, 2002; Harris and Stennett, 1990). Also, direct methods (Engel et al., 2018; Seo and Wuest, 2016; Crivellaro and Lepetit, 2014), which optimize a photometric error and can be traced back to Lucas and Kanade (1981), have been proposed. While all three classes of techniques have valid applications, unfortunately, they also have significant drawbacks. First, approaches based on key-points and direct optimization require rich texture, limiting the range of suitable objects. At the same time, edge-based methods, which perform better for low-textured objects, often fail in cluttered scenes. Finally, motion blur changes the appearance of both texture and edges, leading to additional problems.

To overcome these issues and train the algorithm on data, recently, deep-learning-based approaches that use convolutional neural networks (CNNs) to consider full image information have been proposed. While they achieve good results, only few approaches (Wen et al., 2020) run in real-time, with most methods (Deng et al., 2021; Wang et al., 2019; Li et al., 2018; Xiang et al., 2018; Garon and Lalonde, 2017) reporting less than 30 frames per second. However, even the most efficient algorithms require significant resources from high-end GPUs. In addition, typical disadvantages include time-consuming training and the requirement for a textured 3D model. Another relatively new development is the availability of affordable depth cameras that measure the surface distance for each pixel. While purely depth-based object tracking is possible, most methods (Ren et al., 2017; Kehl et al., 2017; Tan et al., 2017; Krull et al., 2015; Krainin et al., 2011) combine information from both depth and RGB cameras. In general, this leads to superior results. Unfortunately, in many applications, using an additional depth sensor is not an option. Also, note that algorithms require images with high quality. Depending on hardware, surface distances, surface characteristics, and lighting conditions, such images are hard to obtain.

Because of the discussed shortcomings, region-based techniques (Stoiber et al., 2020; Zhong et al., 2020b; Tjaden et al., 2018; Prisacariu and Reid, 2012) have become increasingly popular. The big advantage of such methods is that they are able to reliably track a wide variety of objects in cluttered scenes, using only a monocular RGB camera and a texture-less 3D model of the object. The main assumption is thereby that objects are distinguishable from the background. As a consequence, no object texture is needed. While past approaches were computationally expensive, our sparse formulation overcomes this disadvantage. Finally, based on our experience, region-based methods are robust to motion blur, making it possible to track fast-moving objects. Because of these excellent properties, the following work focuses on region-based techniques.

1.2 Related Work

Region-based methods use image statistics to differentiate between a foreground region that corresponds to the object and a background region. Typically, color statistics are used to model the membership of each pixel. Based on the two regions, the goal is to find the object pose and corresponding silhouette that best explains the segmentation of the image. The great potential of this technique was already demonstrated by early approaches that allowed robust tracking in many challenging scenarios (Schmaltz et al., 2012; Brox et al., 2010; Rosenhahn et al., 2007). Segmentation and pose tracking were thereby treated as independent problems, with an initial step to extract the contour and a subsequent optimization to find the pose. Dambreville et al. (2008) later combined the two processes in a single energy function, leading to improved tracking robustness. Building on this approach and including the pixel-wise posterior membership of Bibby and Reid (2008), Prisacariu and Reid (2012) developed *PWP3D*, a real-time-capable algorithm that uses a level-set pose embedding. It is the foundation of almost all state-of-the-art region-based methods.

Based on *PWP3D*, multiple algorithms were suggested that incorporate additional information, extend the segmentation model, or improve efficiency. For the combination of both depth- and region-based information, Kehl et al. (2017) extended the energy function of *PWP3D* with a term that is based on the *Iterative Closest Point (ICP)* algorithm. In a different approach, Ren et al. (2017) tightly coupled region and depth information in a probabilistic formulation that uses 3D
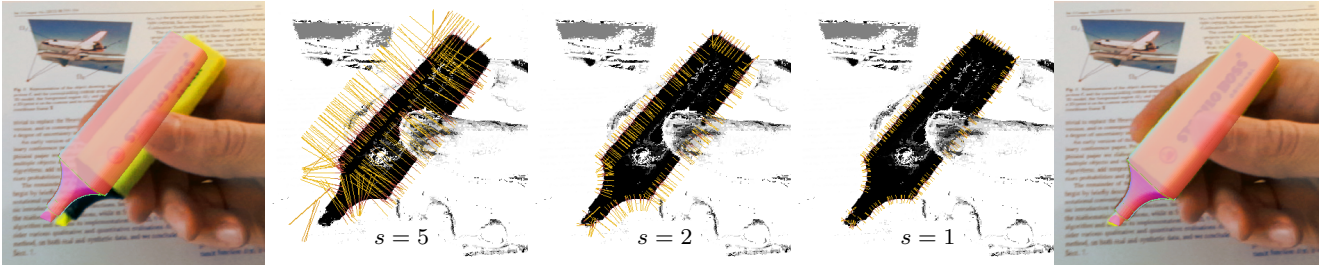
Fig. 1 Tracking of a marker pen in the real world. The image on the left shows a rendered overlay of the object model for the initial pose. The estimated pose after the optimization is visualized in the image on the right. The three illustrations in the middle show yellow correspondence lines for different scales s. High probabilities for the contour location are illustrated in red. Pixel-wise posterior probabilities that describe the probability that a pixel belongs to the background are encoded in grayscale images. Note that during tracking, pixel-wise posteriors are only calculated along correspondence lines.

signed distance functions. Recently, object texture was considered using direct optimization of pixel intensity values (Liu et al., 2020; Zhong and Zhang, 2019) or descriptor fields (Liu et al., 2021). Also, a combination with an edge-based technique that uses a contour-part model was introduced by Sun et al. (2021). Later, Li et al. (2021) developed adaptively weighted local bundles that combine region and edge information. To improve occlusion handling, Zhong et al. (2020a) suggested the use of learning-based object segmentation. Finally, the incorporation of measurements from a mobile phone's inertial sensor was suggested by Prisacariu et al. (2015).

To improve segmentation, Zhao et al. (2014) extended the appearance model of *PWP3D* with a boundary term that considers spatial distribution regularities of pixels. Later, Hexner and Hagege (2016) proposed the use of local appearance models that were inspired by the localized contours of Lankton and Tannenbaum (2008). The idea was further improved by Tjaden et al. (2018) with the development of temporally consistent local color histograms. Finally, Zhong et al. (2020b) proposed a method that introduces polar-based region partitioning and edge-based occlusion detection.

For better efficiency, Zhao et al. (2014) suggested a particle-filter-like stochastic optimization that initializes a subsequent damped Newton method. Later, a hierarchical rendering approach that uses the Levenberg-Marquardt algorithm was developed by Prisacariu et al. (2015). Also, Tjaden et al. (2018) proposed the use of a Gauss-Newton method to improve convergence. In addition to optimization, another idea towards better efficiency is the use of simplified signed distance functions (Liu et al., 2020). A different approach by Kehl et al. (2017) suggested the use of precomputed contour points to represent the object's 3D geometry and calculate the energy function sparsely along rays. Finally, in our previous work (Stoiber et al., 2020), we improved

on this idea and developed a sparse approach that is based on correspondence lines, making our algorithm significantly more efficient than the previous state of the art while achieving better tracking results.

1.3 Contribution

Starting from the ideas presented in the previous section, we focus on the development of *SRT3D*, a highly efficient, sparse approach to region-based tracking. To keep complexity at a minimum, we only use region information and, like *PWP3D*, adopt a global segmentation model. For our formulation, we build on our previous method and consider image information sparsely along correspondence lines. Also, Newton optimization with Tikhonov regularization is used to estimate the object pose. An illustration of the tracking process with converging correspondence lines at different scales is given in Fig. 1. While the formulation is similar to our previous method (Stoiber et al., 2020), our main motivation is to advance the approach and the current state of the art using improved uncertainty modeling and better optimization techniques. In addition, we provide a more formal derivation and analysis of the highly efficient correspondence line model. In detail, the main contributions of this work are as follows:

- A formal definition of correspondence lines and a thorough mathematical derivation of the probabilistic model that describes the contour location.
- Novel smoothed step functions that allow the modeling of both local and global uncertainty.
- A detailed theoretical analysis that shows how different parameter settings affect the characteristics of posterior probability distributions.
- Global and local optimization strategies and a new approximation for the local first-order derivative.

In the remainder, we first provide a detailed derivation of the correspondence line model. This is followed
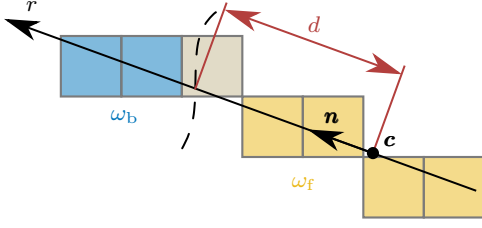
Fig. 2 Correspondence line defined by a center $\boldsymbol{c}$ and a normal vector $\boldsymbol{n}$. The illustration shows pixels along the correspondence line as well as the foreground region ω_{f} in yellow and the background region ω_{b} in blue. The contour distance d points from the correspondence line center to an estimated contour, indicated by a dashed line.

by the development of a 3D tracking approach that combines the correspondence line model with a sparse representation of the 3D object geometry. Subsequently, implementation details for the resulting algorithm are discussed. Finally, we conduct a thorough evaluation on the *RBOT* and the *OPT* dataset, showing that our approach outperforms the current state of the art by a considerable margin in terms of efficiency and quality.

2 Correspondence Line Model

In this section, we first provide a formal mathematical definition of correspondence lines. This is followed by a probabilistic model that considers the segmentation of a correspondence line into foreground and background. To improve computational efficiency, we extend this model and provide a discrete scale-space formulation. Finally, we introduce novel smoothed step functions and discuss how their configuration affects the contour location's posterior probability.

2.1 Correspondence Lines

In contrast to most state-of-the-art algorithms, we do not consider image information densely over the entire image. Instead, inspired by *RAPID* (Harris and Stennett, 1990), pixel values are processed sparsely along correspondence lines (Stoiber et al., 2020). The name correspondence line is motivated by the term *correspondence point* used in *ICP* (Besl and McKay, 1992). Similar to *ICP*, correspondences are first defined and the optimization with respect to them is then conducted in a second step. While for *ICP*, individual 3D points are used as data, multiple pixel values along a line are considered in this case. A visualization of a single correspondence line is shown in Fig. 2.

Starting from our earlier work (Stoiber et al., 2020) and inspired by the commonly used definition of im-

ages as $\boldsymbol{I} \colon \boldsymbol{\Omega} \to \{0, \ldots, 255\}^3$, we formally denote a correspondence line as a map $\boldsymbol{l} \colon \omega \to \{0, \ldots, 255\}^3$. In this notation $\boldsymbol{\Omega} \subset \mathbb{R}^2$ describes the image domain while $\omega \subset \mathbb{R}$ is considered the correspondence line domain. Image values $\boldsymbol{y}$, which are typically accessed using the image coordinate $\boldsymbol{x} = \begin{bmatrix} x & y \end{bmatrix}^\top$ and the image function $\boldsymbol{y} = \boldsymbol{I}(\boldsymbol{x})$, are described using the line coordinate r and the correspondence line function $\boldsymbol{y} = \boldsymbol{l}(r)$. Correspondence lines are located in the image and remain fixed once they have been established. The location and orientation of each correspondence line is defined by a center $\boldsymbol{c} = \begin{bmatrix} c_x & c_y \end{bmatrix}^\top \in \mathbb{R}^2$ in image coordinates and a normal vector $\boldsymbol{n} = \begin{bmatrix} n_x & n_y \end{bmatrix}^\top \in \mathbb{R}^2$, with $\|\boldsymbol{n}\|_2 = 1$. Using this definition, the relation between an image $\boldsymbol{I}$ and a correspondence line $\boldsymbol{l}$ is expressed as follows

$$\boldsymbol{l}(r) = \boldsymbol{I}(\boldsymbol{c} + r\boldsymbol{n}), \tag{1}$$

where image coordinates in $\boldsymbol{I}$ are rounded to the center of the next closest pixel.

2.2 Probabilistic Model

Inspired by the generative model of Bibby and Reid (2008), we derive a probabilistic model for the segmentation of a correspondence line into a foreground region ω_{f} and a background region ω_{b}. Note that this is the 1D equivalent of the segmentation of a 2D image into the regions Ω_{f} and Ω_{b}. We assume that there is only one transition between foreground and background. The location of this transition relative to the line center $\boldsymbol{c}$ is described by the contour distance $d \in \mathbb{R}$. A visualization of the contour distance is shown in Fig. 2.

To derive the probabilistic model, we first describe the formation process for a single pixel on the correspondence line. The joint probability distribution writes thereby as follows

$$p(r, \boldsymbol{y}, d, m) = p(r \mid d, m)p(\boldsymbol{y} \mid m)p(m)p(d), \tag{2}$$

where $m \in \{m_{\mathrm{f}}, m_{\mathrm{b}}\}$ is the model parameter that can denote either foreground or background. If we condition this distribution on the image value $\boldsymbol{y}$, we obtain

$$p(r, d, m \mid \boldsymbol{y}) = p(r \mid d, m)p(m \mid \boldsymbol{y})p(d). \tag{3}$$

Following Bibby and Reid (2008), we use Bayes' theorem and the marginalization over m to calculate the pixel-wise posterior probability

$$p(m_i \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid m_i)p(m_i)}{\sum_{j \in \{\mathrm{f,b}\}} p(\boldsymbol{y} \mid m_j)p(m_j)}, \quad i \in \{\mathrm{f,b}\}, \tag{4}$$

where $p(\boldsymbol{y} \mid m_{\mathrm{f}})$ and $p(\boldsymbol{y} \mid m_{\mathrm{b}})$ are probability distributions that describe how likely it is that a specific color

value is part of the foreground region or the background region, respectively. The two distributions can be estimated by calculating two color histograms, one over the foreground region and one over the background region. A detailed explanation of their computation is given in Sect. 4.2. Using the knowledge that foreground and background are equally likely along the correspondence line, i.e. $p(m_\mathrm{f}) = p(m_\mathrm{b})$, we obtain

$$p(m_i \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid m_i)}{p(\boldsymbol{y} \mid m_\mathrm{f}) + p(\boldsymbol{y} \mid m_\mathrm{b})}, \quad i \in \{\mathrm{f}, \mathrm{b}\}. \tag{5}$$

Finally, based on Eq. (3), we are able to marginalize over m and condition on r to express the posterior probability for the contour distance d as

$$p(d \mid r, \boldsymbol{y}) = \frac{1}{p(r)} \sum_{i \in \{\mathrm{f}, \mathrm{b}\}} p(r \mid d, m_i) p(m_i \mid \boldsymbol{y}) p(d). \tag{6}$$

To calculate the posterior probability over the entire correspondence line domain ω, we assume pixel-wise independence and, based on Eq. (6), write

$$p(d \mid \omega, \boldsymbol{l}) \propto \prod_{r \in \omega} \sum_{i \in \{\mathrm{f}, \mathrm{b}\}} p(r \mid d, m_i) p(m_i \mid \boldsymbol{l}(r)). \tag{7}$$

Note that $p(r)$ and $p(d)$ are considered to be uniform and constant and are thus dropped. Also, while pixel-wise independence does not hold in general, it is a well-established approximation that allows us to avoid ill-defined assumptions for spatial regularities and is close enough to reality to yield good results. The conditional line coordinate probability $p(r \mid d, m)$ will be discussed in Sect. 2.4. Similar to the probabilistic model of Bibby and Reid (2008), which describes the probability of a shape kernel given information from an image, Eq. (7) provides the probability of the contour distance d given data from a correspondence line.

2.3 Discrete Scale-Space Formulation

Estimating the distribution of posterior probabilities is computationally expensive since, for each distance d, the product in Eq. (7) has to be computed over the entire domain ω. This results in quadratic complexity for the calculation of the entire distribution. In contrast, pixel-wise posterior probabilities $p(m \mid \boldsymbol{y})$ are used in the posterior probability calculation of multiple distances d, leading to linear complexity. Shifting computation from the calculation of the distribution to the calculation of pixel-wise posterior probabilities thus allows us to improve computational efficiency. Also, it is advantageous to normalize correspondence lines in a way that ensures that a line coordinate pointing to a segment center for one correspondence line points to a
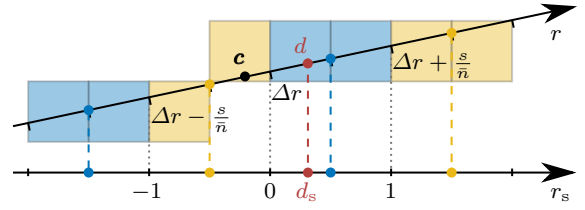

Fig. 3 Example of the relation between the unscaled space r along the correspondence line and the scale-space r_s. Neighboring pixels that are combined into segments are visualized by the same color in blue or yellow. Blue and yellow dots indicate the center of each segment and the corresponding discretized value in the scale-space. An example of the contour distance is illustrated in red. The offset $\varDelta r$ is chosen in a way that ensures that discretized values in the scale-space are the same for all correspondence lines. In this example, $\varDelta r$ points to the closest edge between pixels.

segment center for all correspondence lines. This uniformity can be used in the precalculation of smoothed step function values to further improve efficiency.

In the following, we thus adopt the discrete scale-space formulation from our previous method (Stoiber et al., 2020) to combine multiple pixels into segments. In addition, the formulation projects from the continuous space along the correspondence line into a discrete space that is independent of a correspondence line's location and orientation. An illustration of this transformation is shown in Fig. 3. Both line coordinates and contour distances are projected as follows

$$r_\mathrm{s} = (r - \varDelta r) \frac{\bar{n}}{s}, \tag{8}$$

$$d_\mathrm{s} = (d - \varDelta r) \frac{\bar{n}}{s}, \tag{9}$$

with $s \in \mathbb{N}^+$ the scale that describes the number of pixels combined into a segment, $\bar{n} = \max(|n_x|, |n_y|)$ the major absolute normal component that projects a correspondence line to the closest horizontal or vertical image coordinate, and $\varDelta r \in \mathbb{R}$ the offset from the correspondence line center $\boldsymbol{c}$ to a defined pixel location.

Based on Eq. (7), the posterior probability in the discrete scale-space is calculated as

$$p(d_\mathrm{s} \mid \omega_\mathrm{s}, \boldsymbol{l}_\mathrm{s}) \propto \prod_{r_\mathrm{s} \in \omega_\mathrm{s}} \sum_{i \in \{\mathrm{f}, \mathrm{b}\}} p(r_\mathrm{s} \mid d_\mathrm{s}, m_i) p(m_i \mid \boldsymbol{l}_\mathrm{s}(r_\mathrm{s})), \tag{10}$$

where ω_s is the scaled correspondence line domain and $\boldsymbol{s} = \boldsymbol{l}_\mathrm{s}(r_\mathrm{s})$ a set-valued function that maps from the scaled line coordinate r_s to the segment $\boldsymbol{s}$, which is a set of the closest s pixel values $\boldsymbol{y}$. Similar to pixel-wise posteriors in Eq. (5) and assuming pixel-wise indepen-

dence, segment-wise posteriors are defined as

$$p(m_i \mid \boldsymbol{s}) = \frac{\prod\limits_{\boldsymbol{y} \in \boldsymbol{s}} p(\boldsymbol{y} \mid m_i)}{\prod\limits_{\boldsymbol{y} \in \boldsymbol{s}} p(\boldsymbol{y} \mid m_{\mathrm{f}}) + \prod\limits_{\boldsymbol{y} \in \boldsymbol{s}} p(\boldsymbol{y} \mid m_{\mathrm{b}})}, \quad i \in \{\mathrm{f}, \mathrm{b}\}.$$
(11)

The derived formulation allows to efficiently cover the correspondence line domain ω, using the scale parameter s to set the segment size and to adjust between accuracy and efficiency. In the following, we will again drop the index s for all variables to simplify the notation. Note, however, that all definitions and derivations are valid both for the original space and for the discrete scale-space formulation.

2.4 Smoothed Step Functions

To model the conditional probabilities of the line coordinate $p(r \mid d, m_{\mathrm{f}})$ and $p(r \mid d, m_{\mathrm{b}})$, different smoothed step functions h_{f} and h_{b} have been used. While most state-of-the-art algorithms (Zhong et al., 2020b; Tjaden et al., 2018) use a function based on the arctangent, we previously proved that a hyperbolic tangent results in a Gaussian distribution for the posterior probability $p(d \mid \omega, \boldsymbol{l})$ (Stoiber et al., 2020). In both functions, the smoothed slope is used to model a local uncertainty with respect to the exact location of the foreground and background transition. Considering the plots of the two models in Fig. 4, one notices that the functions quickly converge towards either zero or one for increasing absolute values of $x = r - d$. Except for a small area around zero, both models thus assume that, given the model m and the contour distance d, one knows perfectly on which side of the contour the line coordinate r lies. In the following, we will argue that for real-world applications, this assumption is wrong.

While the pixel-wise posterior probability in Eq. (5) provides very good predictions for the model m, it is still an imperfect simplification of the real world. Typical effects that are not considered by the statistical model are image noise or fast appearance changes that can lead to pixel colors that are not yet present in the color histograms. Another effect originates from pixels that are wrongly classified due to imperfect segmentation and that are then assigned to the wrong color histograms. Finally, there also remains the question if a statistical model that purely relies on pixel colors is sufficient to capture all the statistical effects in the real world and is able to perfectly predict the model m.

To take those limitations into account and consider a constant, global uncertainty in the foreground and background model, we extend the formulation from our
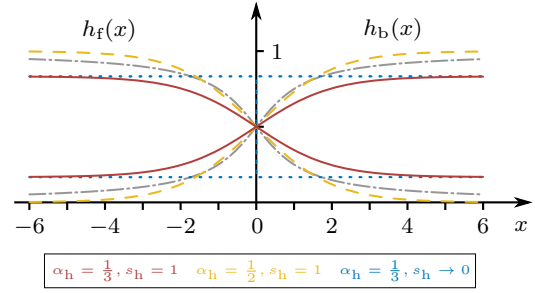

Fig. 4 Smoothed step functions h_{f} and h_{b} that model the conditional line coordinate probabilities $p(r \mid d, m_{\mathrm{f}})$ and $p(r \mid d, m_{\mathrm{b}})$. The functions $h_{\mathrm{f}}(x) = \frac{1}{2} - \frac{1}{\pi} \tan^{-1}\left(\frac{x}{s_{\mathrm{h}}}\right)$ and $h_{\mathrm{b}}(x) = \frac{1}{2} + \frac{1}{\pi} \tan^{-1}\left(\frac{x}{s_{\mathrm{h}}}\right)$ used by Zhong et al. (2020b) and Tjaden et al. (2018) are illustrated by dash-dotted gray lines. The definitions $h_{\mathrm{f}}(x) = \frac{1}{2} - \frac{1}{2} \tanh\left(\frac{x}{2s_{\mathrm{h}}}\right)$ and $h_{\mathrm{b}}(x) = \frac{1}{2} + \frac{1}{2} \tanh\left(\frac{x}{2s_{\mathrm{h}}}\right)$ from our previous work (Stoiber et al., 2020) are shown as dashed yellow lines. In both plots, the slope parameter $s_{\mathrm{h}} = 1$ is used. For the proposed functions in Eq. (12) and (13), solid red lines correspond to $\alpha_{\mathrm{h}} = \frac{1}{3}$ and $s_{\mathrm{h}} = 1$, while dotted blue lines show the functions for $\alpha_{\mathrm{h}} = \frac{1}{3}$ and $s_{\mathrm{h}} \rightarrow 0$. In addition to visualizing the definitions from our previous work, the dashed yellow lines illustrate the proposed functions for $\alpha_{\mathrm{h}} = \frac{1}{2}$ and $s_{\mathrm{h}} = 1$.

previous work (Stoiber et al., 2020) and propose the following functions

$$h_{\mathrm{f}}(x) = \frac{1}{2} - \alpha_{\mathrm{h}} \tanh\left(\frac{x}{2s_{\mathrm{h}}}\right),$$
(12)

$$h_{\mathrm{b}}(x) = \frac{1}{2} + \alpha_{\mathrm{h}} \tanh\left(\frac{x}{2s_{\mathrm{h}}}\right).$$
(13)

Note that the amplitude parameter $\alpha_{\mathrm{h}} \in [0, 0.5]$ was added to the original definitions that only considered the slope parameter $s_{\mathrm{h}} \in \mathbb{R}^+$. For $\alpha_{\mathrm{h}} = \frac{1}{2}$ the equations are equal to our previous formulation. Examples of the proposed functions are shown in Fig. 4.

In addition to viewing α_{h} as a simple amplitude parameter, we are able to demonstrate that there is also another interpretation. For this, we assume that the model m is extended with a third class m_{n} that considers external effects that are independent of the foreground and background model m_{f} and m_{b}. For this scenario, we can show that $p(m_{\mathrm{f}}) = p(m_{\mathrm{b}}) = \alpha_{\mathrm{h}}$ and that $p(m_{\mathrm{n}}) = 1 - 2\alpha_{\mathrm{h}}$. Following this interpretation, the amplitude parameter thus allows us to set the probability that a pixel's color is generated by the foreground or background model in contrast to some other effect that is considered as noise. This again shows that the amplitude parameter α_{h} is able to model a constant, global uncertainty. Note that in this scenario, the original smoothed step functions that converge to zero or one are used for $p(r \mid d, m_{\mathrm{f}})$ and $p(r \mid d, m_{\mathrm{b}})$ and a constant function $p(r \mid d, m_{\mathrm{n}}) = \frac{1}{2}$ is adopted for the noise model. A detailed derivation of this extended model and

a proof of its equivalence to the use of the functions in Eq. (12) and (13) is given in Appendix A.

2.5 Posterior Probability Distribution

Given the smoothed step functions h_f and h_b that model the conditional line coordinate probabilities $p(r \mid d, m_f)$ and $p(r \mid d, m_b)$, the final expression of the posterior probability distribution from Eq. (7) can be written as

$$p(d \mid \omega, \boldsymbol{l}) \propto \prod_{r \in \omega} h_f(r - d) p_f(r) + h_b(r - d) p_b(r), \quad (14)$$

with the abbreviations $p_f(r) = p(m_f \mid \boldsymbol{l}(r))$ and $p_b(r) = p(m_b \mid \boldsymbol{l}(r))$. In the following, we provide a detailed analysis to understand how the slope parameter s_h and the amplitude parameter α_h affect this distribution. We thereby assume a contour at the correspondence line center and step functions for the pixel-wise posteriors p_f and p_b. Note that the assumption of step functions corresponds well with real-world experiments that show that, in most cases, there is a distinct split between foreground and background.

For the analysis, we start with the calculation of the first-order derivative of the log-posterior with respect to the contour distance d. The derivation is conducted similar to our previous work (Stoiber et al., 2020) and assumes continuous functions with infinitesimally small pixels. Based on a detailed derivation given in Appendix B, the closed-form solution is written as

$$\frac{\partial \ln \left(p(d \mid \omega, \boldsymbol{l}) \right)}{\partial d} = -2 \tanh^{-1} \left(2\alpha_h \tanh \left(\frac{d}{2s_h} \right) \right). \quad (15)$$

A visualization of this function for different slope and amplitude parameters α_h and s_h is given in Fig. 5. The plot shows that the amplitude parameter α_h controls not only the amplitude of h_f and h_b but also the amplitude of the first-order derivative. For $\alpha_h = \frac{1}{2}$, the first-order derivative converges to a linear function. At the same time, the parameter s_h affects both the slope of h_f and h_b and the slope of the first-order derivative. For $s_h \to 0$ it leads to a perfect step function.

For the two edge cases with $\alpha_h = \frac{1}{2}$ and $s_h \to 0$, Eq. (15) can be simplified, and we are able to calculate a closed-form solution for the posterior probability distribution. In the case of $\alpha_h = \frac{1}{2}$, for which we obtain the smoothed step functions of our previous approach (Stoiber et al., 2020), the posterior probability distribution results in a perfect Gaussian

$$p(d \mid \omega, \boldsymbol{l}) = \frac{1}{\sqrt{2\pi s_h}} \exp \left(-\frac{d^2}{2s_h} \right), \quad (16)$$
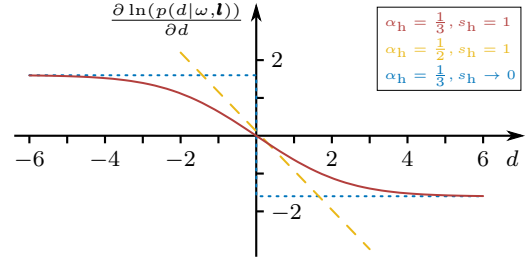
Fig. 5 First-order derivatives of the log-posterior with respect to the contour distance d for different slope and amplitude parameters s_h and α_h. The solid red line shows the derivative for $\alpha_h = \frac{1}{3}$ and $s_h = 1$, which yields a function with a smooth transition from an upper bound to a lower bound. The dashed yellow line shows the function for $\alpha_h = \frac{1}{2}$ and $s_h = 1$. This produces a linear first-order derivative. Finally, using $\alpha_h = \frac{1}{3}$ and $s_h \to 0$ results in a perfect step function illustrated by the dotted blue line.
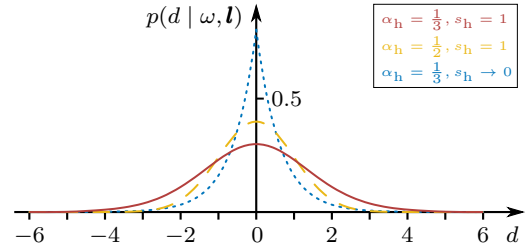
Fig. 6 Posterior probability distributions for different slope and amplitude parameters s_h and α_h. The solid red line shows the function for $\alpha_h = \frac{1}{3}$ and $s_h = 1$, which leads to a very flat distribution. Note that the function was computed numerically. Using $\alpha_h = \frac{1}{2}$ and $s_h = 1$ results in a Gaussian distribution shown by the dashed yellow line. The parameters $\alpha_h = \frac{1}{3}$ and $s_h \to 0$ yield a Laplace distribution for the posterior probability that is illustrated by a dotted blue line.

where the slope parameter s_h is equal to the variance. In the case of $s_h \to 0$, which leads to sharp step functions for h_f and h_b, the posterior probability distribution becomes a perfect Laplace distribution

$$p(d \mid \omega, \boldsymbol{l}) = \frac{1}{2b} \exp \left(-\frac{|d|}{b} \right), \quad b = \frac{1}{2 \tanh^{-1}(2\alpha_h)}, \quad (17)$$

where $b \in \mathbb{R}^+$ is the scale parameter of the Laplace distribution that depends on α_h. A detailed derivation of the two functions is provided in Appendix C. Examples for both distributions, as well as a mixed posterior distribution with $s_h = 1$ and $\alpha_h = \frac{1}{3}$, are visualized in Fig. 6. The plot shows that while the Laplace distribution has a pronounced peak, the Gaussian distribution has a smoothed maximum for which nearby values have similarly high probabilities. This coincides with our intuition that the slope parameter s_h controls local uncertainty, allowing multiple values d to be almost equally likely. At the same time, the amplitude

parameter α_h controls the size of the peak compared to its surroundings, thereby controlling global uncertainty. Combining the two parameters in a mixed distribution results in a function that is able to consider both local and global uncertainty simultaneously. Given the detailed knowledge about correspondence lines and the posterior probability distribution, we are now able to develop *SRT3D*, a highly efficient, sparse approach to region-based 3D object tracking.

3 Region-Based 3D Tracking

In this section, we first define basic mathematical concepts. This is followed by the description of a sparse viewpoint model, which avoids the rendering of the 3D model during tracking. Combining this geometry representation with the correspondence line model developed in the previous section, we are able to formulate a joint posterior probability with respect to the pose. The probability is maximized using Newton optimization with Tikhonov regularization. Finally, we define the required gradient vector and Hessian matrix for the Newton method. We thereby differentiate between global and local optimization to ensure both fast convergence and high accuracy.

3.1 Preliminaries

In the following work, we define 3D model points as $\boldsymbol{X} = \begin{bmatrix} X\ Y\ Z \end{bmatrix}^\top \in \mathbb{R}^3$ and use the tilde notation to write the homogeneous form $\widetilde{\boldsymbol{X}} = \begin{bmatrix} X\ Y\ Z\ 1 \end{bmatrix}^\top$. For the projection of a 3D model point $\boldsymbol{X}$ into the image space, we assume an undistorted image and use the pinhole camera model

$$\boldsymbol{x} = \boldsymbol{\pi}(\boldsymbol{X}) = \begin{bmatrix} \frac{X}{Z} f_x + p_x \\ \frac{Y}{Z} f_y + p_y \end{bmatrix}, \tag{18}$$

with f_x and f_y the focal lengths and p_x and p_y the principal point coordinates given in units of pixels. The inverse operation, which is the reconstruction of a 3D model point from an image coordinate $\boldsymbol{x}$ and corresponding depth value d_Z along the optical axis, can be written as

$$\boldsymbol{X} = \boldsymbol{\pi}^{-1}(\boldsymbol{x}, d_Z) = d_Z \begin{bmatrix} \frac{x - p_x}{f_x} \\ \frac{y - p_y}{f_y} \\ 1 \end{bmatrix}. \tag{19}$$

To describe the relative pose between the model reference frame M and the camera reference frame C, we
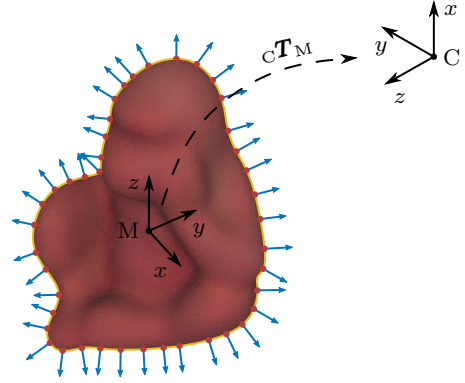

Fig. 7 Illustration of a 2D rendering computed from a 3D mesh model. The model reference frame M is shown at the center of the object, while a camera reference frame C is shown at the right upper corner of the image. The transformation from the model to the camera reference frame that is described by $_\mathrm{C}\boldsymbol{T}_\mathrm{M}$ is indicated by a dashed arrow. The contour of the rendered model is highlighted by a yellow line. Red points and blue arrows illustrate 2D contour points and approximated normal vectors.

use the homogeneous matrix $_\mathrm{C}\boldsymbol{T}_\mathrm{M} \in \mathbb{SE}(3)$. For the transformation of a 3D model point, we can then write

$$_\mathrm{C}\widetilde{\boldsymbol{X}} = {}_\mathrm{C}\boldsymbol{T}_\mathrm{M}{}_\mathrm{M}\widetilde{\boldsymbol{X}} = \begin{bmatrix} _\mathrm{C}\boldsymbol{R}_\mathrm{M}\ {}_\mathrm{C}\boldsymbol{t}_\mathrm{M} \\ \mathbf{0}\ \ 1 \end{bmatrix} {}_\mathrm{M}\widetilde{\boldsymbol{X}}, \tag{20}$$

where $_\mathrm{C}\widetilde{\boldsymbol{X}}$ and $_\mathrm{M}\widetilde{\boldsymbol{X}}$ are 3D model points written in the camera reference frame C and the model reference frame M, respectively, and where $_\mathrm{C}\boldsymbol{R}_\mathrm{M} \in \mathbb{SO}(3)$ and $_\mathrm{C}\boldsymbol{t}_\mathrm{M} \in \mathbb{R}^3$ are the rotation matrix and the translation vector that define the transformation from M to C. An illustration of the two reference frames and a homogeneous transformation matrix is given in Fig. 7.

For small variations, the angle-axis representation, which is a minimal representation, is used. With the exponential map, the rotation matrix writes as

$$\boldsymbol{R} = \exp([\boldsymbol{r}]_\times) = \boldsymbol{I} + [\boldsymbol{r}]_\times + \frac{1}{2!}[\boldsymbol{r}]_\times^2 + \frac{1}{3!}[\boldsymbol{r}]_\times^3 + ..., \tag{21}$$

where $[\boldsymbol{r}]_\times$ is the skew-symmetric matrix of $\boldsymbol{r} \in \mathbb{R}^3$. By neglecting higher-order terms of the series expansion, Eq. (21) can be linearized. We are then able to write the linear variation of a 3D model point in the camera reference frame C as

$$_\mathrm{C}\widetilde{\boldsymbol{X}}(\boldsymbol{\theta}) = \begin{bmatrix} _\mathrm{C}\boldsymbol{R}_\mathrm{M}\ {}_\mathrm{C}\boldsymbol{t}_\mathrm{M} \\ \mathbf{0}\ \ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I} + [\boldsymbol{\theta}_\mathrm{r}]_\times\ \boldsymbol{\theta}_\mathrm{t} \\ \mathbf{0}\ \ 1 \end{bmatrix} {}_\mathrm{M}\widetilde{\boldsymbol{X}}, \tag{22}$$

with the rotational variation $\boldsymbol{\theta}_\mathrm{r} \in \mathbb{R}^3$, the translational variation $\boldsymbol{\theta}_\mathrm{t} \in \mathbb{R}^3$, and the full variation vector $\boldsymbol{\theta}^\top = \begin{bmatrix} \boldsymbol{\theta}_\mathrm{r}^\top\ \boldsymbol{\theta}_\mathrm{t}^\top \end{bmatrix}$. Note that, since the object is typically moved significantly more than the camera, it is more natural to variate 3D points in the model reference frame M

instead of the camera reference frame C. Also, the variation in the model reference frame has the advantage that a simple extension of the algorithm to multiple cameras is possible.

3.2 Sparse Viewpoint Model

In contrast to most state-of-the-art region-based methods, we do not use the 3D geometry in the form of a mesh model directly. Instead, similar to Tan et al. (2017), we employ a representation that we call a sparse viewpoint model. To create this model, the 3D geometry is rendered from a number of n_v viewpoints all around the object. Virtual cameras are thereby placed on the vertices of a geodesic grid that surrounds the object. For each rendering, n_c points $\boldsymbol{x}_i \in \mathbb{R}^2$ are randomly sampled from the contour of the model. Subsequently, the vectors $\boldsymbol{n}_i \in \mathbb{R}^2$ that are normal to the contour are approximated for each point. Note that $\|\boldsymbol{n}_i\|_2 = 1$. An illustration of a rendering with sampled 2D contour points and normal vectors is shown in Fig. 7. Based on the 2D entities, 3D vectors with respect to the model reference frame are then reconstructed as follows

$$_\mathrm{M}\widetilde{\boldsymbol{X}}_i = {}_\mathrm{M}\boldsymbol{T}_\mathrm{C}\,\tilde{\boldsymbol{\pi}}^{-1}(\boldsymbol{x}_i, d_{Zi}), \tag{23}$$

$$_\mathrm{M}\boldsymbol{N}_i = {}_\mathrm{M}\boldsymbol{R}_\mathrm{C}\begin{bmatrix}\boldsymbol{n}_i \\ 0\end{bmatrix}, \tag{24}$$

where the tilde notation in $\tilde{\boldsymbol{\pi}}^{-1}$ indicates that the 3D model point is returned in homogeneous form and d_{Zi} is the depth value from the rendering. Note that in this case, C denotes the reference frame of the virtual camera from which the rendering was created. In addition to those vectors, we also compute the orientation vector $_\mathrm{M}\boldsymbol{v} = {}_\mathrm{M}\boldsymbol{R}_\mathrm{C}\boldsymbol{e}_\mathrm{Z}$ that points from the camera to the model center, where $\boldsymbol{e}_\mathrm{Z} = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^\top$. The computed 3D model points, normal vectors, and the orientation vector are then stored for each view.

The sparse viewpoint model allows for a highly efficient representation of the model contour. Given a specific pose with $_\mathrm{M}\boldsymbol{R}_\mathrm{C}$ and $_\mathrm{C}\boldsymbol{t}_\mathrm{M}$, the process of rendering the model and computing the contour reduces to a simple search for the closest precomputed view i_v

$$i_\mathrm{v} = \underset{i\in\{1,\dots,n_\mathrm{v}\}}{\arg\max}\,({}_\mathrm{M}\boldsymbol{v}_i^\top\,{}_\mathrm{M}\boldsymbol{R}_\mathrm{C}\,{}_\mathrm{C}\boldsymbol{t}_\mathrm{M}), \tag{25}$$

and the subsequent projection of the corresponding 3D model points and normal vectors into the image. Note that this high efficiency is especially important during the optimization of the joint posterior probability, where the pose changes in each iteration.

3.3 Joint Posterior Probability

In the following, we combine the developed sparse viewpoint model with the correspondence line model from Sect. 2 to define a joint posterior probability with respect to the pose variation. However, before probabilities can be calculated, the location and orientation of correspondence lines need to be defined. For this, 3D model points and normal vectors from the closest view of the sparse viewpoint model are projected into the image using the following equations

$$\boldsymbol{c}_i = \boldsymbol{\pi}\big({}_\mathrm{C}\boldsymbol{T}_\mathrm{M}\,{}_\mathrm{M}\widetilde{\boldsymbol{X}}_i\big), \tag{26}$$

$$\boldsymbol{n}_i \propto \big({}_\mathrm{C}\boldsymbol{R}_\mathrm{M}\,{}_\mathrm{M}\boldsymbol{N}_i\big)_{2\times1}, \tag{27}$$

where the normal vector $\boldsymbol{n}_i$ is normalized to $\|\boldsymbol{n}_i\|_2 = 1$ and $()_{2\times1}$ denotes the first two elements of a vector.

Once all correspondence lines have been defined, we are able to variate the current pose and calculate contour distances d_i with respect to the pose variation vector $\boldsymbol{\theta}$. Contour distances are thereby calculated as the distances along normal vectors $\boldsymbol{n}_i$ from correspondence line centers $\boldsymbol{c}_i$ to projected 3D model points $\boldsymbol{X}_i$

$$d_i(\boldsymbol{\theta}) = \boldsymbol{n}_i^\top\big(\boldsymbol{\pi}({}_\mathrm{C}\boldsymbol{X}_i(\boldsymbol{\theta})) - \boldsymbol{c}_i\big). \tag{28}$$

Note that the same 3D model points $\boldsymbol{X}_i$ are used as for the definition of correspondence lines. Also, while we do not write this explicitly, 3D model points $_\mathrm{C}\boldsymbol{X}_i$ and contour distances d_i also depend on the current pose estimate $_\mathrm{C}\boldsymbol{T}_\mathrm{M}$, which might be different from the pose that was used to define correspondence lines. An example of multiple correspondence lines with variated contour distances is shown in Fig. 8.

Finally, assuming a number of n_c independent correspondence lines and using the discrete scale-space formulation from Sect. 2.3 to improve efficiency, the joint posterior probability can be calculated as

$$p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}}) \propto \prod_{i=1}^{n_\mathrm{c}} p(d_{\mathrm{s}i}(\boldsymbol{\theta}) \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i}), \tag{29}$$

where $\boldsymbol{\mathcal{D}}$ describes the data from all correspondence lines. Note that the transformation of contour distances d_i from the original space to the discrete scale-space is given by Eq. (9). The developed joint posterior probability describes how well the current pose estimate explains the segmentation of the image into a foreground region, that corresponds to the tracked object, and a background region.

3.4 Optimization

To maximize the joint posterior probability, we estimate the variation vector $\hat{\boldsymbol{\theta}}$ and iteratively update the pose.
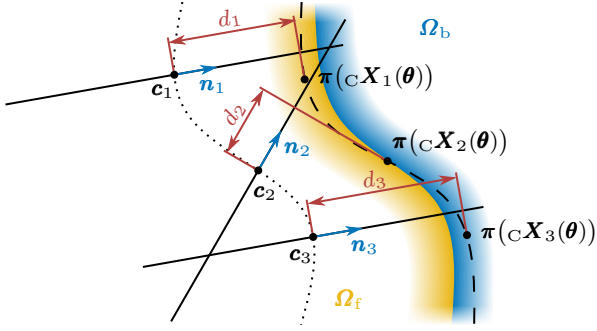
Fig. 8 Correspondence lines defined by a center $\boldsymbol{c}_i$ and a normal vector $\boldsymbol{n}_i$. Variated contour distances d_i are measured along the correspondence lines from the centers $\boldsymbol{c}_i$ to the projected 3D model points ${}_{\mathrm{C}}\boldsymbol{X}_i$ that depend all on the same pose variation $\boldsymbol{\theta}$. The object contour of the original pose estimate, which was used to define the correspondence lines, is indicated by a dotted line. The current estimate of the contour that depends on the pose variation vector $\boldsymbol{\theta}$ is shown by a dashed line. The ground truth segmentation that we try to estimate is given by the foreground region $\boldsymbol{\Omega}_{\mathrm{f}}$ in yellow and the background region $\boldsymbol{\Omega}_{\mathrm{b}}$ in blue. Note that while contours are illustrated as continuous lines, in our method, they are represented by points and normal vectors from the closest view of the sparse viewpoint model.

For a single iteration, the variation vector is calculated using the Newton method with Tikhonov regularization

$$\hat{\boldsymbol{\theta}} = \left(-\boldsymbol{H} + \begin{bmatrix} \lambda_{\mathrm{r}}\boldsymbol{I}_3 & \boldsymbol{0} \\ \boldsymbol{0} & \lambda_{\mathrm{t}}\boldsymbol{I}_3 \end{bmatrix} \right)^{-1} \boldsymbol{g}, \tag{30}$$

where $\boldsymbol{g}$ is the gradient vector, $\boldsymbol{H}$ is the Hessian matrix, $\boldsymbol{I}_3$ the 3×3 identity matrix, and λ_{r} and λ_{t} are the regularization parameters for rotation and translation, respectively. The gradient vector and the Hessian matrix are defined as the first- and second-order derivatives of the joint log-posterior

$$\boldsymbol{g}^{\top} = \frac{\partial}{\partial\boldsymbol{\theta}} \ln\left(p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}})\right)\Big|_{\boldsymbol{\theta}=0}, \tag{31}$$

$$\boldsymbol{H} = \frac{\partial^2}{\partial\boldsymbol{\theta}^2} \ln\left(p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{D}})\right)\Big|_{\boldsymbol{\theta}=0}. \tag{32}$$

Using the logarithm has the advantage that scaling terms vanish and products turn into summations. Note that the Hessian represents the curvature of the distribution at a specific location, which for Gaussian probability functions is constant and directly corresponds to the negative inverse variance. Given this probabilistic interpretation, it can be argued that regularization parameters correspond to a prior probability. This prior controls how much we believe in the previous pose compared to the current estimate described by the gradient and Hessian. Consequently, for directions in which the Hessian indicates high uncertainty, the regularization helps to keep the optimization stable and to avoid pose changes that are not supported by sufficient data.

Finally, given a robust estimate for the variation vector, the predicted pose can be updated as follows

$$_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}} = {}_{\mathrm{C}}\boldsymbol{T}_{\mathrm{M}} \begin{bmatrix} \exp([\hat{\boldsymbol{\theta}}_{\mathrm{r}}]_{\times}) \, \hat{\boldsymbol{\theta}}_{\mathrm{t}} \\ \boldsymbol{0} & 1 \end{bmatrix}. \tag{33}$$

Because of the exponential map, no orthonormalization is necessary. By iteratively repeating this process, we are able to optimize towards the pose that best explains the segmentation of the image.

3.5 Gradient and Hessian Approximation

In the following, the gradient vector and the Hessian matrix are approximated in a way that ensures both fast convergence and high accuracy. Using the chain rule, we write

$$\boldsymbol{g}^{\top} = \sum_{i=1}^{n_{\mathrm{c}}} \frac{\partial \ln\left(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\right)}{\partial d_{\mathrm{s}i}} \frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=0}, \tag{34}$$

$$\boldsymbol{H} \approx \sum_{i=1}^{n_{\mathrm{c}}} \frac{\partial^2 \ln\left(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\right)}{\partial d_{\mathrm{s}i}^2} \left(\frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial\boldsymbol{\theta}} \right)^{\top} \\ \left(\frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} \frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial\boldsymbol{\theta}} \right)\bigg|_{\boldsymbol{\theta}=0}. \tag{35}$$

Note that for the Hessian matrix, second-order partial derivatives with respect to $d_{\mathrm{s}i}$ and ${}_{\mathrm{C}}\boldsymbol{X}_i$ are neglected. Resulting errors are left to the iterative nature of the optimization. Using Eq. (22), the first-order derivative of the 3D model point ${}_{\mathrm{C}}\boldsymbol{X}_i$ is calculated as

$$\frac{\partial {}_{\mathrm{C}}\boldsymbol{X}_i}{\partial\boldsymbol{\theta}} = {}_{\mathrm{C}}\boldsymbol{R}_{\mathrm{M}} \left[-[{}_{\mathrm{M}}\boldsymbol{X}_i]_{\times} \ \ \boldsymbol{I}_3 \right]. \tag{36}$$

With respect to the scaled contour distance $d_{\mathrm{s}i}$, both Eq. (28) and (9) are used to write

$$\frac{\partial d_{\mathrm{s}i}}{\partial {}_{\mathrm{C}}\boldsymbol{X}_i} = \frac{\bar{n}_i}{s} \frac{1}{{}_{\mathrm{C}}Z_i^2} \big[n_{xi}f_{x\,\mathrm{C}}Z_i \ \ n_{yi}f_{y\,\mathrm{C}}Z_i \\ -n_{xi}f_x {}_{\mathrm{C}}X_i - n_{yi}f_y {}_{\mathrm{C}}Y_i \big]. \tag{37}$$

For the calculation of the required first- and second-order derivatives of the log-posterior, we differentiate between global and local optimization. To some extent, this is similar to our previous approach (Stoiber et al., 2020). However, in contrast to that work, we propose different approximations for the local optimization. Also, we either apply global or local optimization and use the same definition of derivatives for all correspondence lines instead of mixing them.

In the case of global optimization, the posterior probability distribution of individual correspondence lines is approximated by a normal distribution $\mathcal{N}(d_{\mathrm{s}i} \mid \mu_i, \sigma_i^2)$. The required mean and standard deviation μ_i and σ_i
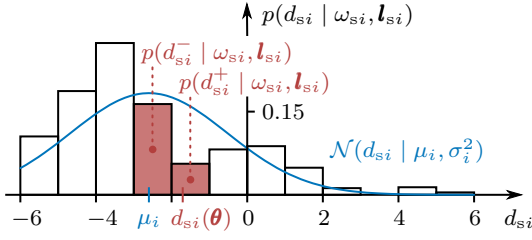
Fig. 9 Discrete posterior probability distribution with noisy measurements. For global optimization, the distribution is approximated by a normal distribution $\mathcal{N}(d_{\mathrm{s}i} \mid \mu_i, \sigma_i^2)$. The normal distribution and its mean μ_i are illustrated in blue. In the case of local optimization, only two discrete probability values that are closest to the current estimate of the contour distance $d_{\mathrm{s}i}(\boldsymbol{\theta})$ are considered. The two discrete probability values $p(d_{\mathrm{s}i}^- \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})$ and $p(d_{\mathrm{s}i}^+ \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})$, which are used to approximate the first-order derivative, are colored in red.

are thereby estimated from a set of discretized contour distances $d_{\mathrm{s}i}$ and their corresponding probability values. An example of the approximation of a discrete posterior probability distribution is shown in Fig. 9. Based on the normal distribution, the first- and second-order derivatives are calculated as

$$\frac{\partial \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}} \approx -\frac{1}{\sigma_i^2}(d_{\mathrm{s}i} - \mu_i), \qquad (38)$$

$$\frac{\partial^2 \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}^2} \approx -\frac{1}{\sigma_i^2}. \qquad (39)$$

The approximated derivatives direct the optimization towards the mean μ_i, using the variance σ_i^2 to consider uncertainty. Note that while in the real world, the mean does not exactly coincide with the maximum, it is typically quite close. At the same time, using the approximation has the advantage of fast convergence and that the optimization avoids local minima resulting from invalid pixel-wise posteriors and image noise.

Once the optimization is closer to the maximum, the global mean is not a good enough estimate, and more detailed refinement is required. In such cases, the algorithm switches to local optimization. We thereby use the probability values of the two discrete contour distances $d_{\mathrm{s}i}^-$ and $d_{\mathrm{s}i}^+$ that are closest to the current estimate $d_{\mathrm{s}i}(\boldsymbol{\theta})$ and approximate the first-order derivatives using a weighting term $\frac{\alpha_{\mathrm{s}}}{\sigma_i^2}$ and finite differences

$$\frac{\partial \ln\big(p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}} \approx \frac{\alpha_{\mathrm{s}}}{\sigma_i^2} \ln\left(\frac{p(d_{\mathrm{s}i}^+ \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})}{p(d_{\mathrm{s}i}^- \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})}\right). \qquad (40)$$

For second-order derivatives, the global approximation from Eq. (39) is used. Note that weighting the first-order derivative with the variance σ_i^2 improves robustness because correspondence lines with high uncertainty are considered less important. Simultaneously, the step size α_{s} helps to balance the weight and specifies how far

the optimization proceeds, directly scaling the variation vector $\hat{\boldsymbol{\theta}}$. The same first- and second-order derivatives can also be derived using inverse-variance weighting and a constant curvature of $\frac{1}{\alpha_{\mathrm{s}}}$ for the second-order derivative. A detailed derivation of this interpretation is given in Appendix D.

Finally, apart from the choice of derivatives, the parameterization of smoothed step functions and the corresponding shape of posterior probability distributions significantly influences the optimization. To study this effect, we consider the first-order derivatives of the log-posteriors that are shown in Fig. 5. While for Gaussian distributions, linear first-order derivatives lead to the estimation of the weighted mean over all correspondence lines, for Laplace distributions, binary derivatives guide the optimization towards the weighted median. Note that this again corresponds well to the interpretation of local and global uncertainty modeled by the slope parameter s_{h} and the amplitude parameter α_{h}. If only local uncertainty exists, it is advantageous to consider the magnitude of errors in the contour distance and optimize for the mean. At the same time, in the case of global noise, it is reasonable to only consider the direction of errors, and conduct the optimization with respect to the median.

4 Implementation

The following section provides implementation details for the developed algorithm. We thereby start with the generation of the sparse viewpoint model and the calculation of color histograms. This is followed by a description of the tracking process. Finally, we explain how known occlusions can be considered. All mentioned parameter values are carefully chosen to maximize tracking quality while not requiring unreasonable amounts of computation. Note that the source code of *SRT3D* is publicly available on *GitHub*[1] to ensure reproducibility and to allow full reusability.

4.1 Sparse Viewpoint Model

For the sparse viewpoint model, $n_{\mathrm{v}} = 2562$ different views are considered. They are generated by subdividing the triangles of an icosahedron 4 times, resulting in an angle of approximately $4°$ between neighboring views. Virtual cameras that are used for the rendering are placed at a distance of $0.8\,\mathrm{m}$ to the object center. For all views, the orientation vector $_{\mathrm{M}}\boldsymbol{v}$ and a constant number of $n_{\mathrm{c}} = 200$ model points $_{\mathrm{M}}\boldsymbol{X}_i$ and normal vectors $_{\mathrm{M}}\boldsymbol{N}_i$ are computed. In addition, for each

---

[1] https://github.com/DLR-RM/3DObjectTracking

point and view, we also compute so-called continuous distances for the foreground and background. Continuous distances thereby describe the distance from the 2D model point $\boldsymbol{x}_i$ along the line defined by the normal vector $\boldsymbol{n}_i$ for which the foreground and background are not interrupted by each other. After their computation in the rendered image, they are converted and stored in meters. The values are later used by the tracker to disable individual correspondence lines for which continuous distances are below a certain threshold, and the assumption that only a single transition between foreground and background is present in the correspondence line is not sufficiently fulfilled.

4.2 Color Histograms

For the estimation of the color probability distributions $p(\boldsymbol{y} \mid m_\mathrm{f})$ and $p(\boldsymbol{y} \mid m_\mathrm{b})$, color histograms are used. Each dimension of the RGB color space is discretized by 32 equidistant bins, leading to a total of 32768 values. The computation of the color histograms is started either from the current pose estimate or from an initial pose, provided, for example, by a 3D object detection pipeline. Based on this pose, 3D model points and normal vectors are projected into the image using Eq. (26) and (27). After an offset of one pixel, the first 18 pixels are considered in both the positive and negative direction of the normal vector. Pixel colors along this line are assigned to either the foreground or background histogram, depending on which side of the projected model point they are. Note that fewer than 18 pixels are considered if a transition between foreground and background occurs within a shorter distance. Also, in cases where the contour location is more uncertain, it is reasonable to use an offset larger than one pixel.

Due to motion or dynamic illumination, color statistics of both the foreground and background are continuously changing during tracking. To take those changes into account while at the same time considering previous observations, we use online adaptation. Based on Bibby and Reid (2008), we thereby update the histograms as follows

$$p_t(\boldsymbol{y} \mid m_i) = \alpha_i p(\boldsymbol{y} \mid m_i) + (1 - \alpha_i) p_{t-1}(\boldsymbol{y} \mid m_i), \quad (41)$$

with $i \in \{\mathrm{f}, \mathrm{b}\}$ and $\alpha_\mathrm{f} = 0.2$ and $\alpha_\mathrm{b} = 0.2$ the learning rates for the foreground and background, respectively. Note that $p(\boldsymbol{y} \mid m_i)$ is the observed histogram, while $p_t(\boldsymbol{y} \mid m_i)$ and $p_{t-1}(\boldsymbol{y} \mid m_i)$ are the adapted histograms of the current and previous time step, respectively. For initialization, we directly use the observed histograms instead of blending them with previous values.

4.3 Tracking Process

To start tracking, an initial pose is required, which is typically provided either from a 3D object detection pipeline or from dataset annotations. Based on this pose and a corresponding camera image, the color histograms for the foreground and background are initialized. After initialization, a tracking step is executed for each new image that is streamed from the camera. An overview of all computation that is performed in a single tracking step is given in Algorithm 1.

Algorithm 1 Tracking Step

1: Update camera image
2: **for** $i = 1, 2, \ldots, 7$ **do**
3:    **Optional:** Render occlusion mask
4:    Find closest view of the sparse viewpoint model
5:    Define correspondence lines in the image
6:    Compute discrete distributions $p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})$
7:    **for** $j = 1, 2$ **do**
8:       Calculate gradient $\boldsymbol{g}$ and Hessian $\boldsymbol{H}$
9:       Estimate variation $\hat{\boldsymbol{\theta}}$ and update pose $_\mathrm{C}\boldsymbol{T}_\mathrm{M}$
10:    **end for**
11: **end for**
12: Update color histograms $p(\boldsymbol{y} \mid m_\mathrm{f})$ and $p(\boldsymbol{y} \mid m_\mathrm{b})$

Starting from a new image and the previous pose estimate $_\mathrm{C}\boldsymbol{T}_\mathrm{M}$, we first retrieve the closest view of the sparse viewpoint model. Model points $_\mathrm{M}\boldsymbol{X}_i$ and normal vectors $_\mathrm{M}\boldsymbol{N}_i$ are then projected into the image plane to define correspondence lines. After that, continuous distances from the sparse viewpoint model are used to reject correspondence lines with distances that are below 6 segments. For the remaining correspondence lines, the posterior probability distribution $p(d_{\mathrm{s}i} \mid \omega_{\mathrm{s}i}, \boldsymbol{l}_{\mathrm{s}i})$ is evaluated at 12 discrete values $d_{\mathrm{s}i} \in \{-5.5, -4.5, \ldots, 5.5\}$. In the calculation, we use 8 precomputed values for the smoothed step functions h_f and h_b, corresponding to $x \in \{-3.5, -2.5, \ldots, 3.5\}$. Also, a minimal offset Δr_i is chosen such that the line coordinates r_i point to pixel centers while the scaled line coordinates $r_{\mathrm{s}i}$ ensure matching values for $x = r_{\mathrm{s}i} - d_{\mathrm{s}i}$. In our case, this means that $r_{\mathrm{s}i} \in \mathbb{Z}$. Having computed the distributions, two iterations of the regularized Newton optimization are executed. For the first iteration, the global optimization is used to quickly converge towards a rough pose estimate. In the second iteration, the local optimization is employed to refine this pose, using a step size of $\alpha_\mathrm{s} = 1.3$. As regularization parameters, we use $\lambda_\mathrm{r} = 5000$ and $\lambda_\mathrm{t} = 500000$.

To find the final pose, the process is repeated seven times. We thereby choose larger scales of $s = 5$ for the first iteration and $s = 2$ for the second and third iterations. In all other iterations, a scale of $s = 1$ is adopted.

This choice has the effect that a large area with low resolution is considered in the beginning, while short lines with high resolution are used in later iterations. An example of correspondence lines at different scales is shown in Fig. 1. Note that scale values typically depend on the area that needs to be covered by the tracker and the size of frame-to-frame pose differences. Finally, having estimated the pose for the current image, the prediction is used to update the color histograms. After that, the tracker waits for a new image to arrive.

4.4 Occlusion Modeling

While the algorithm is quite robust to unknown occlusions, tracking results can be further improved by explicitly considering known occlusions. For this, an ID is assigned to each known object. All objects are then rendered into a depth image and an image that contains object ID values. Using a custom shader, we combine information from the two images and compute an occlusion mask that binary encodes in each pixel which objects are visible. To consider uncertainty in the object pose, the shader evaluates a region with a radius of 4 pixels and assigns the object ID with the smallest depth value to the center. If only the background is present, all object IDs are considered visible. In order to improve efficiency, a smaller image with a fourth of the camera resolution is used. Finally, to reject occluded correspondence lines, the algorithm simply checks occlusion mask values at correspondence line centers.

5 Evaluation

In this section, we present an extensive evaluation of our approach, *SRT3D*. Both the *Region-Based Object Tracking (RBOT)* dataset (Tjaden et al., 2018) and the *Object Pose Tracking (OPT)* dataset (Wu et al., 2017) are used to compare our method to the current state of the art in region-based tracking. We thereby evaluate the quality of the predicted pose as well as the speed of the algorithm. Also, a detailed parameter analysis is conducted that assesses the importance of different settings. Finally, we discuss essential design considerations and remaining limitations. In addition to the content in this section, we provide real-world videos on our project site[1] that demonstrate the tracker's performance.

Fig. 10 Overview of all objects in the *RBOT* dataset (Tjaden et al., 2018). Objects from the *LINEMOD* dataset (Hinterstoisser et al., 2013) and *Rigid Pose* dataset (Pauwels et al., 2013) are marked with ⋆ and ⋄, respectively.
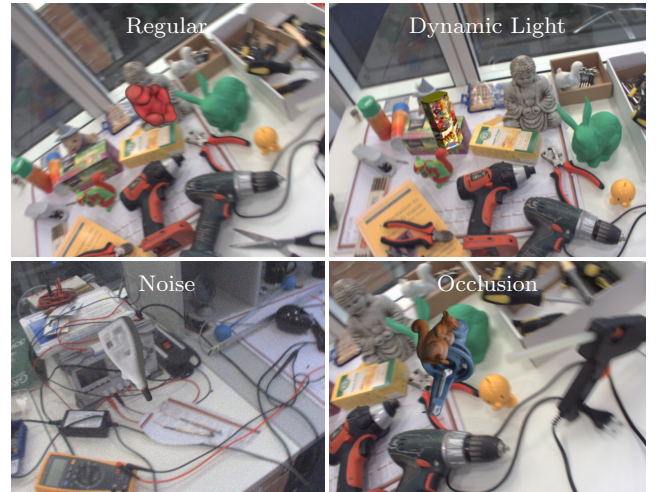
Fig. 11 Images from the *RBOT* dataset (Tjaden et al., 2018) with one example image for the *regular*, *dynamic light*, *noise*, and *occlusion* sequence. The sequences show the *ape*, *candy*, *glue*, and *vise* objects, respectively. In addition, the *occlusion* sequence features a squirrel object that occludes the *vise*.

5.1 RBOT Dataset

In the following, we first introduce the *RBOT* dataset, discuss the conducted experiments, and finally compare our results to the current state of the art. The *RBOT* dataset consists of a collection of 18 objects that are shown in Fig. 10. For each object, four sequences exist: a *regular* version, one with *dynamic light*, a sequence with both dynamic light and Gaussian *noise*, and one with dynamic light and an additional squirrel object that leads to *occlusion*. An example image for each sequence is shown in Fig. 11. Each sequence consists of

Table 1 Tracking success rates for state-of-the-art approaches on the *RBOT* dataset (Tjaden et al., 2018). Methods that are not purely region-based are indicated by a *. The best results are highlighted in bold. The second-best values are underlined.

Approach	Ape	Soda	Vise	Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	Egg Box	Glue	Iron	Candy	Lamp	Phone	Squirrel	**Avg.**
Regular																			
Tjaden et al. (2018)	85.0	39.0	98.9	82.4	79.7	87.6	95.9	93.3	78.1	93.0	86.8	74.6	38.9	81.0	46.8	<u>97.5</u>	80.7	99.4	79.9
Zhong et al. (2020b)	88.8	41.3	94.0	85.9	86.9	89.0	98.5	93.7	83.1	87.3	86.2	78.5	58.6	86.3	57.9	91.7	85.0	96.2	82.7
Huang et al. (2020)*	91.9	44.8	**99.7**	89.1	89.3	90.6	97.4	95.9	83.9	<u>97.6</u>	91.8	84.4	59.0	<u>92.5</u>	74.3	97.4	86.4	99.7	86.9
Stoiber et al. (2020)	<u>96.4</u>	53.2	98.8	<u>93.9</u>	93.0	92.7	<u>99.7</u>	<u>97.1</u>	92.5	92.5	<u>93.7</u>	88.5	<u>70.0</u>	92.1	<u>78.8</u>	95.5	92.5	99.6	<u>90.0</u>
Liu et al. (2021)*	93.7	39.3	98.4	91.6	84.6	89.2	97.9	95.9	86.3	95.1	93.4	77.7	61.5	87.8	65.0	95.2	85.7	**99.8**	85.5
Li et al. (2021)*	92.8	42.6	96.8	87.5	90.7	86.2	99.0	96.9	86.8	94.6	90.4	87.0	57.6	88.7	59.9	96.5	90.6	99.5	85.8
Sun et al. (2021)*	93.0	<u>55.2</u>	99.3	85.4	<u>96.1</u>	<u>93.9</u>	98.0	95.6	79.5	**98.2**	89.7	<u>89.1</u>	66.5	91.3	60.6	**98.6**	<u>95.6</u>	99.6	88.1
SRT3D (Ours)	**98.8**	**65.1**	<u>99.6</u>	**96.0**	**98.0**	**96.5**	**100.0**	**98.4**	**94.1**	96.9	**98.0**	**95.3**	**79.3**	**96.0**	**90.3**	97.4	**96.2**	**99.8**	**94.2**
Dynamic Light																			
Tjaden et al. (2018)	84.9	42.0	99.0	81.3	84.3	88.9	95.6	92.5	77.5	94.6	86.4	77.3	52.9	77.9	47.9	96.9	81.7	99.3	81.2
Zhong et al. (2020b)	89.7	40.2	92.7	86.5	86.6	89.2	98.3	93.9	81.8	88.4	83.9	76.8	55.3	79.3	54.7	88.7	81.0	95.8	81.3
Huang et al. (2020)*	91.8	42.3	98.9	89.9	91.3	87.8	97.6	94.5	84.5	<u>98.1</u>	91.9	86.7	66.2	90.9	73.2	<u>97.1</u>	89.2	99.6	87.3
Stoiber et al. (2020)	<u>96.5</u>	54.6	99.1	<u>93.9</u>	93.1	<u>94.7</u>	99.5	<u>97.0</u>	<u>93.0</u>	93.4	<u>93.3</u>	<u>92.6</u>	<u>74.9</u>	91.0	<u>79.2</u>	95.6	89.8	99.5	<u>90.6</u>
Liu et al. (2021)*	93.5	38.2	98.4	88.8	87.0	88.5	98.1	94.4	85.1	95.1	92.7	76.1	58.1	79.6	62.1	93.2	84.7	99.6	84.1
Li et al. (2021)*	93.5	43.1	96.6	88.5	92.8	86.0	<u>99.6</u>	95.5	85.7	96.8	91.1	90.2	68.4	86.8	59.7	96.1	91.5	99.2	86.7
Sun et al. (2021)*	93.8	<u>55.9</u>	<u>99.6</u>	85.6	<u>97.7</u>	93.7	97.7	96.5	78.3	**98.6**	91.0	91.6	72.1	90.7	63.0	**98.9**	<u>94.4</u>	**100.0**	88.8
SRT3D (Ours)	**98.2**	**65.2**	<u>99.2</u>	**95.6**	<u>97.5</u>	**98.1**	**100.0**	**98.5**	**94.2**	97.5	**97.9**	**96.9**	**86.1**	**95.2**	**89.3**	97.0	**95.9**	<u>99.9</u>	**94.6**
Noise																			
Tjaden et al. (2018)	77.5	44.5	91.5	82.9	51.7	38.4	95.1	69.2	24.4	64.3	88.5	11.2	2.9	46.7	32.7	57.3	44.1	96.6	56.6
Zhong et al. (2020b)	79.3	35.2	82.6	86.2	65.1	56.9	96.9	67.0	37.5	75.2	85.4	35.2	18.9	63.7	35.4	64.6	66.3	93.2	63.6
Huang et al. (2020)*	89.0	45.0	89.5	90.2	68.9	38.3	95.9	72.8	20.1	85.5	92.2	26.8	15.8	66.2	52.2	58.3	65.1	98.4	65.0
Stoiber et al. (2020)	91.9	53.3	90.2	<u>92.6</u>	67.9	59.3	98.4	80.6	<u>43.5</u>	78.1	92.5	44.0	31.3	72.3	<u>62.0</u>	59.9	71.7	98.3	71.5
Liu et al. (2021)*	84.7	33.0	88.8	89.5	56.4	50.1	94.1	66.5	32.3	79.6	<u>94.2</u>	29.6	19.9	63.4	40.3	61.6	62.4	96.9	63.5
Li et al. (2021)*	89.1	44.0	91.6	89.4	75.2	62.3	98.6	77.3	41.2	81.5	91.6	54.5	31.8	65.0	46.0	<u>78.5</u>	69.6	97.6	71.4
Sun et al. (2021)*	<u>92.5</u>	<u>56.2</u>	**98.0**	85.1	**91.7**	**79.0**	97.7	<u>86.2</u>	40.1	**96.6**	90.8	**70.2**	**50.9**	**84.3**	49.9	**91.2**	**89.4**	99.4	<u>80.5</u>
SRT3D (Ours)	**96.9**	**61.9**	<u>95.4</u>	**95.7**	<u>84.5</u>	<u>73.9</u>	**99.9**	**90.3**	**62.2**	<u>87.8</u>	**97.6**	<u>62.2</u>	<u>43.4</u>	**84.3**	**78.2**	73.3	<u>83.1</u>	**99.7**	**81.7**
Unmodeled Occlusion																			
Tjaden et al. (2018)	80.0	42.7	91.8	73.5	76.1	81.7	89.8	82.6	68.7	86.7	80.5	67.0	46.6	64.0	43.6	88.8	68.6	86.2	73.3
Zhong et al. (2020b)	83.9	38.1	92.4	81.5	81.3	85.5	<u>97.5</u>	88.9	76.1	87.5	81.7	72.7	52.5	77.2	53.9	88.5	79.3	92.5	78.4
Huang et al. (2020)*	86.2	46.3	<u>97.8</u>	87.5	86.5	86.3	95.7	90.7	78.8	96.5	86.0	80.6	59.9	86.8	69.6	93.3	81.8	95.8	83.6
Stoiber et al. (2020)	90.8	51.7	95.9	<u>88.5</u>	88.0	<u>90.5</u>	96.9	91.6	<u>87.1</u>	90.3	86.4	<u>85.6</u>	65.8	87.0	<u>72.7</u>	91.2	84.0	97.0	<u>85.6</u>
Liu et al. (2021)*	87.1	36.7	91.7	78.8	79.2	82.5	92.8	86.1	78.0	90.2	83.4	72.0	52.3	72.8	55.9	86.9	77.8	93.0	77.6
Li et al. (2021)*	89.3	43.3	92.2	83.1	84.1	79.0	94.5	88.6	76.2	90.4	<u>87.0</u>	80.7	61.6	75.3	53.1	91.1	81.9	93.4	80.3
Sun et al. (2021)*	<u>91.3</u>	<u>56.7</u>	<u>97.8</u>	82.0	<u>92.8</u>	89.9	96.6	<u>92.2</u>	71.8	**97.0**	85.0	84.6	<u>66.9</u>	<u>87.7</u>	56.1	<u>95.1</u>	89.8	98.2	85.1
SRT3D (Ours)	**96.5**	**66.8**	**99.0**	**95.8**	**95.0**	**95.9**	**100.0**	**97.6**	**92.2**	<u>96.6</u>	**95.0**	**94.4**	**79.0**	**94.7**	**89.8**	**95.7**	**93.6**	**99.6**	**93.2**
Modeled Occlusion																			
Tjaden et al. (2018)	82.0	42.0	95.7	81.1	78.7	83.4	92.8	87.9	74.3	91.7	84.8	71.0	49.1	73.0	46.3	90.9	76.2	96.9	77.7
Huang et al. (2020)*	87.8	45.5	<u>98.1</u>	87.2	89.0	89.8	95.1	91.4	77.4	**97.1**	87.7	83.0	62.5	88.6	69.7	<u>94.1</u>	86.0	98.9	84.9
Stoiber et al. (2020)	<u>95.0</u>	<u>53.8</u>	97.8	<u>92.4</u>	90.6	93.5	99.1	96.3	91.5	92.6	<u>90.9</u>	91.3	70.5	91.8	<u>77.2</u>	93.7	<u>87.0</u>	<u>99.0</u>	<u>89.1</u>
SRT3D (Ours)	**97.9**	**68.3**	**99.2**	**95.4**	**96.8**	**96.4**	**99.6**	**98.6**	**93.0**	<u>96.4</u>	**96.6**	**96.2**	**82.9**	**95.1**	**91.0**	**96.0**	**94.5**	**99.6**	**94.1**

1001 semi-synthetic monocular images, where objects were rendered into real-world images, recorded from a hand-held camera that moves around a cluttered desk.

In the evaluation, experiments are performed as defined by Tjaden et al. (2018). The required translational and rotational errors are calculated as

$$e_\mathrm{t}(t_k) = \big\| {}_\mathrm{C}\boldsymbol{t}_\mathrm{M}(t_k) - {}_\mathrm{C}\boldsymbol{t}_{\mathrm{M_{gt}}}(t_k) \big\|_2, \tag{42}$$

$$e_\mathrm{r}(t_k) = \cos^{-1}\left(\frac{\mathrm{trace}({}_\mathrm{C}\boldsymbol{R}_\mathrm{M}(t_k)^\top {}_\mathrm{C}\boldsymbol{R}_{\mathrm{M_{gt}}}(t_k)) - 1}{2} \right), \tag{43}$$

where ${}_\mathrm{C}\boldsymbol{R}_{\mathrm{M_{gt}}}(t_k)$ and ${}_\mathrm{C}\boldsymbol{t}_{\mathrm{M_{gt}}}(t_k)$ are the ground-truth rotation matrix and translation vector for the frame $k \in \{0, \ldots, 1000\}$. A pose is considered successful if both $e_\mathrm{t}(t_k) < 5\,\mathrm{cm}$ and $e_\mathrm{r}(t_k) < 5°$. After the initialization of the tracker with the ground-truth pose at t_0, the tracker runs until either the recorded sequence ends or tracking was unsuccessful. In the case of unsuccessful tracking, the algorithm is re-initialized with the ground-truth pose at t_k. For the *occlusion* sequence, the method is evaluated with and without occlusion modeling. In the case of occlusion modeling, both objects are tracked simultaneously. Unsuccessful tracking of the occluding squirrel object is not considered in the reported tracking success. Finally, for the remaining tracker settings, we use $\alpha_\mathrm{h} = 0.36$ and $s_\mathrm{h} \to 0$. A detailed analysis of this choice is given in Sect. 5.3.

Results of the evaluation are shown in Table 1. Our approach is compared to the current state of the art in region-based tracking, as well as the edge-based methods of Huang et al. (2020), algorithms of Li et al. (2021) and Sun et al. (2021) that combine edge and region in-

Table 2 Average runtimes per frame and usage of a GPU for state-of-the-art approaches. Methods that are not purely region-based are indicated by a ⋆. For the occlusion modeling scenario, which considers the tracking of two objects, values are shown in parenthesis.

Approach	No GPU	Runtime
Tjaden et al. (2018)	✗	$15.5 \sim 21.8 \, \text{ms}$
Zhong et al. (2020b)	✗	$41.2 \, \text{ms}$
Huang et al. (2020)⋆	✗	$33.1 \, \text{ms}$
Stoiber et al. (2020)	✓(✗)	$1.0 \, \text{ms} \, (7.4 \, \text{ms})$
Liu et al. (2021)⋆	✗	$6.9 \, \text{ms}$
Li et al. (2021)⋆	✗	$32.1 \, \text{ms}$
Sun et al. (2021)⋆	✗	$40.0 \sim 50.0 \, \text{ms}$
SRT3D (Ours)	✓(✗)	$1.1 \, \text{ms} \, (5.1 \, \text{ms})$

formation, and the method of Liu et al. (2021) that uses descriptor fields in addition to region-based techniques. The comparison shows that *SRT3D* performs significantly better than previous methods, achieving superior results for most objects and performing best on average. This difference becomes even larger for purely region-based methods, with our algorithm performing best for almost all objects and sequences. Considering the average success rate, our approach performs about five percentage points better than the combined method of Sun et al. (2021), six percentage points better than Stoiber et al. (2020), nine percentage points better than Li et al. (2021), and more than 14 percentage points better than the next best, dense, region-based approach by Zhong and Zhang (2019). The superior tracking success compared to our previous approach is especially interesting since the main differences are only an extended smoothed step function and some changes with respect to optimization. Also, in comparison to all other, dense approaches, no advanced segmentation model is used, which, in theory, is a significant disadvantage.

In addition to tracking success, we also compare average runtimes. A summary for the different algorithms is given in Table 2. The evaluation of *SRT3D* and our previous method was conducted on the same computer with an *Intel Xeon E5-1630 v4* CPU and a *Nvidia Quadro P600* GPU. Because of the similarities of the two approaches, we obtain a comparable average runtime of 1.1 ms for the case without occlusion modeling and an improved average execution time of 5.1 ms for the *modeled occlusion* scenario. Note that in the case of occlusion modeling, occlusion masks have to be rendered, and the reported time is for the simultaneous tracking of two objects. In comparison, except for the algorithm of Liu et al. (2021), for which the execution time is six times higher, all other methods report average runtimes that are more than one order
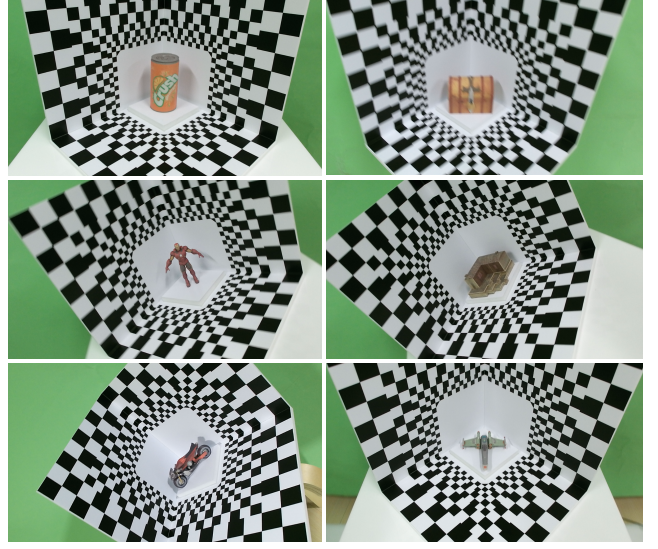

Fig. 12 Images from the *OPT* dataset (Wu et al., 2017), featuring the *soda*, *chest*, *ironman*, *house*, *bike*, and *jet* object.

of magnitude larger. The difference is even more impressive since *SRT3D* and our previous approach only utilize a single CPU core and do not require a GPU. In contrast, most competing methods typically use multi-threading and heavily depend on a GPU. In conclusion, while different resources and computers were used, the obtained results highlight the superior efficiency of our sparse region-based method.

5.2 OPT Dataset

While the semi-synthetic *RBOT* dataset features a large number of objects, a difficult, highly cluttered background, and perfect ground-truth, objects are simulated with limited realism, and only very little motion blur is applied. Those shortcomings are complemented by the *OPT* dataset (Wu et al., 2017), which contains real-world recordings of 3D printed objects on a white background with different speeds and levels of motion blur. In total, the dataset includes six objects and consists of 552 real-world sequences with various lighting conditions and defined trajectories recorded by a robot arm. An example image for each object is shown in Fig. 12. The sequences are classified into the following categories: translation, forward and backward, in-plane rotation, out-of-plane rotation, flashing light, moving light, and free motion.

In the experiments, the metric of Wu et al. (2017) is used. For this, we compute the average vertex error

$$e_{\text{v}}(t_k) = \frac{1}{n} \sum_{i=1}^{n} \left\| \left({}_{\text{M}}\widetilde{\boldsymbol{X}}_i - {}_{\text{M}}\boldsymbol{T}_{\text{M}_{\text{gt}}}(t_k){}_{\text{M}}\widetilde{\boldsymbol{X}}_i \right)_{3 \times 1} \right\|_2, \quad (44)$$

Table 3 AUC scores between zero and twenty for the evaluation on the *OPT* dataset (Wu et al., 2017), comparing our approach to multiple other algorithms. The best results are highlighted in bold. The second-best values are underlined.

Approach	Soda	Chest	Ironman	House	Bike	Jet	Avg.
PWP3D	5.87	5.55	3.92	3.58	5.36	5.81	5.01
ElasticFusion	1.90	1.53	1.69	2.70	1.57	1.86	1.87
UDP	8.49	6.79	5.25	5.97	6.10	2.34	5.82
ORB-SLAM2	13.44	<u>15.53</u>	11.20	**17.28**	10.41	9.93	12.97
Bugaev et al. (2018)	<u>14.85</u>	14.97	<u>14.71</u>	14.48	12.55	**17.17**	<u>14.79</u>
Tjaden et al. (2018)	8.86	11.76	11.99	10.15	11.90	13.22	11.31
Zhong et al. (2020b)	9.01	12.24	11.21	13.61	12.83	15.44	12.39
Li et al. (2021)	9.00	14.92	13.44	13.60	<u>12.85</u>	10.64	12.41
SRT3D (Ours)	**15.64**	**16.30**	**17.41**	<u>16.36</u>	**13.02**	<u>15.64</u>	**15.73**

with $\widetilde{\boldsymbol{X}}_i$ a vertex in the 3D mesh geometry of the object and n the number of vertices. Tracking is considered successful if $e_v(t_k) < k_e d$, where d is the object diameter computed from the maximum vertex distance and k_e is an error threshold. The tracking quality for all frames is then measured using an area under curve (AUC) score that integrates the percentage value of successfully tracked poses over the interval $k_e \in [0, 0.2]$. This results in AUC scores between zero and twenty. For the tracker, the amplitude parameter $\alpha_h = 0.42$ and the slope parameter $s_h = 0.5$ are used. Also, for the rotationally symmetric *soda* object, a larger rotational regularization parameter of $\lambda_r = 500000$ is adopted. The main reason is that the object geometry of the *soda* object does not constrain the rotation around the vertical axis. In such cases, fluctuations in the gradient and Hessian can lead to drift in the object's orientation. Using more regularization allows us to mitigate this problem.

Results for the experiments on the *OPT* dataset are shown in Table. 3. We thereby compare *SRT3D* to state-of-the-art region-based tracking approaches, as well as an approach from Bugaev et al. (2018) and prominent methods such as *PWP3D* (Prisacariu and Reid, 2012), *ElasticFusion* (Whelan et al., 2015), *UDP* (Brachmann et al., 2016), and *ORB-SLAM2* (Mur-Artal and Tardós, 2017). Note that not all algorithms are dedicated 3D tracking solutions. *UDP* is a monocular pose detection method, while *ElasticFusion* and *ORB-SLAM2* are visual SLAM approaches for camera pose localization that are applied to the silhouette of the object. For more information on the evaluation of those algorithms, please refer to Wu et al. (2017).

The comparison shows that our approach performs significantly better than the current state of the art in region-based tracking developed by Li et al. (2021), Zhong et al. (2020b) and Tjaden et al. (2018), achieving higher AUC scores for each of the six objects. Also,
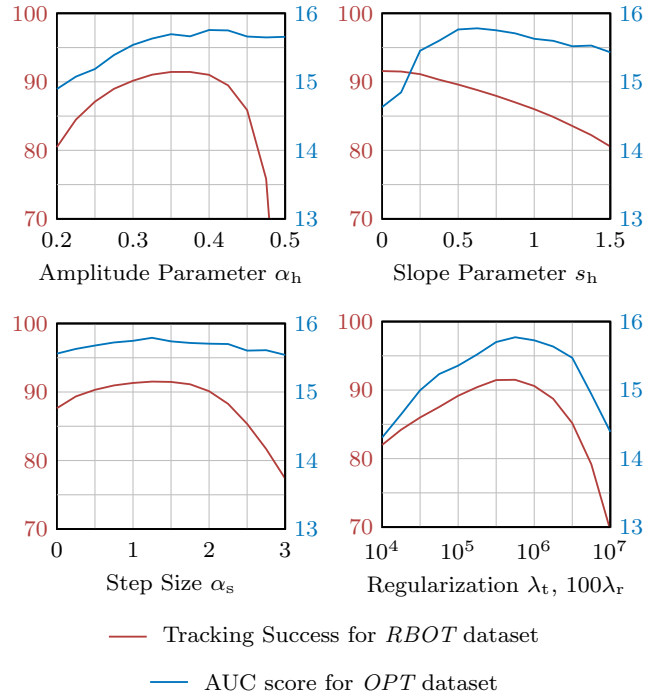

Fig. 13 Average tracking success for the *RBOT* dataset and average AUC score for the *OPT* dataset over different values of the amplitude parameter α_h, slope parameter s_h, step size α_s, and the rotational and translational regularization parameters λ_r and λ_t. For the evaluation of the regularization parameters, we set $\lambda_t = 100\lambda_r$.

compared to none region-based approaches, we are able to report the highest score for four out of six objects and perform best on average. This is even more remarkable since *ORB-SLAM2*, which reports better results for the *house* object, uses gradient-based corner features. In contrast to *SRT3D*, the algorithm is thus not constrained to the contour but considers information over the entire silhouette. Also, the edge-based algorithm of Bugaev et al. (2018), which performs best for the *jet* object, uses basin-hopping for global optimization, and, with an average reported runtime of 683 ms, is not real-time capable. In conclusion, the obtained results demonstrate that the excellent performance of *SRT3D* on simulated data translates well to applications in the real world.

5.3 Parameter Analysis

Having evaluated the performance of our approach, we want to foster our understanding of different parameter values. For this, the average success rate for the *RBOT* dataset and the average AUC score for the *OPT* dataset are plotted over different parameter values. The plots are shown in Fig. 13. Note that the success rate and

AUC score are computed over all objects and sequences. Except for the parameter that is analyzed, the same settings as in Sect. 5.1 and Sect. 5.2 are used.

The evaluation of the amplitude parameter α_h shows that while it significantly influences the tracking success, the effect on the AUC score is much smaller. Knowing that the amplitude parameter models a constant level of noise, this makes sense since the RBOT dataset features highly cluttered images while the OPT dataset only contains a constant white background. For the slope parameter s_h, the highest tracking success is observed for $s_h \to 0$, and the best AUC score is obtained at $s_h = 0.5$. Again, this is well explained by the theoretical interpretation according to which the slope parameter models local uncertainty. Given perfect information about the object geometry for the semi-synthetic RBOT dataset, we do not expect any local uncertainty. At the same time, for the OPT dataset, with imperfectly 3D printed objects and recorded real-world images, it is important that a larger parameter is chosen that allows for a defined level of uncertainty.

Studying the plot of the step size α_s, we observe a relatively large plateau around one, with maximum values at $\alpha_s = 1.3$ for both the tracking success and the AUC score. This suggests a low dependency between the parameter and different image data. Particularly interesting are also the results for $\alpha_s = 0$. For this setting, no local optimization is considered, showing the capability of the global optimization alone. The good results highlight the excellent performance of the adopted global approximation.

Finally, for the evaluation of regularization, the rotational and translational parameters are modified simultaneously. To consider the different units of radians and meters, we define $\lambda_t = 100\lambda_r$. Like in previous evaluations of the soda object, we increase the rotational parameter and use $\lambda_r = \lambda_t$. The resulting plot of the tracking success and the AUC score demonstrates the high importance of regularization. If values are chosen too small, the optimization is unstable for directions in which no or very little information is available. At the same time, if parameters are too large, the optimization is slowed down, and the final pose might not be reached. It is thus important to find values that lie in between. In our experience, a good approximation is to use regularization parameters that are in the same order of magnitude as the maximum rotational and translational diagonal elements of the Hessian matrix.

In conclusion, the parameter analysis demonstrates that theoretical interpretations from Sect. 2 and Sect. 3 correspond well to experimental results. In addition to fostering our understanding, this explainability helps to guide the parameter search for new applications. More-over, the results in Fig. 13 demonstrate that all parameters are well-behaved, with large plateaus around the maximum and no sudden jumps. This has the advantage that parameters are easy to tune, with a broad range of values achieving satisfying results.

5.4 Discussion

The conducted experiments demonstrate the excellent performance of SRT3D. In the following, we want to discuss design considerations that are essential in achieving those results and shed some light on the remaining limitations of the algorithm. With respect to computational efficiency, the biggest performance gain is attributed to the correspondence line model and the sparse nature of the method. In addition, the sparse viewpoint model provides a highly efficient representation, which requires only a simple search to obtain the object contour for the current pose. Also, in contrast to dense methods, it is not necessary to compute a 2D signed distance function, but one can simply use the contour distance. Finally, the discrete scale-space formulation reduces the amount of computation further by combining multiple pixels into segments and supporting the use of precomputed smoothed step functions.

For the quality of the pose estimate, multiple aspects have to be considered. The first important factor is the use of smoothed step functions that provide a realistic modeling of local and global uncertainty. Consequently, this leads to reliable posterior probability distributions. Also, due to the one-dimensionality of correspondence lines and the discrete scale-space implementation, we are able to sample values over posterior probability distributions in reasonable time. This allows us to calculate the mean and the variance. Both estimates constitute the basis for fast-converging global optimization that is independent of local minima. In addition, knowledge about the uncertainty of individual correspondence lines is also used for local optimization, where numerical first-order derivatives are weighted according to the inverse variance. Finally, Tikhonov regularization is another important factor, which helps to constrain the estimate with respect to the previous pose, stabilizing the optimization for directions in which no or very little information is available.

While the described algorithm achieves remarkable results and works very well in a wide variety of applications, some challenges remain. The main limitations are thereby very similar to other region-based methods. The biggest constraint is that objects have to be rigid and that an accurate 3D model has to be known. Also, the background has to be distinguishable from the object. If large areas in the background contain colors

that are also present in the object, the final result might be perturbed. Another challenge comes from ambiguities where the object silhouette is very similar in the vicinity of a particular pose. Naturally, in such cases, there is not enough information, and it is impossible for the algorithm to converge towards the correct pose. Also, like most tracking approaches, the algorithm can only be used for local optimization with a limit to the maximum pose difference from one frame to the next. Finally, if large parts of the object are occluded, the visible part of the contour might not fully constrain the pose of the object, leading to erroneous estimates. To illustrate all the described failure cases, we provide a video on our project site[1].

6 Conclusion

In this work, we proposed *SRT3D*, a highly efficient, sparse approach to region-based 3D object tracking that uses correspondence lines to find the pose that best explains the segmentation of the image. In addition to a thorough mathematical derivation of correspondence lines, a big contribution of this work is the development of smoothed step functions that allow the modeling of both local and global uncertainty. The effects of this modeling were analyzed in detail with respect to both theoretical posterior probability distributions and the quality of the final tracking result. For the maximization of the pose-dependent joint posterior probability, we proposed the use of an initial, global optimization towards the mean and a consecutive, local optimization that considers discrete distribution values. We also developed a novel approximation for the local first-order derivative that weights the finite difference value with the inverse variance. Finally, in multiple experiments on the *RBOT* and the *OPT* dataset, we demonstrated that our algorithm outperforms the current state of the art in region-based tracking by a considerable margin both in terms of quality and efficiency.

Thanks to this superior performance, we are confident that our approach is useful to a wide range of applications in robotics and augmented reality. Because of its general formulation, it is easy to conceive ideas that extend the method. One possible direction would be to include other developments in region-based tracking, such as advanced segmentation models or occlusion detection. Also, it might be useful to consider additional information, like depth or texture. Finally, we want to highlight that the developed correspondence line model is not limited to the context of 3D tracking but might also be useful to other applications. One

possible example is image segmentation. Other methods might thereby show similar progress in terms of quality and efficiency, improving their applicability to the real world.

Appendix A   Extended Probabilistic Model

In the following, we establish the relation between the smoothed step functions proposed in Sect. 2.4 and an extended probabilistic model with $m \in \{m_\mathrm{f}, m_\mathrm{b}, m_\mathrm{n}\}$. For the derivation, we start from an extended definition of the pixel-wise posterior probability

$$p(m_i \mid \boldsymbol{y}) = \frac{p(\boldsymbol{y} \mid m_i)p(m_i)}{\sum_{j \in \{\mathrm{f},\mathrm{b},\mathrm{n}\}} p(\boldsymbol{y} \mid m_j)p(m_j)}, \quad i \in \{\mathrm{f},\mathrm{b},\mathrm{n}\}, \tag{45}$$

where, in contrast to Eq. (4), a noise model m_n is considered in addition to the foreground and background model. Using the parameter $\alpha_\mathrm{h} \in [0, 0.5]$, the model probabilities are defined as

$$p(m_\mathrm{f}) = p(m_\mathrm{b}) = \alpha_\mathrm{h}, \tag{46}$$

$$p(m_\mathrm{n}) = 1 - 2\alpha_\mathrm{h}. \tag{47}$$

For the conditional color probability given the noise model, the conditional probabilities with respect to the foreground and background are combined as follows

$$p(\boldsymbol{y} \mid m_\mathrm{n}) = \frac{1}{2}\big(p(\boldsymbol{y} \mid m_\mathrm{f}) + p(\boldsymbol{y} \mid m_\mathrm{b})\big). \tag{48}$$

Introducing the definitions from Eqs. (46) to (48) into Eq. (45) and performing some simplifications results in the following pixel-wise posterior probabilities for the foreground and background model

$$p(m_i \mid \boldsymbol{y}) = \frac{2\alpha_\mathrm{h}p(\boldsymbol{y} \mid m_i)}{p(\boldsymbol{y} \mid m_\mathrm{f}) + p(\boldsymbol{y} \mid m_\mathrm{b})}, \quad i \in \{\mathrm{f},\mathrm{b}\}. \tag{49}$$

Also, we obtain the following constant pixel-wise posterior probability for the noise model

$$p(m_\mathrm{n} \mid \boldsymbol{y}) = p(m_\mathrm{n}) = 1 - 2\alpha_\mathrm{h}. \tag{50}$$

Based on Eq. (6), the extended posterior probability can be calculated as follows

$$p(d \mid r, \boldsymbol{y}) \propto \sum_{i \in \{\mathrm{f},\mathrm{b},\mathrm{n}\}} p(r \mid d, m_i)p(m_i \mid \boldsymbol{y}). \tag{51}$$

To abbreviate some of the terms in Eq. (49), the following definition of pixel-wise posterior probabilities from Eq. (5) is introduced

$$p_i(r) = \frac{p(\boldsymbol{y} \mid m_i)}{p(\boldsymbol{y} \mid m_\mathrm{f}) + p(\boldsymbol{y} \mid m_\mathrm{b})}, \quad i \in \{\mathrm{f},\mathrm{b}\}. \tag{52}$$

We then use the derived pixel-wise posterior probabilities from Eq. (49) and (50) together with the abbreviation from Eq. (52) to write the posterior probability in Eq. (51) as follows

$$p(d \mid r, \boldsymbol{y}) \propto 2\alpha_{\mathrm{h}} h_{\mathrm{f}}(r-d) p_{\mathrm{f}}(r) + \\ 2\alpha_{\mathrm{h}} h_{\mathrm{b}}(r-d) p_{\mathrm{b}}(r) + \frac{1}{2}(1 - 2\alpha_{\mathrm{h}}), \tag{53}$$

where a constant probability $p(r \mid d, m_{\mathrm{n}}) = \frac{1}{2}$ was used to model the indifference of the line coordinate r given the noise model m_{n}, and where the smoothed step functions h_{f} and h_{b} model the line coordinate probabilities $p(r \mid d, m_{\mathrm{f}})$ and $p(r \mid d, m_{\mathrm{b}})$. For the extension of Eq. (53), we apply the following definitions

$$h_{\mathrm{f}}(x) = \frac{1}{2} - f(x), \tag{54}$$

$$h_{\mathrm{b}}(x) = \frac{1}{2} + f(x), \tag{55}$$

and the identity

$$p_{\mathrm{f}}(r) + p_{\mathrm{b}}(r) = 1, \tag{56}$$

to write

$$p(d \mid r, \boldsymbol{y}) \propto \alpha_{\mathrm{h}} p_{\mathrm{f}}(r) - 2\alpha_{\mathrm{h}} f(r-d) p_{\mathrm{f}}(r) + \\ \alpha_{\mathrm{h}} p_{\mathrm{b}}(r) + 2\alpha_{\mathrm{h}} f(r-d) p_{\mathrm{b}}(r) + \\ \frac{1}{2}\big(p_{\mathrm{f}}(r) + p_{\mathrm{b}}(r)\big) - \alpha_{\mathrm{h}}\big(p_{\mathrm{f}}(r) + p_{\mathrm{b}}(r)\big). \tag{57}$$

This can then be simplified to

$$p(d \mid r, \boldsymbol{y}) \propto \left(\frac{1}{2} - 2\alpha_{\mathrm{h}} f(r-d)\right) p_{\mathrm{f}}(r) + \\ \left(\frac{1}{2} + 2\alpha_{\mathrm{h}} f(r-d)\right) p_{\mathrm{b}}(r). \tag{58}$$

Finally, after introducing the slope function $f(x) = \frac{1}{2}\tanh\left(\frac{x}{2s_{\mathrm{h}}}\right)$ of Stoiber et al. (2020), we obtain

$$p(d \mid r, \boldsymbol{y}) \propto \left(\frac{1}{2} - \alpha_{\mathrm{h}} \tanh\left(\frac{r-d}{2s_{\mathrm{h}}}\right)\right) p_{\mathrm{f}}(r) + \\ \left(\frac{1}{2} + \alpha_{\mathrm{h}} \tanh\left(\frac{r-d}{2s_{\mathrm{h}}}\right)\right) p_{\mathrm{b}}(r). \tag{59}$$

This is the same probability function as the one derived in Sect. 2.4. Note, however, that in Sect. 2.4 the smoothed step functions h_{f} and h_{b} from Eq. (12) and (13) were used instead of a noise model m_{n} to take into account a defined constant uncertainty. In conclusion, this shows that extending the probabilistic model with a noise model m_{n} and using the foreground and background probabilities $p(m_{\mathrm{f}}) = p(m_{\mathrm{b}}) = \alpha_{\mathrm{h}}$ is equivalent to the introduction of a simple amplitude parameter α_{h} into the smoothed step functions.

Appendix B Derivative of Log-Posterior

To analyze the posterior probability distribution, it is desirable to have a closed-form solution that allows an easy interpretation. In the following, we will thus derive a general formulation for the first-order derivative of the log-posterior, which will then be used in Appendix C to calculate the posterior probability distribution for specific parameter configurations. Note that the derivation is similar to the proof of Gaussian equivalence developed in our previous work (Stoiber et al., 2020).

For the derivation, we assume a contour at the line center and perfect step functions for the pixel-wise posterior probabilities defined by

$$p_{\mathrm{f}}(r) = \frac{1}{2} - \frac{1}{2}\,\mathrm{sgn}(r), \tag{60}$$

$$p_{\mathrm{b}}(r) = \frac{1}{2} + \frac{1}{2}\,\mathrm{sgn}(r). \tag{61}$$

Also, we consider infinitesimally small pixels and write the posterior probability distribution from Eq. 14 in continuous form for an infinite correspondence line

$$p(d \mid \omega, \boldsymbol{l}) \propto \prod_{r=-\infty}^{\infty} \big(h_{\mathrm{f}}(r-d) p_{\mathrm{f}}(r) + h_{\mathrm{b}}(r-d) p_{\mathrm{b}}(r)\big)^{\mathrm{d}r}. \tag{62}$$

Starting from those assumptions, we first convert the product integral to the classical Riemann integral

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\left(\int_{r=-\infty}^{\infty} \ln\big(h_{\mathrm{f}}(r-d) p_{\mathrm{f}}(r) + \\ h_{\mathrm{b}}(r-d) p_{\mathrm{b}}(r)\big)\,\mathrm{d}r\right). \tag{63}$$

The integral is then split at $r = 0$, and the pixel-wise posterior probabilities from Eq. (60) and (61) are introduced

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\left(\int_{r=-\infty}^{0} \ln\big(h_{\mathrm{f}}(r-d)\big)\,\mathrm{d}r + \\ \int_{r=0}^{\infty} \ln\big(h_{\mathrm{b}}(r-d)\big)\,\mathrm{d}r\right). \tag{64}$$

Finally, we substitute $x = r - d$ to write

$$p(d \mid \omega, \boldsymbol{l}) \propto \exp\left(\int_{r=-\infty}^{-d} \ln\big(h_{\mathrm{f}}(x)\big)\,\mathrm{d}x + \\ \int_{r=-d}^{\infty} \ln\big(h_{\mathrm{b}}(x)\big)\,\mathrm{d}x\right). \tag{65}$$

The first-order derivative with respect to d of the log-posterior can now be calculated using Leibniz's rule for differentiation under the integral

$$\frac{\partial \ln\big(p(d \mid \omega, \boldsymbol{l})\big)}{\partial d} = -\ln\big(h_{\mathrm{f}}(-d)\big) + \ln\big(h_{\mathrm{b}}(-d)\big). \tag{66}$$

We then adopt the definitions of the smoothed step functions from Eq. (12) and (13) to write

$$\frac{\partial \ln\big(p(d\mid\omega,\boldsymbol{l})\big)}{\partial d} = -\ln\left(\frac{1}{2} - \alpha_{\mathrm{h}}\tanh\left(\frac{-d}{2s_{\mathrm{h}}}\right)\right) + \\ \ln\left(\frac{1}{2} + \alpha_{\mathrm{h}}\tanh\left(\frac{-d}{2s_{\mathrm{h}}}\right)\right). \tag{67}$$

Finally, using the inverse hyperbolic tangent

$$2\tanh^{-1}(x) = -\ln\left(\frac{1}{2} - \frac{x}{2}\right) + \ln\left(\frac{1}{2} + \frac{x}{2}\right), \tag{68}$$

one is able to write the following closed-form expression for the first-order derivative of the log-posterior

$$\frac{\partial \ln\big(p(d\mid\omega,\boldsymbol{l})\big)}{\partial d} = -2\tanh^{-1}\left(2\alpha_{\mathrm{h}}\tanh\left(\frac{d}{2s_{\mathrm{h}}}\right)\right). \tag{69}$$

Appendix C Closed-Form Posteriors

Building on Appendix B, we derive closed-form posterior probability distributions for the two edge cases with either the amplitude parameter $\alpha_{\mathrm{h}} = \frac{1}{2}$ or the slope parameter $s_{\mathrm{h}} \to 0$. We thereby start from the closed-form first-order derivative of the log-posterior that is given in Eq. (69). The full distribution can then be calculated using integration with a subsequent normalization

$$p(d\mid\omega,\boldsymbol{l}) \propto \exp\left(\int \frac{\partial \ln\big(p(d\mid\omega,\boldsymbol{l})\big)}{\partial d}\,\mathrm{d}d\right). \tag{70}$$

For the case with an amplitude parameter $\alpha_{\mathrm{h}} = \frac{1}{2}$, the first-order derivative in Eq. (69) simplifies to

$$\frac{\partial \ln\big(p(d\mid\omega,\boldsymbol{l})\big)}{\partial d} = -\frac{d}{s_{\mathrm{h}}}. \tag{71}$$

Introducing this term in Eq. (70) and calculating the integral leads to the following expression for the posterior probability distribution

$$p(d\mid\omega,\boldsymbol{l}) \propto \exp\left(-\frac{d^2}{2s_{\mathrm{h}}}\right). \tag{72}$$

Because the posterior probability distribution has to be a valid probability density function (PDF) that integrates to one, there is only one possible solution. Knowing that, except for a constant scaling factor, the function looks like a Gaussian distribution, the final solution can only be the Gaussian distribution itself

$$p(d\mid\omega,\boldsymbol{l}) = \frac{1}{\sqrt{2\pi s_{\mathrm{h}}}}\exp\left(-\frac{d^2}{2s_{\mathrm{h}}}\right). \tag{73}$$

For configurations with a slope parameter $s_{\mathrm{h}} \to 0$, the first-order derivative in Eq. (69) simplifies to

$$\frac{\partial \ln\big(p(d\mid\omega,\boldsymbol{l})\big)}{\partial d} = -2\tanh^{-1}(2\alpha_{\mathrm{h}})\,\mathrm{sgn}(d). \tag{74}$$

Introducing this term in Eq. (70) and calculating the integral leads to the following closed-form posterior probability distribution

$$p(d\mid\omega,\boldsymbol{l}) \propto \exp\big(-2\tanh^{-1}(2\alpha_{\mathrm{h}})|d|\big). \tag{75}$$

Similar to the previous derivation, we know that, with the exception of a constant scaling factor, the function is equal to a Laplace distribution, which again is a valid PDF. Introducing the scale parameter b, the final solution can thus only be the Laplace distribution itself

$$p(d\mid\omega,\boldsymbol{l}) = \frac{1}{2b}\exp\left(-\frac{|d|}{b}\right), \quad b = \frac{1}{2\tanh^{-1}(2\alpha_{\mathrm{h}})}. \tag{76}$$

Appendix D Inverse-Variance Weighting

In the following, we demonstrate that the derivatives for the local optimization that were defined in Sect. 3.5 can be derived using inverse-variance weighting and a constant curvature of $\frac{1}{\alpha_{\mathrm{s}}}$ for the second-order derivative. Instead of the joint posterior probability defined in Eq. (29), we start with an energy function that combines probabilities from individual correspondence lines using inverse-variance weighting

$$E(\boldsymbol{\theta}) = \sum_{i=1}^{n_{\mathrm{c}}} \frac{1}{\sigma_i^2}\ln\big(p(d_{\mathrm{s}i}(\boldsymbol{\theta})\mid\omega_{\mathrm{s}i},\boldsymbol{l}_{\mathrm{s}i})\big). \tag{77}$$

Based on this function, the gradient vector and the Hessian matrix are calculated as the first- and second-order derivative with respect to $\boldsymbol{\theta}$

$$\boldsymbol{g}^{\top} = \sum_{i=1}^{n_{\mathrm{c}}} \frac{1}{\sigma_i^2}\frac{\partial \ln\big(p(d_{\mathrm{s}i}\mid\omega_{\mathrm{s}i},\boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}}\frac{\partial d_{\mathrm{s}i}}{\partial\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{0}}, \tag{78}$$

$$\boldsymbol{H} \approx \sum_{i=1}^{n_{\mathrm{c}}} \frac{1}{\sigma_i^2}\frac{\partial^2 \ln\big(p(d_{\mathrm{s}i}\mid\omega_{\mathrm{s}i},\boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}^2} \\ \left(\frac{\partial d_{\mathrm{s}i}}{\partial\boldsymbol{\theta}}\right)^{\top}\left(\frac{\partial d_{\mathrm{s}i}}{\partial\boldsymbol{\theta}}\right)\bigg|_{\boldsymbol{\theta}=\boldsymbol{0}}. \tag{79}$$

For the first-order derivative of the scaled contour distance $d_{\mathrm{s}i}$, the derivations from Eq. (36) and (37) can be used. In contrast to Eq. (40), we use the definition of finite differences without a weighting term to calculate the first-order derivative of the log-posterior

$$\frac{\partial \ln\big(p(d_{\mathrm{s}i}\mid\omega_{\mathrm{s}i},\boldsymbol{l}_{\mathrm{s}i})\big)}{\partial d_{\mathrm{s}i}} \approx \ln\left(\frac{p(d_{\mathrm{s}i}^{+}\mid\omega_{\mathrm{s}i},\boldsymbol{l}_{\mathrm{s}i})}{p(d_{\mathrm{s}i}^{-}\mid\omega_{\mathrm{s}i},\boldsymbol{l}_{\mathrm{s}i})}\right), \tag{80}$$

where d_{si}^- and d_{si}^+ are again the two discrete contour distances that are closest to $d_{si}(\boldsymbol{\theta})$. Because the variance is already considered in the energy function, we simply define a constant curvature for the second-order derivative of the log-posterior

$$\frac{\partial^2 \ln\left(p(d_{si} \mid \omega_{si}, \boldsymbol{l}_{si})\right)}{\partial d_{si}^2} \approx \frac{1}{\alpha_s}. \tag{81}$$

Knowing that constant scaling terms do not affect the Newton optimization, both the gradient vector and the Hessian matrix can be multiplied with the step size α_s. Together with the inverse variance $\frac{1}{\sigma_i^2}$ that is already present in Eq. (78) and (79), this results in exactly the same expressions for the gradient vector and the Hessian matrix as defined in Sect. 3.5. In conclusion, the derivation thus shows that weighting the first-order derivative in Eq. (40) with a factor $\frac{\alpha_s}{\sigma_i^2}$ is the same as using inverse-variance weighting and a constant curvature of $\frac{1}{\alpha_s}$ for the second-order derivative.

References

Besl PJ, McKay ND (1992) A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence 14(2):239–256

Bibby C, Reid I (2008) Robust real-time visual tracking using pixel-wise posteriors. In: European Conference on Computer Vision, pp 831–844

Brachmann E, Michel F, Krull A, Yang MY, Gumhold S, Rother C (2016) Uncertainty-driven 6D pose estimation of objects and scenes from a single RGB image. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 3364–3372

Brox T, Rosenhahn B, Gall J, Cremers D (2010) Combined region and motion-based 3D tracking of rigid and articulated objects. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(3):402–415

Bugaev B, Kryshchenko A, Belov R (2018) Combining 3D model contour energy and keypoints for object tracking. In: European Conference on Computer Vision, pp 55–70

Comport AI, Marchand E, Pressigout M, Chaumette F (2006) Real-time markerless tracking for augmented reality: The virtual visual servoing framework. IEEE Transactions on Visualization and Computer Graphics 12(4):615–628

Crivellaro A, Lepetit V (2014) Robust 3D tracking with descriptor fields. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 3414–3421

Dambreville S, Sandhu R, Yezzi A, Tannenbaum A (2008) Robust 3D pose estimation and efficient 2D region-based segmentation from a 3D shape prior. In:

European Conference on Computer Vision, pp 169–182

Deng X, Mousavian A, Xiang Y, Xia F, Bretl T, Fox D (2021) PoseRBPF: A Rao–Blackwellized particle filter for 6-D object pose tracking. IEEE Transactions on Robotics 37(5):1328–1342

Drummond T, Cipolla R (2002) Real-time visual tracking of complex structures. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7):932–946

Engel J, Koltun V, Cremers D (2018) Direct sparse odometry. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(3):611–625

Garon M, Lalonde JF (2017) Deep 6-DOF tracking. IEEE Transactions on Visualization and Computer Graphics 23(11):2410–2418

Harris C, Stennett C (1990) RAPID - A video rate object tracker. In: Proceedings of the British Machine Vision Conference, pp 15.1–15.6

Hexner J, Hagege RR (2016) 2D-3D pose estimation of heterogeneous objects using a region based approach. International Journal of Computer Vision 118(1):95–112

Hinterstoisser S, Lepetit V, Ilic S, Holzer S, Bradski G, Konolige K, Navab N (2013) Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes. In: Asian Conference on Computer Vision, pp 548–562

Huang H, Zhong F, Sun Y, Qin X (2020) An occlusion-aware edge-based method for monocular 3d object tracking using edge confidence. Computer Graphics Forum 39(7):399–409

Kehl W, Tombari F, Ilic S, Navab N (2017) Real-time 3D model tracking in color and depth on a single CPU core. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 465–473

Krainin M, Henry P, Ren X, Fox D (2011) Manipulator and object tracking for in-hand 3D object modeling. International Journal of Robotics Research 30(11):1311–1327

Krull A, Michel F, Brachmann E, Gumhold S, Ihrke S, Rother C (2015) 6-DOF model based tracking via object coordinate regression. In: Asian Conference on Computer Vision, pp 384–399

Lankton S, Tannenbaum A (2008) Localizing region-based active contours. IEEE Transactions on Image Processing 17(11):2029–2039

Lepetit V, Fua P (2005) Monocular Model-Based 3D Tracking of Rigid Objects: A Survey, vol 1. Foundations and Trends in Computer Graphics and Vision

Leutenegger S, Chli M, Siegwart RY (2011) BRISK: Binary robust invariant scalable keypoints. In: IEEE International Conference on Computer Vision, pp 2548–2555

Li JC, Zhong F, Xu SH, Qin XY (2021) 3D object tracking with adaptively weighted local bundles. Journal of Computer Science and Technology 36(3):555–571

Li Y, Wang G, Ji X, Xiang Y, Fox D (2018) DeepIM: Deep iterative matching for 6D pose estimation. In: European Conference on Computer Vision, pp 695–711

Liu F, Wei Z, Zhang G (2021) An off-board vision system for relative attitude measurement of aircraft. IEEE Transactions on Industrial Electronics

Liu Y, Sun P, Namiki A (2020) Target tracking of moving and rotating object by high-speed monocular active vision. IEEE Sensors Journal 20(12):6727–6744

Lowe DG (2004) Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2):91–110

Lucas BD, Kanade T (1981) An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, vol 2, pp 674–679

Mur-Artal R, Tardós JD (2017) ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. IEEE Transactions on Robotics 33(5):1255–1262

Pauwels K, Rubio L, Díaz J, Ros E (2013) Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In: IEEE Conference on Computer Vision and Pattern Recognition, pp 2347–2354

Prisacariu VA, Reid ID (2012) PWP3D: Real-time segmentation and tracking of 3D objects. International Journal of Computer Vision 98(3):335–354

Prisacariu VA, Kähler O, Murray DW, Reid ID (2015) Real-time 3D tracking and reconstruction on mobile phones. IEEE Transactions on Visualization and Computer Graphics 21(5):557–570

Ren CY, Prisacariu VA, Kähler O, Reid ID, Murray DW (2017) Real-time tracking of single and multiple objects from depth-colour imagery using 3D signed distance functions. International Journal of Computer Vision 124(1):80–95

Rosenhahn B, Brox T, Weickert J (2007) Three-dimensional shape knowledge for joint image segmentation and pose tracking. International Journal of Computer Vision 73(3):243–262

Rublee E, Rabaud V, Konolige K, Bradski G (2011) ORB: An efficient alternative to SIFT or SURF. In: IEEE International Conference on Computer Vision, pp 2564–2571

Sarlin PE, DeTone D, Malisiewicz T, Rabinovich A (2020) SuperGlue: Learning feature matching with graph neural networks. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4937–4946

Schmaltz C, Rosenhahn B, Brox T, Weickert J (2012) Region-based pose tracking with occlusions using 3D models. Machine Vision and Applications 23(3):557–577

Seo B, Park H, Park J, Hinterstoisser S, Ilic S (2014) Optimal local searching for fast and robust textureless 3D object tracking in highly cluttered backgrounds. IEEE Transactions on Visualization and Computer Graphics 20(1):99–110

Seo BK, Wuest H (2016) A direct method for robust model-based 3D object tracking from a monocular RGB image. In: European Conference on Computer Vision Workshop, pp 551–562

Stoiber M, Pfanne M, Strobl KH, Triebel R, Albu-Schaeffer A (2020) A sparse gaussian approach to region-based 6DoF object tracking. In: Asian Conference on Computer Vision, pp 666–682

Sun X, Zhou J, Zhang W, Wang Z, Yu Q (2021) Robust monocular pose tracking of less-distinct objects based on contour-part model. IEEE Transactions on Circuits and Systems for Video Technology

Tan DJ, Navab N, Tombari F (2017) Looking beyond the simple scenarios: Combining learners and optimizers in 3D temporal tracking. IEEE Transactions on Visualization and Computer Graphics 23(11):2399–2409

Tjaden H, Schwanecke U, Schömer E, Cremers D (2018) A region-based Gauss-Newton approach to real-time monocular multiple object tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 41(8):1797–1812

Vacchetti L, Lepetit V, Fua P (2004) Stable real-time 3D tracking using online and offline information. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(10):1385–1391

Wagner D, Reitmayr G, Mulloni A, Drummond T, Schmalstieg D (2010) Real-time detection and tracking for augmented reality on mobile phones. IEEE Transactions on Visualization and Computer Graphics 16(3):355–368

Wang C, Xu D, Zhu Y, Martín-Martín R, Lu C, Fei-Fei L, Savarese S (2019) DenseFusion: 6D object pose estimation by iterative dense fusion. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 3338–3347

Wen B, Mitash C, Ren B, Bekris KE (2020) se(3)-TrackNet: Data-driven 6D pose tracking by calibrating image residuals in synthetic domains. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp 10367–10373

Whelan T, Leutenegger S, Moreno RS, Glocker B, Davison A (2015) ElasticFusion: Dense SLAM without a

pose graph. In: Robotics: Science and Systems

Wu PC, Lee YY, Tseng HY, Ho HI, Yang MH, Chien SY (2017) A benchmark dataset for 6DoF object pose tracking. In: IEEE International Symposium on Mixed and Augmented Reality, pp 186–191

Xiang Y, Schmidt T, Narayanan V, Fox D (2018) PoseCNN: A convolutional neural network for 6D object pose estimation in cluttered scenes. In: Robotics: Science and Systems

Yi KM, Trulls E, Lepetit V, Fua P (2016) LIFT: Learned invariant feature transform. In: European Conference on Computer Vision, pp 467–483

Yilmaz A, Javed O, Shah M (2006) Object tracking: A survey. ACM Computing Surveys 38(4):13

Zhao S, Wang L, Sui W, Wu H, Pan C (2014) 3D object tracking via boundary constrained region-based model. In: IEEE International Conference on Image Processing, pp 486–490

Zhong L, Zhang L (2019) A robust monocular 3D object tracking method combining statistical and photometric constraints. International Journal of Computer Vision 127(8):973–992

Zhong L, Zhang Y, Zhao H, Chang A, Xiang W, Zhang S, Zhang L (2020a) Seeing through the occluders: Robust monocular 6-DOF object pose tracking via model-guided video object segmentation. IEEE Robotics and Automation Letters 5(4):5159–5166

Zhong L, Zhao X, Zhang Y, Zhang S, Zhang L (2020b) Occlusion-aware region-based 3D pose tracking of objects with temporally consistent polar-based local partitioning. IEEE Transactions on Image Processing 29:5065–5078