Deep Active Contours for Real-time 6-DoF Object Tracking

Long Wang[1*] Shen Yan[3,2*] Jianan Zhen[1] Yu Liu[3]
Maojun Zhang[3] Guofeng Zhang[1,2] Xiaowei Zhou[2†]

[1]SenseTime Research [2]Zhejiang University [3]National University of Defense Technology

Input	Operator	Output
$h \times w \times k$	Conv2d 1×1, ReLU6	$h \times w \times (tk)$
$h \times w \times tk$	dwise s=s 3×3, ReLU6	$\frac{h}{s} \times \frac{w}{s} \times (tk)$
$\frac{h}{s} \times \frac{w}{s} \times tk$	Conv2d 1×1, D Linear	$\frac{h}{s} \times \frac{w}{s} \times k'$

Table 1. **The Structure of Bottleneck.** Bottleneck block transforming from k to k' channels, with stride s, and expansion factor t. Please refer to MobileNetV2 [10] for more details.

1. Method Details

1.1. The Architecture of the FPN-Lite Network

This section introduces the FPN-Lite network, which is based on MobileNetV2 [10] as the encoder. Figure 1 illustrates the detailed structure of the network, which comprises a downsample path (left side) and an upsample path (right side). The downsample path adopts the standard architecture of a convolutional network, but utilizes Bottleneck layers from MobileNetV2 to extract feature maps efficiently. The upsample path also follows a conventional convolutional architecture, but fuses features from different levels of the downsample path with skip connections. The Bottleneck layers are applied again to refine features. The final output consists of orange blocks that capture multi-scale information. Table 1 shows the specific configuration of each Bottleneck layer.

1.2. The Calculation of Statistical Information

To enhance the performance of our model, we incorporate statistical information with deep features as the input of our boundary prediction module. The statistical information is obtained by extracting the contour RGB map $\mathbf{I}_k^c$ from a process similar to that used for the contour feature map $\mathbf{F}_k^c$. Specifically, we fix the length of each correspondence line and sample discrete values of $\bar{r}$ from the set $\{-m, \ldots, 0, \ldots, m\}$. This yields a 2D point set $\boldsymbol{l}_i(\bar{r})$, which is then used to extract the contour RGB map $\mathbf{I}_k^c$ by applying $grid\_sample$ from PyTorch. The location $(\bar{r}+m, i)$ corresponds to the RGB value $\mathbf{I}_k(\boldsymbol{l}_i(\bar{r}))$. Based on each pixel on the correspondence line and its corresponding RGB value $\mathbf{y}_i(\bar{r}) = \mathbf{I}_k(\boldsymbol{l}_i(\bar{r}))$, we calculate pixel-wise posteriors using the following formula:

$$p_{ji}(\bar{r}) = \frac{p(\mathbf{y}_i(\bar{r})|m_j)}{p(\mathbf{y}_i(\bar{r})|m_f) + p(\mathbf{y}_i(\bar{r})|m_b)}, \tag{1}$$

where $j \in \{f, b\}$, and m_f and m_b represent the RGB color distributions for the foreground and background regions, respectively. The color probability distributions $p(\mathbf{y}|m_f)$ and $p(\mathbf{y}|m_b)$ are estimated by color histograms. We discretize the RGB color space into 32 equidistant bins along each dimension, giving a total of 32768 values. The statistical foreground probability map is calculated as follows:

$$\mathbf{FG}_k^c(\bar{r}+m, i) = p_{fi}(\bar{r}), \tag{2}$$

where $\bar{r} \in \{-m, \ldots, 0, \ldots, m\}$.

Following RBGT [11], we first compute the statistical probability of the 2D point $\boldsymbol{l}_i(\bar{r})$ being the boundary of the ith correspondence line can be calculate as follows:

$$p(\boldsymbol{\mathcal{D}}_i|\bar{r}) = \prod_{r=\bar{r}-w}^{\bar{r}+w} (h_f(r-\bar{r})p_{fi}(r) + h_b(r-\bar{r})p_{bi}(r)), \tag{3}$$

where w is used to filter the border range, and h_f and h_b are the step functions:

$$\begin{aligned} h_f(x) &= \frac{1}{2} - \alpha_h \mathrm{sign}(x), \\ h_b(x) &= \frac{1}{2} + \alpha_h \mathrm{sign}(x), \end{aligned} \tag{4}$$

where we set α_h to 0.36 and sign operation is then defined as:

$$\mathrm{sign}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} \tag{5}$$

Finally, the statistical boundary probability map is calculated as:

$$\widetilde{\mathbf{B}}_k(\bar{r}+m-w, i) = p(\boldsymbol{\mathcal{D}}_i|\bar{r}), \tag{6}$$

where $\bar{r} \in \{-(m-w), \ldots, 0, \ldots, m-w\}$.

*The first two authors contributed equally. The authors from Zhejiang University are affiliated with the State Key Lab of CAD&CG and ZJU-SenseTime Joint Lab of 3D Vision. †Corresponding author: Xiaowei Zhou.

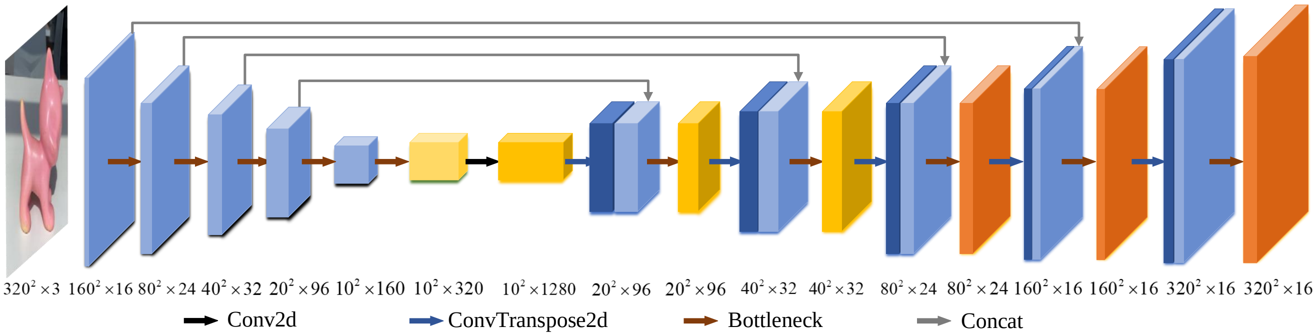

Figure 1. **The Architecture of the FPN-Lite Network.** Each colorful block corresponds to a multi-channel feature map with a specific number of channels and sizes, which are indicated on the edge of each block. The arrows show the different operations performed on the feature maps, such as Conv2d, ConvTranspose2d, Bottleneck or Concat. We use 1×1 Conv2d for the orange blocks, which control the output dimension, to generate the final output feature maps that capture multi-scale information from different levels of the network.

1.3. The Calculation of Derivatives

The Derivation of $\frac{\partial d_i}{\partial \mathbf{X}_{c_i}^{cam}}$ During pose optimization, we need to calculate derivative of the full likelihood $p(\boldsymbol{\mathcal{D}}|\mathbf{P}_k)$ with respect to the pose $\mathbf{P}_k = [\mathbf{R}_k, \mathbf{t}_k]$. We first calculate the projected difference d_i using following formula:

$$\begin{aligned} d_i &= \mathbf{n}_i^\top(\pi(\mathbf{R}_k\mathbf{X}_{c_i} + \mathbf{t}_k) - \mathbf{c}_i) \\ &= \mathbf{n}_i^\top(\pi(\mathbf{X}_{c_i}^{cam}) - \mathbf{c}_i) \end{aligned}, \tag{7}$$

where π is a pinhole camera projection function

$$\pi(\mathbf{X}) = \begin{bmatrix} \frac{X}{Z} f_x + p_x \\ \frac{Y}{Z} f_y + p_y \end{bmatrix}. \tag{8}$$

Then the first-order derivative of d_i with respect to $\mathbf{X}_{c_i}^{cam}$ is calculated as:

$$\begin{aligned} \frac{\partial d_i}{\partial \mathbf{X}_{c_i}^{cam}} &= \begin{bmatrix} n_{x_i} & n_{y_i} \end{bmatrix} \begin{bmatrix} \frac{1}{Z_{c_i}^{cam}} f_x & 0 & -\frac{X_{ci}^{cam}}{(Z_{c_i}^{cam})^2} f_x \\ 0 & \frac{1}{Z_{c_i}^{cam}} f_y & -\frac{Y_{ci}^{cam}}{(Z_{c_i}^{cam})^2} f_y \end{bmatrix} \\ &= \frac{1}{(Z_{c_i}^{cam})^2} [n_{x_i} f_x Z_{c_i}^{cam} \quad n_{y_i} f_y Z_{c_i}^{cam} \\ &\qquad - n_{x_i} f_x X_{c_i}^{cam} - n_{y_i} f_y Y_{c_i}^{cam}]. \end{aligned} \tag{9}$$

The Derivation of $\frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}}$ We add a perturbation to the transformation:

$$\mathbf{X}_{c_i}^{cam} = \mathbf{R}_k(\Delta\mathbf{R}\mathbf{X}_{c_i} + \Delta\mathbf{t}) + \mathbf{t}_k. \tag{10}$$

Since the pose $\mathbf{P}_k$ can be represented by a 6-DoF variation $\boldsymbol{\theta}$, the first order derivative of the 3D point $\mathbf{X}_{c_i}^{cam}$ with respect to the translation is calculated as

$$\left.\frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}^t}\right|_{\boldsymbol{\theta}^t=0} = \frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \Delta\mathbf{t}} = \mathbf{R}_k, \tag{11}$$

Additionally, the first order derivative with respect to each degree of freedom of the rotation is defined as:

$$\begin{aligned} \left.\frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}_x^r}\right|_{\boldsymbol{\theta}_x^r=0} &= \lim_{h\to 0} \frac{\mathbf{R}_k \exp\left(h[\mathbf{1}_x]_\times\right)\mathbf{X}_{c_i} - \mathbf{R}_k\mathbf{X}_{c_i}}{h} \\ &\approx \lim_{h\to 0} \frac{\mathbf{R}_k(\boldsymbol{E}_3 + h[\mathbf{1}_x]_\times)\mathbf{X}_{c_i} - \mathbf{R}_k\mathbf{X}_{c_i}}{h} \\ &= \mathbf{R}_k[\mathbf{1}_x]_\times\mathbf{X}_{c_i}. \end{aligned} \tag{12}$$

Similarly,

$$\begin{aligned} \left.\frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}_y^r}\right|_{\boldsymbol{\theta}_y^r=0} &= \mathbf{R}_k[\mathbf{1}_y]_\times\mathbf{X}_{c_i}, \\ \left.\frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}_z^r}\right|_{\boldsymbol{\theta}_z^r=0} &= \mathbf{R}_k[\mathbf{1}_z]_\times\mathbf{X}_{c_i}, \end{aligned} \tag{13}$$

where $[]_\times$ is the skew-symmetric matrix, and $\mathbf{1}_x$, $\mathbf{1}_y$ and $\mathbf{1}_z$ are defined as:

$$\mathbf{1}_x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \mathbf{1}_y = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \mathbf{1}_z = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{14}$$

Therefore, the first order derivative of the 3D point $X_{c_i}^{cam}$ with respect to the 6-DoF pose is calculated as:

$$\left.\frac{\partial \mathbf{X}_{c_i}^{cam}}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=0} = \mathbf{R}_k[-[\mathbf{X}_{c_i}]_\times \quad \boldsymbol{E}_3], \tag{15}$$

where $\boldsymbol{E}_3$ is the 3×3 identity matrix.

2. Experiment Details

2.1. Training Details

We trained our model on six datasets from the BOP challenge [5]: IC-BIN [1], T-LESS [3], TUD-L [4], LM [2], YCB-V [16], and RU-APC [9]. For training, we selected

Type	Method	Ape	Soda	Vise	Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	EggBox	Glue	Iron	Candy	Lamp	Phone	Squirrel	Avg
Regular	Tjaden et al. [15]	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	97.5	80.7	99.4	79.9
	Zhong et al. [17]	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
	Li et al. [8]	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
	Huang et al. [7]	91.9	44.8	99.7	89.1	89.3	90.6	97.4	95.9	83.9	97.6	91.8	84.4	59.0	92.5	74.3	97.4	86.4	99.7	86.9
	Sun et al. [13]	93.0	55.2	99.3	85.4	96.1	93.9	98.0	95.6	79.5	**98.2**	89.7	89.1	66.5	91.3	60.6	98.6	95.6	99.6	88.1
	Huang et al. [6]	94.6	49.4	99.5	91.0	93.7	96.0	97.8	96.6	90.2	**98.2**	93.4	90.3	64.4	94.0	79.0	**98.8**	92.9	99.8	89.9
	RBGT [11]	96.4	53.2	98.8	93.9	93.0	92.7	99.7	97.1	92.5	92.5	93.7	88.5	70.0	92.1	78.8	95.5	92.5	99.6	90.0
	SRT3D [12]	98.8	65.1	99.6	96.0	98.0	96.5	**100**	98.4	94.1	96.9	98.0	95.3	79.3	96.0	90.3	97.4	96.2	99.8	94.2
	LDT3D [14]	**99.8**	67.1	**100**	**97.8**	97.3	93.7	**100**	**99.4**	**97.4**	97.6	**99.3**	96.9	84.7	**97.7**	93.4	96.7	95.4	**100**	95.2
	DeepAC	98.9	**71.5**	99.4	94.3	**98.2**	**97.6**	99.6	**98.1**	93.0	98.0	95.5	98.1	**93.3**	97.6	**94.3**	96.8	**98.5**	99.4	**95.6**
Dynamic Light	Tjaden et al. [15]	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. [17]	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
	Li et al. [8]	93.5	43.1	96.6	88.5	92.8	86.0	99.6	95.5	85.7	96.8	91.1	90.2	68.4	86.8	59.7	96.1	91.5	99.2	86.7
	Huang et al. [7]	91.8	42.3	98.9	89.9	91.3	87.8	97.6	94.5	84.5	98.1	91.9	86.7	66.2	90.9	73.2	97.1	89.2	99.6	87.3
	Sun et al. [13]	93.8	55.9	99.6	85.6	97.7	93.7	97.7	96.5	78.3	98.6	91.0	91.6	72.1	90.7	63.0	**98.9**	94.4	**100**	88.8
	Huang et al. [6]	94.3	48.3	99.5	90.1	94.6	96.1	97.9	97.3	90.9	**99.1**	92.9	91.5	72.6	94.7	80.0	98.3	95.2	99.8	90.7
	RBGT [11]	96.5	54.6	99.1	93.9	93.1	94.7	99.5	97.0	93.0	93.4	93.3	92.6	74.9	91.0	79.2	95.6	89.8	99.5	90.6
	SRT3D [12]	98.2	65.2	99.2	95.6	97.5	**98.1**	**100**	98.5	94.2	97.5	97.9	96.9	86.1	95.2	89.3	97.0	95.9	99.9	94.6
	LDT3D [14]	**100**	64.5	**99.8**	**97.9**	97.9	94.0	**100**	**99.5**	**97.0**	98.8	**99.3**	97.6	87.5	**97.4**	92.4	97.1	96.4	**100**	95.4
	DeepAC	98.9	**72.1**	**99.8**	93.6	**98.6**	97.6	99.9	98.1	92.6	98.4	94.6	**98.2**	**92.2**	96.8	**94.9**	97.3	**98.6**	99.2	**95.6**
Noisy	Tjaden et al. [15]	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. [17]	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
	Li et al. [8]	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	78.5	69.6	97.6	71.4
	Huang et al. [7]	89.0	60.0	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
	Sun et al. [13]	92.5	56.2	**98.0**	85.1	**91.7**	79.0	97.7	86.2	40.1	**96.6**	90.8	70.2	50.9	84.3	49.9	**91.2**	89.4	99.4	80.5
	Huang et al. [6]	91.0	49.1	95.6	91.0	76.3	54.1	97.1	73.7	27.3	92.8	95.3	30.2	7.8	73.9	56.8	71.4	70.8	98.7	69.6
	RBGT [11]	91.9	53.3	90.2	92.6	67.9	59.3	98.4	80.6	43.5	78.1	92.5	44.0	31.3	72.3	62.0	59.9	71.7	98.3	71.5
	SRT3D [12]	96.9	**61.9**	95.4	95.7	84.5	73.9	99.9	90.3	62.2	87.8	97.6	62.2	43.4	84.3	78.2	73.3	83.1	99.7	81.7
	LDT3D [14]	**99.3**	62.0	95.8	**97.7**	90.4	68.6	**99.9**	91.3	54.2	95.4	**99.0**	64.8	51.6	89.2	75.2	74.7	87.6	**100**	83.2
	DeepAC	94.8	60.6	97.7	93.2	88.8	**90.6**	99.3	**92.6**	**72.1**	93.9	92.3	**83.9**	**70.4**	**91.2**	**83.4**	**91.2**	**89.5**	98.4	**88.0**
Unmodeled Occlusion	Tjaden et al. [15]	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. [17]	83.9	38.1	92.4	81.5	81.3	85.5	97.5	88.9	76.1	87.5	81.7	72.7	52.5	77.2	53.9	88.5	79.3	92.5	78.4
	Li et al. [8]	89.3	43.3	92.2	83.1	84.1	79.0	94.5	88.6	76.2	90.4	87.0	80.7	61.6	75.3	53.1	91.1	81.9	93.4	80.3
	Huang et al. [7]	86.2	46.3	97.8	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
	Sun et al. [13]	91.3	56.7	97.8	82.0	92.8	89.9	96.6	92.2	71.8	97.0	85.0	84.6	66.9	87.7	56.1	95.1	89.8	98.2	85.1
	Huang et al. [6]	92.5	51.5	99.2	90.7	92.1	92.2	97.7	94.2	89.8	**98.4**	91.3	90.7	66.3	91.7	75.3	95.9	92.1	99.0	88.9
	RBGT [11]	90.8	51.7	95.9	88.5	88.0	90.5	96.9	91.6	87.1	90.3	86.4	85.6	65.8	87.0	72.7	91.2	84.0	97.0	85.6
	SRT3D [12]	96.5	66.8	99.0	95.8	95.0	**95.9**	**100**	97.6	92.2	96.6	95.0	94.4	79.0	94.7	89.8	95.7	93.6	99.6	93.2
	LDT3D [14]	**98.7**	68.4	**99.9**	**97.5**	**98.3**	93.0	99.9	**99.4**	**95.1**	97.9	**99.1**	**96.9**	85.5	**97.0**	90.3	**96.3**	95.1	**100**	**94.9**
	DeepAC	96.0	**76.1**	99.4	91.8	97.8	95.3	98.6	96.7	88.6	97.6	93.5	95.9	**88.6**	95.1	**91.0**	95.2	**96.9**	98.0	94.0

Table 2. **Comparison to optimization-based methods on the RBOT benchmark.** We report the tracking successful rate below the threshold of $5cm - 5^{\circ}$.

synthetic slices of obj_1 from IC-BIN, $obj_{1,...,8}$ from T-LESS, $obj_{1,2}$ from TUD-L, $obj_{1,...,10}$ from LM, $obj_{1,...,15}$ from YCB-V and $obj_{1,...,8}$ from RU-APC. We used other objects and their associated slices, both synthetic and real, for validation. During each epoch of training, we randomly sampled 1500 images from each slice, resulting in a total of 69,120 samples per epoch. Since not all of these slices were continuous, we generated an initial pose $\mathbf{P}_k$ by randomly offsetting the rotation by 5-25 degrees and the translation by 5-25 centimeters from the ground truth pose $\mathbf{P}_k^{gt}$. To improve the robustness of DeepAC, we applied image augmentation techniques such as adding Gaussian noise and changing the background. We trained DeepAC for a total of 5 epochs using a batch size of 48. The initial learning rate was set to 1×10^{-3} with a linear learning rate warm-up in 1 epoch, starting from 0.25 of the initial learning rate. The training process took 3 hours using 4 Tesla V100 GPUs.

2.2. More Results on the *RBOT* dataset

Due to the space limitation, we only include the average results of the *RBOT* dataset in the paper. Table 2 presents the results of all objects in *RBOT*.

3. Real-world Examples

We implemented deepAC on mobile devices (iPhone 11) and developed a tracking application. The released version runs at least 25fps. The initial pose is set interactively by aligning the current frame with a standard pose. The user only needs to specify a coarse initial pose, and then our algorithm can converge to the correct pose.

We evaluated our method in real scenes with various challenges such as fast motion, heavy occlusion, and dark light environment. Please refer to the accompanying video for the performance.

References

[1] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *CVPR*, 2016.

[2] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *ACCV*, 2013.

[3] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-LESS: an RGB-D dataset for 6d pose estimation of texture-less objects. In *WACV*, 2017.

[4] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. BOP: benchmark for 6d object pose estimation. In *ECCV*, 2018.

[5] Tomáš Hodaň, Martin Sundermeyer, Bertram Drost, Yann Labbé, Eric Brachmann, Frank Michel, Carsten Rother, and Jiří Matas. BOP challenge 2020 on 6d object localization. In *ECCVW*, 2020.

[6] Hong Huang, Fan Zhong, and Xueying Qin. Pixel-wise weighted region-based 3d object tracking using contour constraints. *TVCG*, 2021.

[7] Hong Huang, Fan Zhong, Yuqing Sun, and Xueying Qin. An occlusion-aware edge-based method for monocular 3d object tracking using edge confidence. *Comput. Graph. Forum.*, 2020.

[8] Jia-Chen Li, Fan Zhong, Song-Hua Xu, and Xue-Ying Qin. 3d object tracking with adaptively weighted local bundles. *JCST*, 2021.

[9] Colin Rennie, Rahul Shome, Kostas E Bekris, and Alberto F De Souza. A dataset for improved rgbd-based object detection and pose estimation for warehouse pick-and-place. *IEEE Robot. Autom. Lett.*, 2016.

[10] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[11] Manuel Stoiber, Martin Pfanne, Klaus H Strobl, Rudolph Triebel, and Alin Albu-Schäffer. A sparse gaussian approach to region-based 6dof object tracking. In *ACCV*, 2020.

[12] Manuel Stoiber, Martin Pfanne, Klaus H Strobl, Rudolph Triebel, and Alin Albu-Schäffer. SRT3D: A sparse region-based 3d object tracking approach for the real world. *IJCV*, 2022.

[13] Xiaoliang Sun, Jiexin Zhou, Wenlong Zhang, Zi Wang, and Qifeng Yu. Robust monocular pose tracking of less-distinct objects based on contour-part model. *IEEE Trans. Circuits. Syst. Video. Technol.*, 2021.

[14] Xuhui Tian, Xinran Lin, Fan Zhong, and Xueying Qin. Large-displacement 3d object tracking with hybrid non-local optimization. In *ECCV*, 2022.

[15] Henning Tjaden, Ulrich Schwanecke, Elmar Schömer, and Daniel Cremers. A region-based gauss-newton approach to real-time monocular multiple object tracking. *T-PAMI*, 2018.

[16] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *RSS*, 2017.

[17] Leisheng Zhong, Xiaolin Zhao, Yu Zhang, Shunli Zhang, and Li Zhang. Occlusion-aware region-based 3d pose tracking of objects with temporally consistent polar-based local partitioning. *TIP*, 2020.