Large-displacement 3D Object Tracking with Hybrid Non-local Optimization

Supplementary Material

1 The Out-of-plane Rotation

1.1 The Decomposition of Rotation

As described in the paper, a rotation matrix $\mathbf{R}$ can be decomposed as in-plane rotation $\mathbf{R}^{in}$ and out-of-plane rotation $\mathbf{R}^{out}$, so that $\mathbf{R} = \mathbf{R}^{in}\mathbf{R}^{out}$. $\mathbf{R}^{out}$ will change the view direction, and $\mathbf{R}^{in}$ is a rotation around the view axis. However, note that this decomposition is not unique, because there is no standard way to change the view direction without introducing in-plane rotation.

We thus adopt a similar approach as the *gluLookAt* function, which can keep an up direction while changing the view direction. To retrieve the view direction $\mathbf{v}$ from $\mathbf{R}$, we should note that in the camera coordinate space, the view axis is always the Z axis. Therefore, $\mathbf{v}$ should be aligned with the Z axis after the rotation, i.e. $\mathbf{R}\mathbf{v} = [0, 0, 1]^\top$. Obviously, $\mathbf{v}$ should be the last row of $\mathbf{R}$.

Given the view direction $\mathbf{v}$, we can compute $\mathbf{R}^{out}$ similarly as *gluLookAt*, by specifying an up vector ($[0, 1, 0]$ in our implementation). We found that this approach can better decompose in-plane and out-of-plane rotations than using Euler angles.

1.2 The Distribution of Failures

As mentioned in the paper, our basic observation is that most failures of previous 3D tracking methods are caused by the out-of-plane rotations. Figure 1 shows the statistics of tracking failures caused by different components (translation, in-plane and out-of-plane rotation) for the representative methods RBOT [7] and SRT3D [5]. As can be seen, about 70% of RBOT failures and 90% of SRT3D failures are due to the out-of-plane rotation. We found that the translation and in-plane rotation introduce more failures for RBOT mainly because the optimization process is not sufficiently converged in order to achieve real-time speed, which results in larger error for the translation and in-plane rotation.

To classify the failures, the rotation error $\mathbf{R}_d = \mathbf{R}\mathbf{R}_{gt}^{-1}$ is first decomposed as in-plane part $\mathbf{R}_d^{in}$ and out-of-plane part $\mathbf{R}_d^{out}$, using the method as described above. The standard 5°-5cm criteria then is applied for the classification. If the translation error is greater than $5cm$, the translation failure is added by one. If the error of $\mathbf{R}_d$ is greater than 5°, the one in $\mathbf{R}_d^{in}$ and $\mathbf{R}_d^{out}$ with larger error is counted as fail.

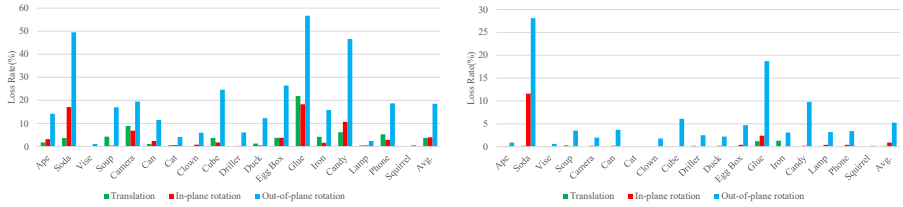


Fig. 1. The distribution of failures with respect to translation, in-plane and out-of-plane rotations on the *regular* variant of RBOT. *left*: The results of RBOT [7]; *right*: The results of SRT3D [5].

2 Pose Optimization

To optimize the pose, the cost function is firstly rewritten as

$$E(\xi) = \sum_{i=1}^{N} \omega_i \parallel F(\xi, i) \parallel^{\alpha}, \quad \text{where } F(\xi, i) = \mathbf{n}_i^{\top}(\mathbf{x}_i^{\xi} - \mathbf{o}_i) - d_i \tag{1}$$

which does not satisfy the general form of least square problem. Therefore, To solve it with *iterative reweighted least square*(IRLS) , we can further rewrite it as

$$E(\xi) = \sum_{i=1}^{N} \omega_i \psi_i F(\xi, i)^2, \quad \text{with } \psi_i = \frac{1}{\parallel F(\xi, i) \parallel^{2-\alpha}} \tag{2}$$

with ψ_i fixed weights computed with the current ξ, which is used to penalize the correspondences that are with large matching residuals. We solve the optimization problem using Gauss-Newton method. By linearizing $F(\xi, i)$ with its first-order Taylor approximation $F(\xi + \Delta\xi) = F(\xi) + \mathbf{J}\Delta\xi$, we have

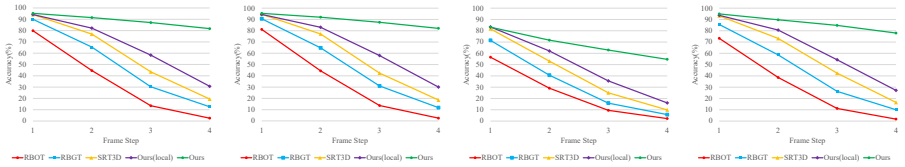
$$E(\xi) \approx \sum_{i=1}^{N} \omega_i \psi_i \left(F(\xi, i) + \mathbf{J}\Delta\xi\right)^2 \tag{3}$$

which is quadratic with respect to $\Delta\xi$, so the optimal solution should be

$$\Delta\xi = -(\sum_{i=1}^{N} \omega_i \psi_i \mathbf{J}^{\top}\mathbf{J})^{-1} \sum_{i=1}^{N} \omega_i \psi_i \mathbf{J} F(\xi, i) \tag{4}$$

with $\mathbf{J}$ the Jacobian of $F(\xi, i)$

$$\mathbf{J} = \frac{\partial F(\xi, i)}{\partial \xi} = n_i^{\top} \frac{\partial \mathbf{x}_i^{\xi}}{\partial \xi} \tag{5}$$



(a) *regular* (b) *dynamic light* (c) *noisy* (d) *unmodeled occ.*

Fig. 2. Results of all variants for different frame steps ($S = 1, 2, 3, 4$). *Ours(local)* is *Ours*$^-$. The compared methods are RBOT [7], RBGT [4], SRT3D [5].

where $\mathbf{x}_i^\xi = \pi(\mathbf{K}(\mathbf{T}\widetilde{\mathbf{X}})_{3\times 1}) = \pi(\mathbf{K}(\mathbf{R}\mathbf{X}+\mathbf{t}))$, therefore

$$\begin{aligned}\frac{\partial \mathbf{x}_i^\xi}{\partial \xi} &= \frac{\partial \pi(\mathbf{K}(\mathbf{R}\mathbf{X}+\mathbf{t}))}{\partial(\mathbf{R}\mathbf{X}+\mathbf{t})}\frac{\partial(\mathbf{R}\mathbf{X}+\mathbf{t})}{\partial \mathbf{X}}\frac{\partial \mathbf{X}}{\partial \xi} \\ &= \begin{bmatrix} \frac{f_x}{Z'} & 0 & -\frac{f_x X'}{(Z')^2} \\ 0 & \frac{f_y}{Z'} & -\frac{f_y Y'}{(Z')^2} \end{bmatrix} \mathbf{R} \begin{bmatrix} 1 & 0 & 0 & 0 & Z & -Y \\ 0 & 1 & 0 & -Z & 0 & X \\ 0 & 0 & 1 & Y & -X & 0 \end{bmatrix}\end{aligned} \tag{6}$$

in which $\mathbf{X}' = [X', Y', Z'] = \mathbf{R}\mathbf{X} + \mathbf{t}$ is the coordinate in the camera space.

After $\Delta\xi$ is computed, the object pose can be updated as

$$\mathbf{T} \leftarrow \mathbf{T}\exp(\Delta\xi) \tag{7}$$

Note that $\exp(\Delta\xi)$ should appear at the right of $\mathbf{T}$, because in Eq. (6) the disturbance is applied in the model space.

3 Results of Other Variants

Due to the space limitation, only the results of the *regular* variant of the RBOT dataset are included in the paper. Here we present results of all variants, including *regular*, *dynamic light*, *noisy*, *unmodeled occlusion*.

Figure 2 shows the results for different frame steps. All variants have similar trends, our local and non-local methods both significantly outperform previous methods. For large displacements, great improvement is achieved with our non-local method. The accuracy of *noisy* is much lower than the accuracy of other three variants, because the severe noise would introduce significant error to the probability map, resulting in errors in the correspondences. However, when compared with previous methods, our method still performs much better for large displacements.

Table 1 contains more details and comparisons for $S = 1$, which is the standard benchmark of the RBOT dataset. Results of the compared methods are from the original paper. As can be seen, for $S = 1$ our method still achieves the best accuracy, slightly better than SRT3D [5] and significantly better than others. Since the displacement is small, using non-local search only slightly improve the accuracy. For the *noisy* variant, using non-local search for $S = 1$ even

decrease the accuracy by 0.2%. As we have analyzed in the paper, this is mainly due to the error function, which may fail to predict the true object pose in complicated cases.

Table 1. Comparisons on the standard RBOT benchmark, with frame step $S = 1$.

Method	Ape	Soda	Vise	Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	EggBox	Glue	Iron	Candy	Lamp	Phone	Squirrel	**Avg.**
										Regular									
[7]	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
[8]	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
[3]	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
[2]	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
[6]	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
[1]	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
[4]	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
[5]	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
Ours$^-$	**99.8**	65.6	99.5	95.0	96.6	92.6	**100**	98.7	95.0	97.1	97.4	96.1	83.3	96.9	91.5	95.8	95.2	99.7	94.2
Ours	**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**
										Dynamic Light									
[7]	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
[8]	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
[3]	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
[2]	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
[6]	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
[1]	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
[4]	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
[5]	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
Ours$^-$	99.7	64.7	99.7	95.2	97.2	93.0	99.8	98.8	94.2	98.4	97.1	97.2	86.9	95.4	91.4	96.2	95.6	99.9	94.5
Ours	**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**
										Noisy									
[7]	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
[8]	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
[3]	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
[2]	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
[6]	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
[1]	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
[4]	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
[5]	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
Ours$^-$	98.4	**62.6**	95.5	94.0	90.6	69.6	99.8	89.1	57.1	95.7	97.2	68.7	**53.2**	87.9	76.1	76.9	89.0	99.2	**83.4**
Ours	**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
										Unmodeled Occlusion									
[7]	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
[8]	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
[3]	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
[2]	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
[6]	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
[1]	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
[4]	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
[5]	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
Ours$^-$	98.0	65.8	98.7	95.2	97.0	92.0	99.8	98.8	92.6	97.3	97.1	95.7	83.8	95.2	89.3	96.0	94.3	99.6	93.7
Ours	**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**

4 More Results and Analysis

Figure 4 provides more visual results with very large frame step ($S = 8$), in which case it is nearly impossible to be successfully tracked with local methods such as SRT3D and *Ours*$^-$. Our non-local method performs much better. Note that in order to achieve real-time speed, we adopted a very simple color-based segmentation method. As can be seen from Figure 4, the probability map is very inaccurate for some objects, introducing many resistant local minimum. Without a doubt, if more computing power (e.g. GPU) is available, better segmentation

Table 2. Tracking success rates with different α on the RBOT dataset [7].

	Ape	Soda	Vise	Soup	Camera	Can	Cat	Clown	Cube	Driller	Duck	EggBox	Glue	Iron	Candy	Lamp	Phone	Squirrel	**Avg.**
$\alpha = 0.125$	99.8	65.6	99.5	95.0	96.6	92.6	100	98.7	95.0	97.1	97.4	96.1	83.3	96.9	91.5	95.8	95.2	99.7	94.2
$\alpha = 0.25$	99.7	65.2	99.6	95.1	96.3	92.1	100	98.7	94.8	97.1	97.2	96.1	83.5	97.1	91.7	95.9	94.8	99.9	94.2
$\alpha = 0.5$	99.3	65.6	99.4	95.2	95.5	90.1	100	98.4	94.6	96.5	97.3	94.2	82.8	97.0	90.7	95.2	92.7	99.8	93.6
$\alpha = 0.75$	99.3	65.1	99.9	95.6	92.8	86.9	100	98.3	94.3	94.8	97.5	91.1	76.9	96.1	90.1	91.6	88.3	99.9	92.1
$\alpha = 1.0$	99.3	63.8	98.9	95.5	86.1	79.7	99.9	98.3	93.9	87.6	97.2	85.1	67.9	94.7	87.3	83.4	80.0	99.5	88.8
$\alpha = 1.5$	98.9	61.2	84.7	95.1	54.2	42.2	99.3	96.5	87.7	56.2	97.4	48.7	38.4	85.6	61.8	46.0	48.4	99.1	72.3

(a) #1 input (b) #1 user specified (c) #2 result

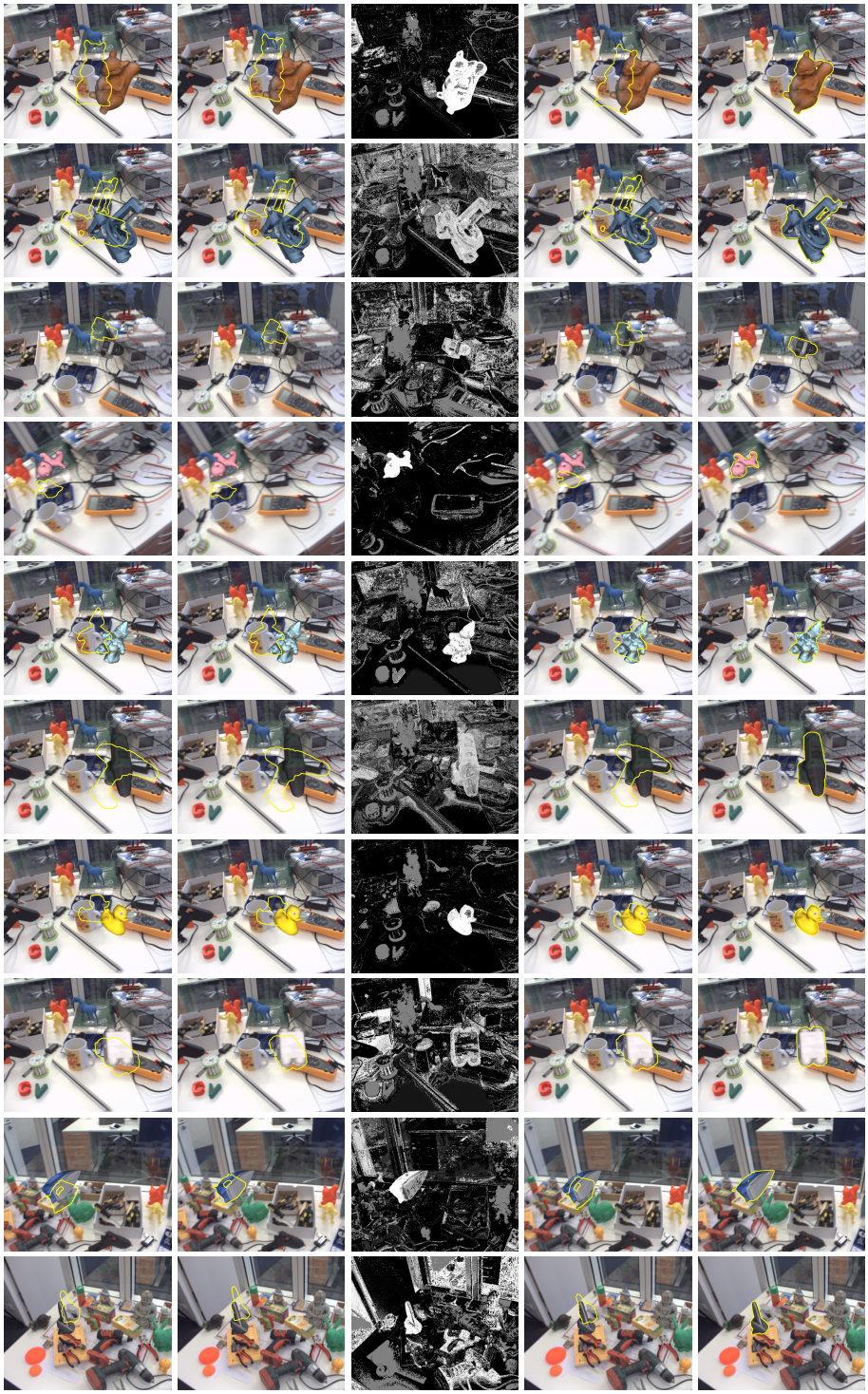
Fig. 3. The interactive initialization process in real scenes.

and larger search range can be achieved in real time, then the accuracy of our method can be further improved.

The ablation study in the paper shows that using smaller α is helpful for achieving higher accuracy. Table 2 contains details of each object. We can find that as the increase of α, the decrease of accuracy for each object is very different. For the objects that are distinctive from the background (*ape*,*cat*,*squirrel*,*duck* etc.), the accuracy is only slightly decreased. On the other hand, for the objects that are indistinctive (*camera*,*can*,*glue*,*lamp*, etc.), the accuracy would decrease dramatically. Therefore, the robust estimation is especially important for handling errors in segmentation.

5 Real-scene Examples

We tested our method in real scenes for further evaluation. The input video is captured by an ordinary web camera (Logitech C270), with 640 × 480 image resolution and 25fps frame rate. The initial pose is set interactively by aligning the current frame with a standard pose. As shown in Figure 3, the user needs only to specify a rough initial pose, then in the second frame our method can converge to the correct pose. The test scenes contain typical challenges such as fast motion, background clutter, occlusion and out of view, etc. Thanks to the advantage in handling large displacements, our method can perform significant better than previous methods for the case of fast motion. Please see the accompany video for the results.



(a) init. (b) srt3d [5] (c) ours prob. (d) ours(local) (e) ours

Fig. 4. Some visual examples of large displacements($S = 8$).

References

1. Huang, H., Zhong, F., Qin, X.: Pixel-Wise Weighted Region-Based 3D Object Tracking using Contour Constraints. IEEE Transactions on Visualization and Computer Graphics pp. 1–1 (2021). https://doi.org/10.1109/TVCG.2021.3085197
2. Huang, H., Zhong, F., Sun, Y., Qin, X.: An Occlusion-aware Edge-Based Method for Monocular 3D Object Tracking using Edge Confidence. Computer Graphics Forum **39**(7), 399–409 (Oct 2020). https://doi.org/10.1111/cgf.14154
3. Jiachen Li, Fan Zhong, S.X.X.Q.: 3d object tracking with adaptively weighted local bundles. Journal of Computer Science and Technology **36**(3), 555 (2021). https://doi.org/10.1007/s11390-021-1272-5, https://jcst.ict.ac.cn/EN/abstract/article_2756.shtml
4. Stoiber, M., Pfanne, M., Strobl, K.H., Triebel, R., Albu-Schaeffer, A.: A sparse gaussian approach to region-based 6dof object tracking. In: Proceedings of the Asian Conference on Computer Vision (2020)
5. Stoiber, M., Pfanne, M., Strobl, K.H., Triebel, R., Albu-Schäffer, A.: Srt3d: A sparse region-based 3d object tracking approach for the real world (2021)
6. Sun, X., Zhou, J., Zhang, W., Wang, Z., Yu, Q.: Robust monocular pose tracking of less-distinct objects based on contour-part model. IEEE Transactions on Circuits and Systems for Video Technology **31**(11), 4409–4421 (2021). https://doi.org/10.1109/TCSVT.2021.3053696
7. Tjaden, H., Schwanecke, U., Schomer, E., Cremers, D.: 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 (Aug 2019). https://doi.org/10.1109/TPAMI.2018.2884990
8. Zhong, L., Zhao, X., Zhang, Y., Zhang, S., Zhang, L.: Occlusion-Aware Region-Based 3D Pose Tracking of Objects With Temporally Consistent Polar-Based Local Partitioning. IEEE Transactions on Image Processing **29**, 5065–5078 (2020). https://doi.org/10.1109/TIP.2020.2973512