# 6DOPE-GS: Online 6D Object Pose Estimation using Gaussian Splatting

Supplementary Material

## 1. Joint Optimization of 2D Gaussians and Keyframe Poses

Once we obtain an initial set of coarse keyframe poses, we jointly optimize the 2D Gaussians and the keyframe poses based on the photometric and structural losses used in in [2]. These losses include a color consistency loss $\mathcal{L}_c$, a depth consistency loss $\mathcal{L}_d$, a depth distortion loss $\mathcal{L}_{dd}$, and a normal alignment loss $\mathcal{L}_n$. The losses are computed between the rendered and observed images within the pixels covered in the object segmentation mask $\mathcal{M}$. To further reduce camera depth noise, we additionally use the Huber loss.

The color consistency loss $\mathcal{L}_c$ and the depth consistency loss $\mathcal{L}_d$ is the $L_1$ loss between the observed and the rendered images. The losses are calculated after projecting the Gaussians onto the image frame. Specifically, the Gaussians are ordered according to their ascending z-depth in the camera frame after which the contribution of each Gaussian to the loss is weighted based on their opacity and Gaussian density. This process is called $\alpha$-blending. We represent the blending weight of each of the $N$ ordered Gaussians as

$$\omega_i(\boldsymbol{p}) = \alpha_i G_i^{2D}(\boldsymbol{p}) \prod_{j=1}^{i-1}(1-\alpha_j G_j^{2D}(\boldsymbol{p})) \quad ; \quad \forall i \in N \tag{1}$$

We then calculate the color consistency loss $\mathcal{L}_c$ and the depth consistency loss $\mathcal{L}_d$ as

$$\mathcal{L}_c = \frac{1}{|\mathcal{M}|}\sum_{\boldsymbol{p}\in\mathcal{M}} |\hat{c}(\boldsymbol{p}) - c(\boldsymbol{p})| \tag{2}$$

$$\mathcal{L}_d = \frac{1}{|\mathcal{M}|}\sum_{\boldsymbol{p}\in\mathcal{M}} \rho(|\hat{d}(\boldsymbol{p}) - d(\boldsymbol{p})|) \tag{3}$$

where $\rho(\cdot)$ is the Huber loss, $c(\boldsymbol{p})$ and $d(\boldsymbol{p})$ are the observed color and depth, and $\hat{c}(\boldsymbol{p}) = \sum_{i\in N} c_i\omega_i(\boldsymbol{p})$ and $\hat{d}(\boldsymbol{p}) = \sum_{i\in N} d_i\omega_i(\boldsymbol{p})$ are the rendered color and depth at a pixel $\boldsymbol{p}$.

The depth distortion loss clusters the 2D Gaussians along the ray path, minimizing gaps between intersected Gaussians and enhancing depth accuracy.

$$\mathcal{L}_{dd} = \sum_{i,j\in N} \omega_i\omega_j|z_i - z_j| \tag{4}$$

where $\omega_i$ is the blending weight for the $i^{th}$ Gaussian intersection, and $z_i$ denotes the depth of each intersection point. Adjusting the intersection depth $z_i$ to encourage the concentration of splats along the ray.

The normal loss $\mathcal{L}_\text{n}$ further refines object shape by aligning each Gaussian's normal with the local surface gradient.

$$\mathcal{L}_n = \sum_{\boldsymbol{p}\in\mathcal{M}} 1 - \hat{\boldsymbol{n}}(\boldsymbol{p})^\top \boldsymbol{n}(\boldsymbol{p}) \tag{5}$$

where $\hat{\boldsymbol{n}}(\boldsymbol{p})$ and $\boldsymbol{n}(\boldsymbol{p})$ are the normals at pixel $\boldsymbol{p}$ estimated by the gradient of the rendered and observed depth images respectively. By aligning the splat normal with the estimated normal, we ensure that the 2D splats accurately approximate the local object surface.

For refining the keyframe poses, a learnable affine transformation $\boldsymbol{T}_k \in SE(3)$ of the k$^{th}$ keyframe is applied to the 2D Gaussians similar to [3, 7]. The transformation $\boldsymbol{T}_k$ is represented by a rotation $\boldsymbol{R}_k \in SO(3)$ and a translation $t_k \in \mathbb{R}^3$. We learn the transformations along with the Gaussians by minimizing the above-mentioned losses obtained by projecting each Gaussian $G_i \in \mathcal{G}$ onto each selected keyframe pose $\boldsymbol{T}_k \in \mathcal{K}$ as

$$\begin{aligned}\mathcal{G}^*, \mathcal{K}^* = \arg\min_{\mathcal{G},\mathcal{K}} \sum_{k\in|\mathcal{K}|} & \lambda_c\mathcal{L}_c(\hat{I}(\boldsymbol{T}_k \odot \mathcal{G}), I_k) \\ & +\lambda_d\mathcal{L}_d(\hat{D}(\boldsymbol{T}_k \odot \mathcal{G}), D_k) \\ & +\lambda_{dd}\mathcal{L}_{dd} + \lambda_n\mathcal{L}_n\end{aligned} \tag{6}$$

where $\hat{I}$ and $\hat{D}$ are the rendered color and depth images obtained by projecting the Gaussians $\mathcal{G}$ to the $k^{th}$ keyframe pose $\boldsymbol{T}_k$ and $I_k$, $D_k$ represent the observed color and depth images of the $k^{th}$ keyframe. $\lambda_c$, $\lambda_d$, $\lambda_{dd}$ and $\lambda_n$ are the relative weights for the color, depth, depth distortion and normal losses respectively.

## 2. Metrics

To evaluate 6-DoF object pose estimation, we calculate the Area Under Curve (AUC) percentage based on the ADD and ADD-S metrics. The ADD metric determines the average Euclidean distance between corresponding points on the 3D object model after transformation by the predicted and ground truth poses.

$$\text{ADD} = \frac{1}{N}\sum_{i=1}^{N} \|(Rx_i + t) - (R_\text{gt}x_i + t_\text{gt})\|, \tag{7}$$

where $N$ represents the number of points in the 3D object model, $x_i$ denotes a point on the model, $R$ and $t$ are the predicted rotation matrix and translation vector, and $R_\text{gt}$ and $t_\text{gt}$ are their ground truth counterparts. A lower ADD value indicates a more accurate pose estimation. The estimation is considered successful if the ADD is within a specific threshold. For symmetric objects, where distinct poses may appear identical (e.g., a cylindrical object rotated by 180°), the ADD-S metric is more appropriate. Instead of directly pairing corresponding points, ADD-S measures the average distance between a transformed model point and its nearest neighbor in the ground truth-transformed model.

$$\text{ADD-S} = \frac{1}{N}\sum_{i=1}^{N} \min_{x_j \in \mathcal{M}} \|(Rx_i + t) - (R_{\text{gt}}x_j + t_{\text{gt}})\|, \quad (8)$$

where $x_j \in \mathcal{M}$ denotes the set of all 3D points on the object model. This formulation accounts for the ambiguities inherent to symmetric objects, providing a more robust evaluation of pose estimation.

We assess 3D shape reconstruction performance by calculating the chamfer distance between the reconstructed and ground-truth points, adopting the symmetric formulation.

$$\text{CD}(P, Q) = \frac{1}{|P|}\sum_{p\in P} \min_{q\in Q} \|p - q\|^2 + \frac{1}{|Q|}\sum_{q\in Q} \min_{p\in P} \|q - p\|^2 \quad (9)$$

To extract meshes from reconstructed 2D splats, we render depth maps of the training views by projecting the depth values of the splats onto the pixels. Truncated Signed Distance Fusion (TSDF) is then used to fuse the reconstructed depth maps, implemented using Open3D [8]. During TSDF fusion, we set the voxel size to 0.002 and the truncation threshold to 0.02. Additionally, we extend BundleSDF to render depth maps and apply the same surface reconstruction technique to ensure a fair comparison.

## 3. Implementation Details

For the object masks in the YCBInEOAT dataset, we utilize the original masks provided in the dataset. For the HO3D dataset, we use the masks extracted XMem [1], as done in BundleSDF [6]. In real-time scenarios, we manually input the initial object location as a prompt for a random frame from the camera. Based on the prompted image, SAM2 [4] segments the target object across video frames in real-time.

During coarse pose estimation, a new frame is designated as a keyframe if it has more than 10 feature correspondences with the frames in memory. For online graph optimization, we retain the configuration of BundleSDF [6], restricting the number of frames involved in pose graph optimization to a maximum of 10.

For the Gaussian object field, we transform all graph-optimized camera poses into the OpenGL camera representation, which is utilized for Gaussian splatting. OpenGL adopts a right-handed coordinate system with the camera facing the negative z-axis, whereas OpenCV uses a right-handed system with the camera facing the positive z-axis. Using the information from the first frame, we estimate the object size and rescale and translate all camera poses and point clouds to ensure the generated Gaussians fit within a canonical space ranging from -1 to 1. Starting from the 10th keyframe, we optimize the Gaussian object. We begin by fusing color point clouds from the keyframes, followed by downsampling with a voxel size of 0.01 m. We then cluster the points with a maximum distance of 0.06m between points to remove outlier points. Subsequently, we uniformly upsample the point cloud until the number of points exceeds 5000.

We initialize the Gaussian means with the point cloud positions and random rotations. To simplify training and mitigate floating artifacts caused by oversized splats, Gaussian scales are clipped between 0.005 and 0.01. We use Spherical harmonics to represent the color, beginning at level 0 and incrementing every 200 steps, up to level 2. We initialize the opacity of each Gaussian with 0.1. Pose gradients for each keyframe are initialized with six parameters, (3 for translations and 3 for the rotation in an axis-angle representation). Before optimization, we group the keyframe poses using different anchors based on a default setting of an icosahedron at level 1, yielding a total of 42 anchors. Keyframes with the largest masks in each anchor cluster are selected for joint optimization, which runs for 1000 steps.

After 500 steps, the opacity percentile-based Adaptive Density Control is applied every 100 steps. Gaussians with opacity values in the bottom 5th percentile are removed until the 95th percentile opacity exceeds 0.5. Similarly, every 100 steps, keyframes with reconstruction losses exceeding twice the median absolute deviation (MAD) are removed.

In the training loss, $\lambda_c$ = 0.5, $\lambda_d$ = 0.5, $\lambda_{dd}$ = 0.05, $\lambda_n$ = 0.05. To mitigate depth noise, we apply morphological erosion to the depth using a kernel size of 5. The Adam optimizer is utilized for optimizing both the pose and Gaussian parameters, with a decay rate set to 0.5. The initial learning rate for Gaussians is consistent with the 2DGS [2] configuration. After the joint optimization is run for 1000 steps, the Gaussians' attributes are fixed, and only the keyframe poses are further refined for 500 additional steps.

All experiments were performed on a standard desktop equipped with an AMD Ryzen 9 7950X3D 16-core Processor and a single NVIDIA RTX 4090 GPU. Only the temporal efficiency experiments in Sec. **??** used a PC with a different configuration. Our method utilizes two concurrently running threads. The online tracking thread processes frames at approximately 4.1 Hz, while the Gaussian object field thread operates in the background, requiring an average of 0.23 seconds per keyframe.

## 4. Limitations

Gaussian rasterization rendering is highly efficient and allows for the rapid correction of minor translation and in-plane rotation errors. However, it is less effective in gradient computation compared to the differentiable ray casting employed by neural radiance fields. This limitation stems from

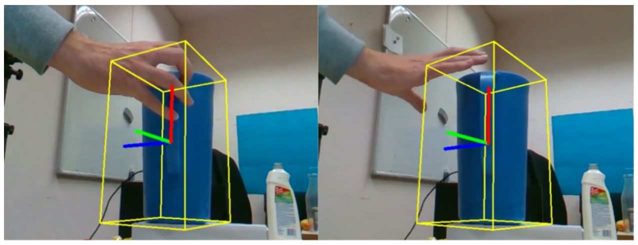

Figure 1. Rotation angle estimation from a side view is often inaccurate for symmetric objects without texture information, as exemplified by the AP10 object from the HO3D dataset.

the fact that pose gradients in Gaussian rasterization are approximated using the covariance matrix projected onto the 2D plane. As a result, gradient-based optimization for non-in-plane rotations or substantial pose corrections becomes problematic. For example, our method struggles to resolve rotational errors around the symmetric axis of a water pitcher, as shown in Fig 1.

## 5. Additional Ablation

A visual comparison of the different ablations in reconstructing a water pitcher from HO3D can be seen in Fig. 2. This example is particularly difficult as there are large rotational motions and large occlusions. In this example, we find that neither of the ablations can accurately reconstruct the pitcher, especially the handle. Rather, they end up with two partially reconstructed handles as a result of inaccurate keyframe estimates which cause points to be added from a new keyframe with the handle visible. However, by combining the dynamic keyframe selection and the opacity percentile-based adaptive density control, our final approach reconstructs the handle more accurately.

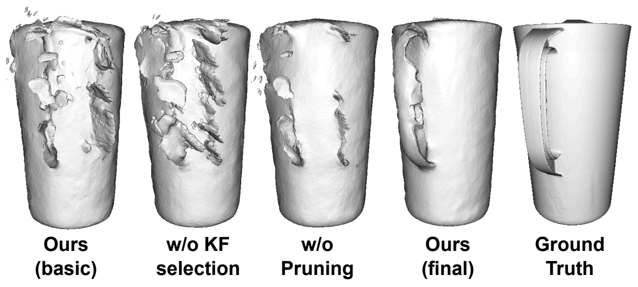

Figure 2. Object reconstruction example using the different ablations over our approach.

## References

[1] Ho Kei Cheng and Alexander G. Schwing. XMem: Long-Term Video Object Segmentation with an Atkinson-Shiffrin Memory Model. 2

[2] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for Geometrically Accurate Radiance Fields. 1, 2

[3] Hidenobu Matsuki, Riku Murai, Paul H. J. Kelly, and Andrew J. Davison. Gaussian Splatting SLAM. 1

[4] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. 2

[5] Bowen Wen and Kostas Bekris. BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8067–8074. 5, 6

[6] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Muller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. BundleSDF: Neural 6-DoF Tracking and 3D Reconstruction of Unknown Objects. 2, 5, 6

[7] Vladimir Yugay, Yue Li, Theo Gevers, and Martin R. Oswald. Gaussian-SLAM: Photo-realistic Dense SLAM with Gaussian Splatting. 1

[8] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018. 2

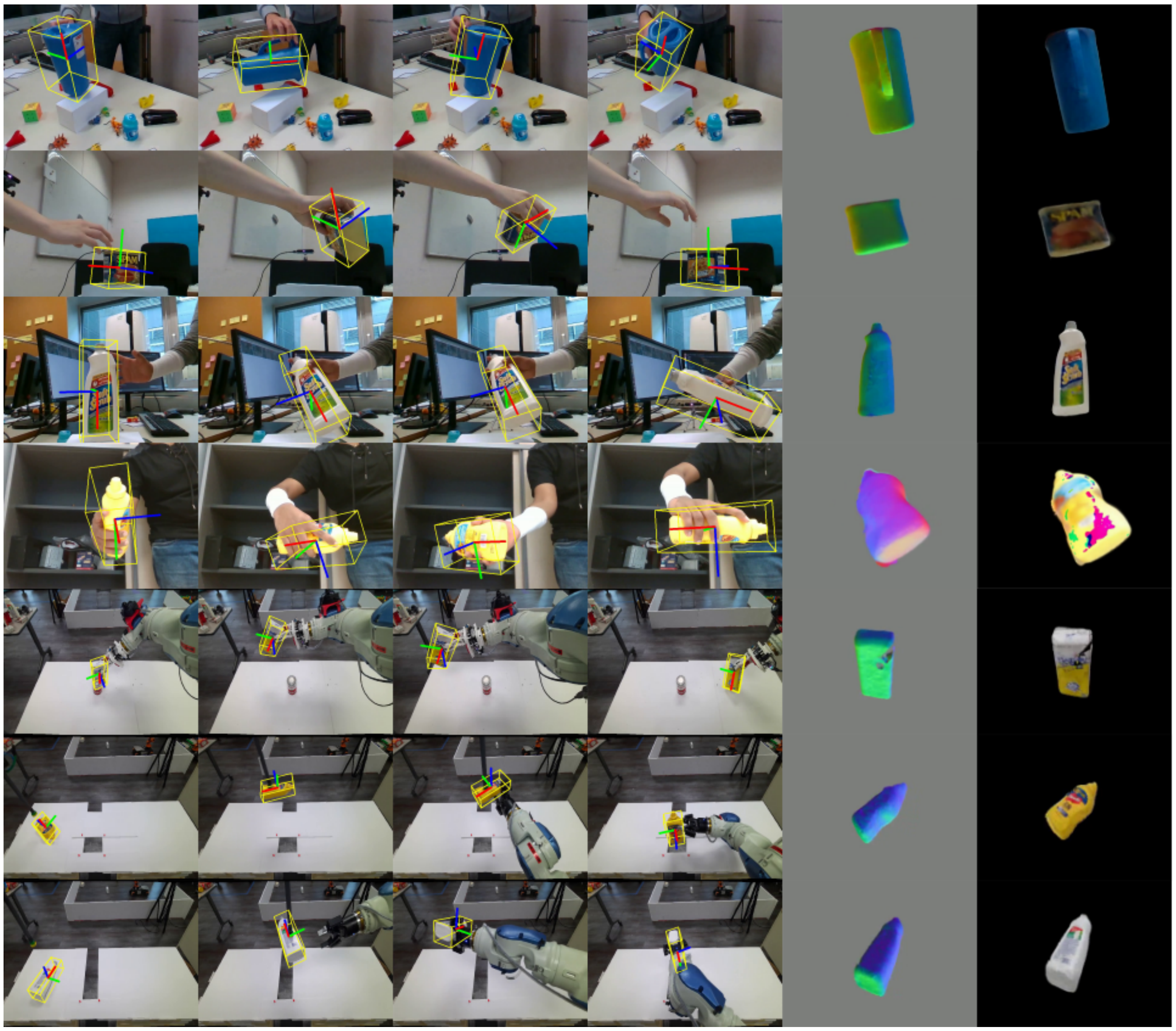

Figure 3. Qualitative results of our method on video sequences from the HO3D and YCBInEOAT datasets

Table 1. Comparison of ADD-S, ADD, and CD metrics, along with the Average Time Per Frame, across different methods on the HO3D dataset. ↑ indicates higher values are better, ↓ indicates lower values are better. The results in the first 2 columns are taken from [6]. * highlights the results reproduced using the open-source code from the authors of [6].

| Video | Metric | BundleTrack [5] | BundleSDF [6] | BundleTrack* | BundleSDF* | BundleSDF-async* | BundleSDF-Lite | Ours |
|---|---|---|---|---|---|---|---|---|
| AP10 | ADD-S(%)↑ | 91.68 | 96.1 | 91.24 | **95.83** | 91.78 | 93.03 | 95.34 |
| | ADD(%)↑ | 36.60 | 91 | 47.98 | **89.66** | 66.38 | 78.66 | 85.88 |
| | CD(cm)↓ | 1.88 | 0.47 | - | 0.13 | 0.55 | 0.76 | **0.20** |
| | ATPF(s)↓ | - | - | 0.27 | 1.55 | 0.26 | 0.39 | **0.23** |
| AP11 | ADD-S(%)↑ | 91.45 | 96.18 | 94.14 | 96.01 | 95.47 | 93.34 | **96.92** |
| | ADD(%)↑ | 41.28 | 91.76 | 84.53 | 90.91 | 89.32 | 81.05 | **93.78** |
| | CD(cm)↓ | 129.18 | 0.56 | - | 0.10 | 0.13 | 0.65 | **0.06** |
| | ATPF(s)↓ | - | - | 0.28 | 1.46 | 0.26 | 0.38 | **0.23** |
| AP12 | ADD-S(%)↑ | 90.79 | 97.06 | 95.42 | **96.98** | 96.51 | 93.87 | 96.77 |
| | ADD(%)↑ | 50.82 | 94.76 | 88.09 | **94.53** | 92.32 | 83.34 | 93.35 |
| | CD(cm)↓ | 2.47 | 0.59 | - | **0.04** | 0.09 | 0.74 | 0.06 |
| | ATPF(s)↓ | - | - | 0.29 | 1.40 | 0.28 | 0.41 | **0.25** |
| AP13 | ADD-S(%)↑ | 90.68 | 96.16 | 95.65 | 96.19 | 95.96 | 92.66 | **96.46** |
| | ADD(%)↑ | 49.03 | 92.73 | 89.95 | 92.77 | 92.11 | 80.14 | **92.90** |
| | CD(cm)↓ | 2.77 | 0.63 | - | **0.03** | 0.05 | 0.78 | 0.06 |
| | ATPF(s)↓ | - | - | 0.28 | 1.40 | 0.29 | 0.41 | **0.24** |
| AP14 | ADD-S(%)↑ | 96.02 | 96.01 | 96.09 | **97.09** | 96.89 | 96.50 | 95.84 |
| | ADD(%)↑ | 90.30 | 91.25 | 91.07 | **94.65** | 94.11 | 92.23 | 91.56 |
| | CD(cm)↓ | 72.40 | 1.28 | - | **0.03** | 0.05 | 0.71 | 0.07 |
| | ATPF(s)↓ | - | - | 0.29 | 1.46 | 0.29 | 0.41 | **0.25** |
| MPM10 | ADD-S(%)↑ | 94.94 | 95.05 | 93.01 | **95.28** | 93.79 | 93.31 | 94.29 |
| | ADD(%)↑ | 87.45 | 88.92 | 75.64 | **89.48** | 83.75 | 83.78 | 84.55 |
| | CD(cm)↓ | 0.97 | 0.56 | - | **0.13** | 0.29 | 0.57 | 0.33 |
| | ATPF(s)↓ | - | - | 0.29 | 2.39 | 0.29 | 0.44 | **0.23** |
| MPM11 | ADD-S(%)↑ | 89.94 | 96.2 | 96.06 | **96.19** | 96.53 | 94.75 | 96.09 |
| | ADD(%)↑ | 53.20 | 91.51 | 91.07 | 91.51 | 92.33 | 88.44 | **91.91** |
| | CD(cm)↓ | 88.97 | 0.49 | - | 0.09 | **0.09** | 0.53 | 0.12 |
| | ATPF(s)↓ | - | - | 0.28 | 2.17 | 0.28 | 0.44 | **0.27** |
| MPM12 | ADD-S(%)↑ | 95.66 | 96.98 | 97.88 | 96.17 | 97.18 | 80.20 | **97.76** |
| | ADD(%)↑ | 90.96 | 93.13 | 95.12 | 91.36 | 93.64 | 60.98 | **95.48** |
| | CD(cm)↓ | 121.33 | 0.46 | - | 0.08 | **0.06** | 1.30 | 0.07 |
| | ATPF(s)↓ | - | - | 0.34 | 1.93 | 0.33 | 0.49 | **0.22** |
| MPM13 | ADD-S(%)↑ | 89.42 | 95.8 | 85.37 | 74.70 | 65.79 | 54.19 | **88.00** |
| | ADD(%)↑ | 38.78 | 90.62 | 32.03 | 51.91 | 26.39 | 27.51 | **32.50** |
| | CD(cm)↓ | 81.39 | 0.57 | - | **0.90** | 1.67 | 0.92 | 2.06 |
| | ATPF(s)↓ | - | - | 0.27 | 1.83 | 0.27 | 0.36 | **0.25** |
| MPM14 | ADD-S(%)↑ | 95.49 | 97.33 | 95.49 | **97.19** | 95.44 | 96.63 | 95.55 |
| | ADD(%)↑ | 90.16 | 94.52 | 88.02 | **94.18** | 87.75 | 92.43 | 88.88 |
| | CD(cm)↓ | 94.99 | 0.47 | - | **0.07** | 0.30 | 0.50 | 0.24 |
| | ATPF(s)↓ | - | - | **0.30** | 2.41 | 0.30 | 0.44 | 0.59 |
| SB11 | ADD-S(%)↑ | 94.44 | 97.27 | 94.54 | **97.07** | 94.40 | 95.17 | 94.55 |
| | ADD(%)↑ | 84.64 | 94.39 | 78.34 | **93.82** | 79.13 | 86.53 | 78.23 |
| | CD(cm)↓ | 75.83 | 0.46 | - | **0.12** | 0.58 | 0.60 | 0.63 |
| | ATPF(s)↓ | - | - | 0.23 | 2.18 | 0.23 | 0.37 | 0.59 |
| SB13 | ADD-S(%)↑ | 95.66 | 97.67 | 97.28 | **97.71** | 97.20 | 96.99 | 97.08 |
| | ADD(%)↑ | 85.47 | 95.24 | 92.68 | **95.31** | 92.90 | 92.09 | 92.30 |
| | CD(cm)↓ | 2.49 | 0.47 | - | **0.08** | 0.16 | 0.75 | 0.17 |
| | ATPF(s)↓ | - | - | 0.25 | 1.82 | 0.25 | 0.39 | **0.22** |
| SM1 | ADD-S(%)↑ | 84.94 | 96.9 | 89.36 | **96.87** | 89.39 | 87.44 | 91.31 |
| | ADD(%)↑ | 59.41 | 94.24 | 56.13 | **94.19** | 58.27 | 57.03 | 75.05 |
| | CD(cm)↓ | 2.04 | 0.44 | - | **0.08** | 1.43 | 0.91 | 0.77 |
| | ATPF(s)↓ | - | - | 0.33 | 5.25 | 0.34 | 0.74 | **0.27** |
| Mean | ADD-S(%)↑ | 92.39 | 96.52 | 93.96 | 94.87 | 92.80 | 89.85 | **95.07** |
| | ADD(%)↑ | 66.01 | 92.62 | 77.75 | **89.56** | 80.65 | 77.25 | 84.34 |
| | CD(cm)↓ | 52.05 | 0.57 | - | **0.15** | 0.42 | 0.75 | 0.41 |
| | ATPF(s)↓ | - | - | 0.29 | 2.10 | 0.28 | 0.44 | **0.24** |

Table 2. Comparison of ADD-S, ADD, and CD metrics, along with the Average Time Per Frame, across different methods on the YCBInEOAT dataset. ↑ indicates higher values are better, ↓ indicates lower values are better. The results in the first 2 columns are taken from [6]. * highlights the results reproduced using the open-source code from the authors of [6].

| Object | Metric | BundleTrack [5] | BundleSDF [6] | BundleTrack* | BundleSDF* | BundleSDF-async* | BundleSDF-Lite | Ours |
|---|---|---|---|---|---|---|---|---|
| cracker_box | ADD-S(%)↑ | 90.2 | 90.63 | 89.41 | 90.23 | 91.78 | 90.52 | **95.33** |
| | ADD(%)↑ | 85.08 | 85.37 | 63.24 | 80.29 | 66.38 | 81.99 | **91.3** |
| | CD(cm)↓ | 1.36 | 0.76 | - | 0.53 | 0.55 | 0.21 | **0.1** |
| | ATPF(s)↓ | - | **-** | 0.20 | 0.59 | 0.20 | 0.21 | **0.18** |
| bleach_cleanser | ADD-S(%)↑ | 95.22 | 94.28 | 82.45 | 93.48 | 95.47 | 92.18 | **93.81** |
| | ADD(%)↑ | 89.34 | 87.46 | 61.83 | 85.48 | 89.32 | 82.56 | **85.78** |
| | CD(cm)↓ | 1.31 | 0.53 | - | 0.44 | **0.13** | 0.16 | 0.25 |
| | ATPF(s)↓ | - | **-** | **0.21** | 1.01 | 0.22 | 0.23 | **0.21** |
| sugar_box | ADD-S(%)↑ | 90.68 | 93.81 | 81.42 | **96.58** | 96.51 | 89.48 | 96.21 |
| | ADD(%)↑ | 85.49 | 88.62 | 51.91 | 92.08 | 92.32 | 82.33 | **92.34** |
| | CD(cm)↓ | 2.25 | 0.46 | - | 0.23 | 0.09 | 0.22 | **0.07** |
| | ATPF(s)↓ | - | **-** | **0.20** | 0.70 | 0.22 | 0.26 | 0.21 |
| tomato_soup_can | ADD-S(%)↑ | 95.24 | 95.24 | 71.61 | 79.19 | **95.96** | 94.84 | 94.26 |
| | ADD(%)↑ | 85.78 | 83.1 | 41.36 | 57.3 | **92.11** | 82.45 | 86.14 |
| | CD(cm)↓ | 7.36 | 3.57 | - | 1.16 | **0.05** | 0.35 | 0.28 |
| | ATPF(s)↓ | - | **-** | **0.20** | 1.07 | 0.20 | 0.25 | 0.23 |
| mustard_bottle | ADD-S(%)↑ | 95.84 | 95.75 | 88.53 | 95.85 | 96.89 | 95.07 | **95.94** |
| | ADD(%)↑ | 92.15 | 89.87 | 71.92 | 90.15 | **94.11** | 86.95 | 92.22 |
| | CD(cm)↓ | 1.76 | 0.45 | - | 0.31 | **0.05** | 0.18 | 0.12 |
| | ATPF(s)↓ | - | **-** | **0.21** | 0.74 | 0.22 | 0.26 | 0.25 |
| Mean | ADD-S(%)↑ | 93.01 | 92.77 | 81.17 | 92.82 | 92.79 | 92.66 | **93.79** |
| | ADD(%)↑ | 87.26 | 86.95 | 57.91 | 84.28 | 83.75 | 83.41 | **87.83** |
| | CD(cm)↓ | 2.81 | 1.16 | - | 0.53 | 0.29 | 0.25 | **0.15** |
| | ATPF(s)↓ | - | **-** | **0.21** | 0.82 | **0.21** | 0.24 | 0.22 |