Model Based Multi-Object 6D Pose Estimation

Helge Meier

Abstract

The stable tracking of 6D poses for multiple objects is important for many fields of robotics. Previous 6D pose tracking algorithms mostly focus on the tracking of a single object at a time. This work proposes and implements a framework for methods that track multiple objects in RGBD-Images that offers multiple functionalities to ensure the stability and accuracy. The framework serves as a starting ground to realize and examine various methods for their pose tracking potential. One such method is implemented and tested in the paper.

1. Introduction

Recent advances in Natural Language Processing and Language Models (OpenAI, 2023) have demonstrated the value of large text corpora available on the internet for the creation of speech-based artificial intelligence. Large amounts of video recordings, as available on YouTube or from custom recordings, could serve to achieve a similar success in the field of robotics by letting an AI learn the behavior of objects subject to physics in a general way. The behavior of rigid objects is described by the 6D pose, containing position and orientation, and must be extracted from rgb images and possibly depth data. An additional application for the extraction of 6D poses from images are robot grasping problems (Yin & Li, 2022) where the perception of position and orientation of the object to grasp is a necessary starting point.
Both purposes require the ability of the 6D pose computation methods to work with sequences of images in a performant manner but while the first application is an offline task that can operate with less than real time performance, the application in robot perception demands for real time capabilities.

This work will focus on data driven methods as these dominate the 6D pose computation in terms of quality. Its aim is to develop a non-divergent and high framerate pose estimator for sequences of images. It contributes the following:

- A framework for the quick implementation of pose tracking methods with preprocessing, reinitialization and filters.
- Ideas for the modification of existing pose tracking algorithms to enable stable and accurate multi object pose tracking.
- An initial demonstration of a method based on the framework.

2. Background

2.1. Pose Estimation

In this paper pose estimation is defined as the computation of a 6D pose on the basis of a single image. Obviously pose estimation methods can be used on sequences of images too but in this section we are concerned which those that work without an initial pose and have no architecture ingredient that is specially designed for the use on image sequences. In addition the pose estimation methods are those that do not emphasize the real-time capabilities of their algorithms.

Pose estimation methods can be structured by the type of their image data, differentiating between pure RGB images and RGBD images, and their methodology to compute a 6D pose. In the following, three types of methodologies will be distinguished.

Direct Pose Estimation is an approach that tries to regress the pose information directly from images. EfficientPose (Bukschat & Vetter, 2020) uses the bidirectional feature pyramid network of EfficientNet (Tan & Le, 2020) as a basis and appends a subnetworks to predict rotation and translation as well as the class and bounding box of one or more objects in an RGB image. For the rotation subnet an additional convolutional refinement layer iteratively regresses a correction for the initial rotation. DenseFusion (Wang et al., 2019b) takes an RGBD image and computes a segmentation mask to do a bitwise cropping of objects in the color channels. The depth information is transformed to a point cloud. Using Neural Networks local features are computed from color image crops and point clouds before merging them pixelwise. In a next step DenseFusion creates global features in on the basis of the local features and concatenates both before estimating the pose. A pose refinement iteration uses the transformed object point cloud to infer a residual pose. MaskedFusion (Pereira & Alexandre,

2020) takes a similar approach as DenseFusion but replaces the pixelwise fusing of the depth and color features and the subsequent computation of global features with a single fusion step. The same pose refinement is used. According to (Pereira & Alexandre, 2020) MaskedFusion achieves comparable results to DenseFusion in the YCB-Video dataset and superior results in the LineMOD dataset.
The direct pose estimation methods all require a refinement method in some form.

Pure Iterative Refinement methods require an initial pose as input next to the image. They can be used as a refinement step for direct pose estimation methods discussed above but they are also quite similar to pose tracking methods discussed in section 2.2. In this way they come close to violating the category of pose estimation as defined previously. DeepIM (Li et al., 2019) renders an image of the CAD model using the input pose. Both the rendered image and the observed RGB image are feed into a Neural Network to compute a pose increment. Iterating multiple times can improve the final pose. (Lipson et al., 2022) present a refinement method for RGBD images. Other than DeepIM not one but several renderimages are created around the previous pose. For the combination of each renderimage and the observed image bidirectional correspondence maps are created. The final pose update is computed using a PnP-Algorithm (Fischler & Bolles, 1981a). DPOD (Zakharov et al., 2019) takes a similar refinement approach as DeepIM with the only difference being the separation of the network heads for the x,y, the z and the rotation refinement. (Zakharov et al., 2019) report superior performance using their refinement compared to DeepIM using the LineMOD dataset.

Correspondence/KeyPoint Matching methods compute either key points or correspondence maps on the observed images and a renderimage and aim to infer the 6D pose through this information. The last step often uses an PnP algorithm and least square fitting. The previously mentioned DPOD for example uses separated encoder decoder networks to compute a correspondence map and segmentation map and PnP+RANSAC (Fischler & Bolles, 1981b) to compute the pose. Perspective Flow Aggregation (Hu et al., 2022) avoids the necessity for a refinement process by rendering a set of sample poses of an object offline, computing a correspondence map for the closest N sample poses to the observed RGB image and inferring the 6D pose from correspondences from all N correspondence maps. ZebraPose (Su et al., 2022) modifies such an approach by introducing a hierarchy of coarse and fine correspondence maps on RGB images. GDR-Net (Wang et al., 2021) also uses dense correspondence maps computed with neural networks and render images but uses a learnable Patch-PnP algorithm resulting in a truly end-to-end learnable algorithm. GDR-Net is at the time of this writing the top performing method in most benchmarks of the BOP Challenge. (Hodan et al., 2018). While the previously presented methods used dense correspondence maps there is also the possibility to use key points for the pose estimation. KeypointCascadeVoting (Wu et al., 2022) uses encoder-decoder networks to predict key points instead of dense correspondence maps.
PVN3D (He et al., 2020) uses a encoder-decoder structure for the color and for the depth channels and fuses the features of both networks before computing keypoints for each object, segmenting them for different objects and inferring the 6D pose by least-squares fitting. FFB6D (He et al., 2021) follows the overall structure of PVN3D but replaces the single dense fusion network of the color and depth features by multiple bidirectional fusion stages in every encoder and decoder step before concatenating the features. Through key point detection and least square fitting the 6D pose is computed. The FFB6D outperforms PVN3D on both the YCB-Video and LineMOD dataset.

2.2. Pose Tracking

In contrast to pose estimation this paper defines pose tracking as a method that is designed to work on sequences of images and offers an inference time that allows for real-time application. 6-PACK (Wang et al., 2019a) relies on keypoint matching to compute the pose increment for subsequent RGBD frames. The authors report an inference frequency of 10Hz on a consumer GPU. BundleTrack (Wen & Bekris, 2021) uses key points too, but additionally stores novel poses in a memory pool to improve the pose update process. They achieve a similar inference frequency as 6-PACK. Some Pose Tracking methods have similarities to pure iterative refinement methods. Inversely, one can often use iterative refinement methods as a starting point for the solution pose tracking problems. Se(3)-TrackNet (Wen et al., 2020) adapts the idea of DeepIM (Li et al., 2019) to a RGBD pose tracking setting and achieve a frequency of 90Hz. Instead of refining the pose iteratively, the method is used to compute a pose update for every frame. A render image of the previous pose and the observed image are feed into separate encoders before being merged into a feature fusion network. At the end a translation and a rotation head predict the pose increment for the current frame's timestep. 6DCenterPose (Herrmann, 2023) tries to enable multi object tracking with a modified Se(3)-TrackNet. It adds a decoder for the object center heat map to the Se(3)-TrackNet structure and tries to predict the center location and the bounding box size in addition to the pose. Other than the Se(3)-TrackNet 6DCenterPose does feed cropped images into its network but the complete image. It achieves an inference frequency of 12.8 Hz.

2.3. Semantic Segmentation

Semantic Segmentation aims to parse a visual scene by assigning an object class label for every pixel. PIDNet (Xu et al., 2023) achieves this result by using three separate network branches analogue to PID controllers (Ang et al., 2005). (Xu et al., 2023) use the proportional branch to extract detailed, high resolution information, the integral branch supplies the global, long-range dependencies and the derivative branch extract the high frequencies at object class boundaries. PIDNet is implemented to be configurable into three different size classes offering a trade-off between inference speed and accuracy. The most accurate configuration enables inference with a frequency of 30Hz while the least accurate and smallest configuration achieves an inference frequency over 90Hz.

2.4. Object Detection

Object Detection detects objects of a certain type in images and regresses a bounding box around them in the image. YOLOX (Ge et al., 2021) splits the image processing into two neural network branches. One learns to capture the object class while the other regressed the bounding box of this object. Like PIDNet, YOLOX can be used in different network sizes resulting in inference frequencies ranging from 57Hz to more than 80Hz on images.

3. Towards multi object pose tracking

Evaluations of 6DCenterPose (Herrmann, 2023) showed results that were worse than Se(3)-TrackNet on the same dataset. We assume that the reasons for the bad results lay in the modification done by 6DCenterPose. While the Se(3)-TrackNet uses a cropped part of the image as an input to its neural network the 6DCenterPose uses the whole image as input. This prevents the 6DCenterPose network from focusing on the object that is supposed to be tracked. (Herrmann, 2023) tries to make up for this deficit by adding a heatmap to the input data that marks the objects centers. The problem is that an object center information is insufficient to give the network the ability to focus on the objects rotation behavior as this mainly depends on the image projection of the objects shape. Using only the center information, the network can not extract the objects shape projection as a feature. This reasoning is supported by the fact that (Herrmann, 2023) specifically reports the failure of his method to track the orientation of objects while their location is estimated more precise.

To prevent this problem we propose a modified 6DCenterPose. Instead of the object center heat map a segmentation map of the current image is used as the third input branch. This segmentation map could be defined as:

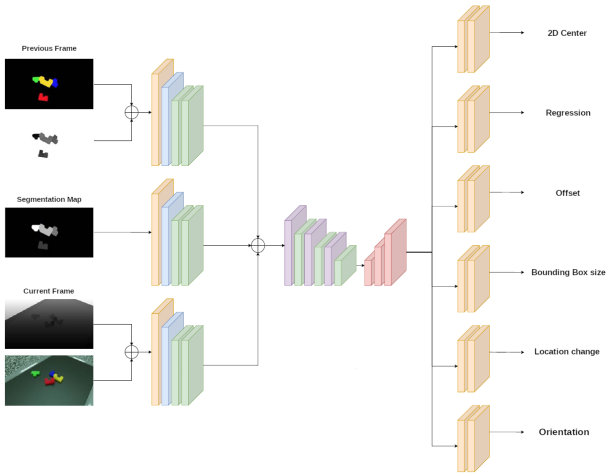

Figure 1. Modification concept of 6DCenterPose. Base concept and image taken from (Herrmann, 2023). Replacement of object center heatmap by segmentation maps as input to give focus to the network. The previous frame input feeds a rendering of the pose of the previous timestep to the network and the current frame input feeds the current image to the network. These inputs remain the same as in the base concept of (Herrmann, 2023).

$$I(x,y) = \begin{cases} 0 & (x,y) \text{ is background} \\ i \in [1,...,255] & (x,y) \text{ is object i} \end{cases}$$

This would allow for up to 255 different objects for an 8 bit image and would enable the network to attend to the shape of the objects and therefore improve the orientation results. The inference frequency would not increase. A visualization of the idea can be seen in figure 1.

Another possibility to adapt a pose tracking method to a multi object setting is to enable the basic Se(3)-TrackNet architecture to track multiple objects with one Se(3)-TrackNet instance and one set of weights. We propose to add an additional input next to the render image and the observed image that signals the object type to the network. Together with an increased depth of the networks this might enable the multi object Se(3)-TrackNet to learn pose tracking for different objects. During inference multiple objects would be feed into the multi object Se(3)-TrackNet simultaneously keeping inference time low. Figure 3 shows that the inference frequency of the Se(3)-TrackNet is not affected for batch sizes up to 10.

The third option is to do multi object tracking by using multiple instances of Se(3)-TrackNets. For each object type in the dataset a different set of weights is trained. Investigations in the memory consumption and the inference frequency of multiple Se(3)-TrackNet shows that the memory consumption scales by:

$$\text{Allocated GPU Memory}(n) \approx 50\text{MB} \cdot n$$

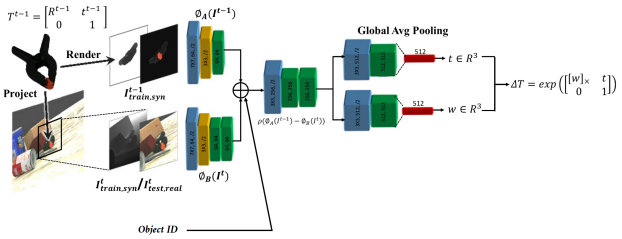

Figure 2. Modification concept of Se(3)-Tracknet. Base concept and image taken from (Wen et al., 2020). Addition of object ID input branch to enable the network to learn object pose track for multiple objects with the same set of weights. The two inputs for the render of the previous pose and the input of the current RGBD image stay the same as in the base concept.

where n is the number of Se(3)-TrackNet instances. As a result even entry level GPUs have enough memory to fit 10 instances of the Se(3)-TrackNets (Nvi). Running multiple Se(3)-TrackNet simultaneously reduces the inference frequency. Figure 3 shows that for up to 10 instances the frequency does not drop below 30 Hz. This inference time does not take the necessary render time of the Se(3)-TrackNet method into account.

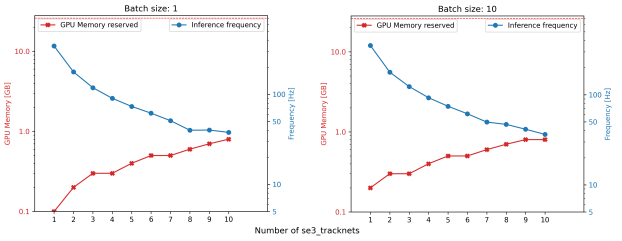

Figure 3. Comparison of GPU memory consumption and inference frequency over the number of se(3)-TrackNets used simultaneously for a batch size of 1 and 10. Vertical axis are logarithmically scaled.

4. A framework for multi object pose tracking with correction

The objective of this work is to create a framework that computes the 6D pose of multiple objects of different types for a sequence of RGBD images. This framework aims to achieve the following goals:

- A framework architecture that allows for the simple modifications of its segments without the need for any changes in other segments or the overlying algorithm.
- Offering a segment to integrate any image preprocessing step required by the main pose tracking algorithm.
- An object association algorithm that allows for the tracking and distinguishment of multiple objects of the same type regardless of any capabilities of the specific pose tracking method.
- Enabling a reinitialization of the tracked pose with a high accuracy method that may have a longer inference time than the basic pose tracker. This allows to tune any method based on the framework to be tuned for the optimal trade-off between accuracy and inference speed.
- Allowing for the asynchronous execution of the previously mentioned high accuracy method but hide the asynchronous implementation to ease the integration of the high accuracy method.
- A database implementation that allows for the insertion of pose information at the current and any previous timestep. The implementation must allow for the integration of any filtering process to smooth out noisy pose information, allow for high accuracy but delayed updates of the track and enable stable pose predictions.

The resulting framework is shown in figure 4. Every frame of the RGBD sequence is feed into the algorithm and results in one or more pose information for every object in the frame. The Object Detection segment is necessary to allow the Object Association to associate objects in the current frame with in previous images detected objects to ensure the continuity of the pose track. The Pose Tracker and the Pose Estimator segments are necessary to allow the implementation of very high accuracy high latency pose computation algorithm in the Pose Estimator segment and a high accuracy low latency pose computation algorithm in the Pose Tracker. It is important to notice that the Pose Estimator runs asynchronously and its results might be delayed by k steps with regard to the overall iteration. The computed pose information is feed into the pose database before working on the next frame or delayed after multiple following frames.

Every RGBD-image is initially processed synchronously by the **Object Detection** segment of the framework. This architecture segment offers a base class for the implementation of any preprocessing step required by following pose computation algorithms. FFB6D (He et al., 2021) for example demands for segmentation maps of the RGBD-Image. These maps can be computed by integrating PIDNet (Xu et al., 2023) segment. In addition to detect objects entering the frame requires a object detection mechanism. This can be realized by integrating YOLOX (Ge et al., 2021) into the Object Detection segment of the framework.

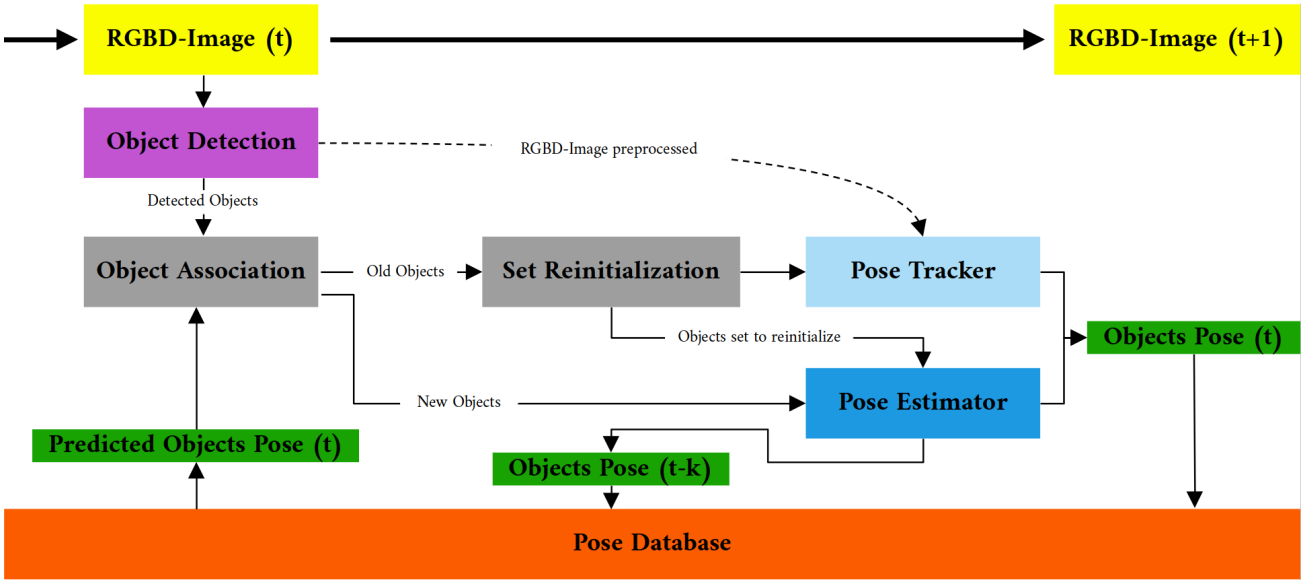

Figure 4. The multi object pose tracking algorithm framework depicted in an abstract form.

The Object Detection results in a list of objects that have to be associated with objects seen in previous images as provided by the pose database. The **Object Association** achieves this by first separating objects with different types. This results in a linear assignment problem for each object type where the detected objects of the given type have to be matched to the previously known objects of this type. The cost function is given by a distance metric between the detected object's image coordinates and the predicted object's image coordinates. The linear assignment problem is solved using the Hungarian method (Kuhn, 1955) with an upper threshold regarding the distance. Objects that can not be associated are marked as new.

The **Set Reinitialization** allows for the implementation of a custom method that decides which old object requires a pose reinitialization by a high accuracy method.

The **Pose Tracker** is the basis for the integration of pose tracking methods with a fast inference speed. It is executed synchronously for all objects that are not new and not set for reinitialization. This segment in the framework is the place to integrate the concepts given in section 3. It is the main ingredient of a realized multi object pose tracking algorithm and therefore dominates the accuracy of the computed poses and the inference speed.

The **Pose Estimator** segment of the framework enables the integration of high accuracy, slow inference time pose estimators. The implementation is asynchronous. For new objects the frameworks blocks until the inference is completed because a valid pose is necessary to continue with fast inference pose tracking methods. For objects that were set to reinitialization the pose estimator is not required to complete before the next RGBD-Image is processed but can finish at any time. The Pose estimator is queried for completed poses. Due to the asynchronous execution these pose belong to a previous timestep.

The **Pose Database** stores all pose observations and offers predictions for non-observed poses. On the basis of the pose database multiple filtering processes can be realized.
The simplest implemented approach is pose averaging combined with the nearest neighbor prediction. Pose averaging is implemented using the Slerp approach (Shoemake, 1985) while the prediction just used the averaged pose observation from timestep $t-1$ to predict a pose for timestep t. This filter approach can handle multiple, irregular observations does not achieve any benefit from asynchronous, high accuracy pose estimation.
An alternative filter implementation can achieve benefit from asynchronous reinitialization. Instead of averaging multiple pose observations, high accuracy, delayed observations are used to reset the observed pose trajectory. Assume at timestep $t-k$ a pose reinitialization is started. This reinitialization ends at timestep t and provides the highly accurate pose T^h_{t-k} while the pose tracker has continued to compute the poses $[T_{t-k}, ..., T_t]$. To compute an updated pose trajectory the pose increment is computed using the formula $\Delta T_{j \to j+1} = T_{j+1} \left(T_j\right)^{-1}$. Using this pose increment one can compute an up-

dated pose trajectory $[T^h_{t-k}, T^u_{t-k+1}, ..., T^u_t]$ using the operations $T^u_{t-k+1} = \Delta T_{t-k \to t-k+1} T_{t-k}$ and $T^u_{t+j} = \Delta T_{t+j-1 \to t+j} T_{t+j-1} \quad \forall j \in [-k+2, ..., 0]$. Although this approach allows a correction of the pose trajectory using delayed pose observations it remains questionable because it treats pose increments as a function of the timestep t rather than as a function of the previous pose T_{t-1} and the current image observation. It assumes that, for example in the case of the se(3)-TrackNet (Wen et al., 2020), if the equality $incr(T) = se(3)(T)$ holds for one T, with $incr(T) = \Delta T \cdot T$ and $se(3)(T)$ being the inference in the se(3)-TrackNet, it will hold for any T. This assumption is not true in general.

The third alternative filter implementation is the usage of a Kalman filter (Zarchan & Musoff, 2015) to predict a pose. This approach can incorporate delayed observations in his predictions but it remains questionable if an additional high accuracy observation at timestep $t - k$ will have a significant influence on prediction $t + 1$ regarding the influence of the numerous, intermediate low accuracy predictions.

The **multi object pose tracking framework** is implemented in a object oriented fashion and relies heavily on abstract classes. Modifications can be realized by creating child classes from the abstract classes and reusing or overwriting member functions. With this implementation style the previously listed goals have been achieved.

5. Experiments

A possible setup for the developed framework only uses the FFB6D as a pose estimator for the computation of poses without any pose tracker. The necessary modification for this is to overwrite the Set Reinitialization function to return true for all objects. The resulting multi object pose estimation method can compute 0.9 frames per second on average on a Intel(R) Core(TM) i5-4670K @ 3.40GHz processor with a NVIDIA GeForce RTX 4090 gpu. The FFB6D was used with the pretrained weight provided by (He et al., 2021). The YCB-Video dataset provided by (He et al., 2021) was used for evaluation.

To evaluate the overall accuracy of this multi object pose tracking method the ADD results of multiple videos are collected and ordered by the tracked object type. The ADD is computed according to $ADD = \frac{1}{m} \sum_{x \in M} \| Rx + t - (R'x + t') \|$. To make YCB videos of different length comparable the ADD result sequences where normalized to the same length. In a next step if multiple ADD tracks for the same objects exist the maximum operator is applied to get a picture of the maximally existing error. Figures 6 and 7 shows the ratio of the ADD score over the maximal diameter of the object for every frame in the sequence. Usually papers treat any pose with an ADD / max Object Diameter of more than 0.1 as false (He et al., 2021). This margin is show in figures 6 and 7 by a black dashed line. While the objects in figure 7 show acceptable results the ones in figure 6 exceed the margin regularly and by a large amount. This shows that this realization of a multi object pose tracker does not achieve the accuracy goals on half the objects although it uses FFB6D (He et al., 2021) with the original neural network weights on the training dataset. The results do not match with the accuracy reported on the original paper (He et al., 2021).

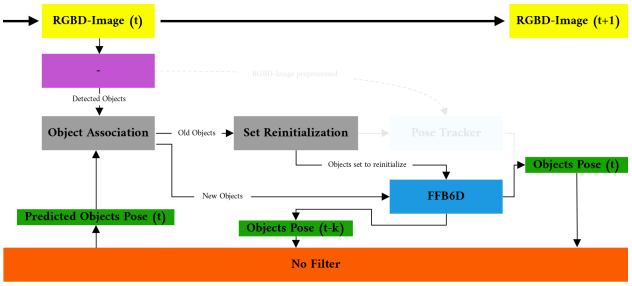

Figure 5. Implementation of a pose tracking method using the framework show in figure 4. Only FFB6D is used for pose estimation. No pose tracker is implemented. The method does not need additional preprocessing methods, so no object detection methods are integrated. There is only one pose observation per timestep and no asynchronous pose update so there is no need for a filter based on the pose database.

6. Conclusion

This work presents a framework for the efficient realization of multi object 6D pose estimation with optional image preprocessing, asynchronous reinitialization of object poses and filters that allow delayed observation updates. Using this framework a method was realized using only a pose estimation method. This methods accuracy was shown on one testcase. The future objective is to use this framework to realize more advanced pose tracking methods that fully utilize the capabilities of the pose tracker, the filter and the asynchronous reinitialization. In addition we aim to realize methods that achieve a higher frame per second rate than the one show on section 5. One example for that will be the multi se(3)-TrackNet approach described in section 3.

References

Nvidia rtx 30 series, technologies. https://www.nvidia.com/en-us/geforce/graphics-cards/compare/?section=compare-specs. Accessed: 2023-09-04.

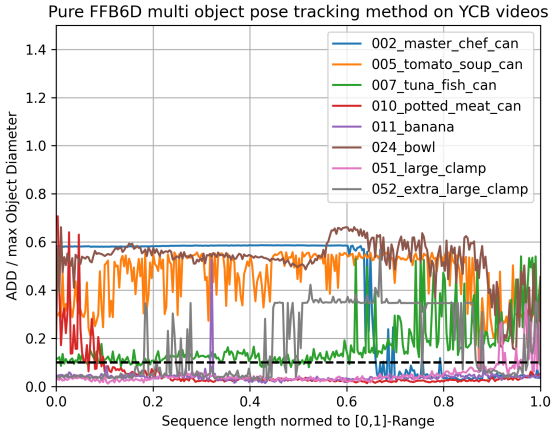

Figure 6. Results of realized multi object pose tracking method using FFB6D on YCB video objects shown in figure 5. The object pose maximum ADD value from multiple videos is plotted for each object type. This figure shows the objects where this value does exceed the 0.1 margin significantly.

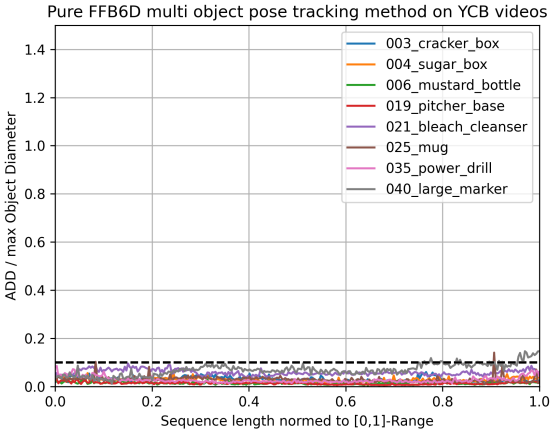

Figure 7. Results of realized multi object pose tracking method using FFB6D on YCB video objects shown in figure 5. The object pose maximum ADD value from multiple videos is plotted for each object type. This figure shows the objects where this value does not or only slightly exceed the 0.1 margin.

Ang, K. H., Chong, G., and Li, Y. Pid control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13(4):559–576, 2005. doi: 10.1109/TCST.2005.847331.

Bukschat, Y. and Vetter, M. Efficientpose: An efficient, accurate and scalable end-to-end 6d multi object pose estimation approach, 2020.

Fischler, M. A. and Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981a. ISSN 0001-0782. doi: 10.1145/358669.358692. URL https://doi.org/10.1145/358669.358692.

Fischler, M. A. and Bolles, R. C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, jun 1981b. ISSN 0001-0782. doi: 10.1145/358669.358692. URL https://doi.org/10.1145/358669.358692.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. Yolox: Exceeding yolo series in 2021, 2021.

He, Y., Sun, W., Huang, H., Liu, J., Fan, H., and Sun, J. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation, 2020.

He, Y., Huang, H., Fan, H., Chen, Q., and Sun, J. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation, 2021.

Herrmann, P. 6dcenterpose: Multi-object rgb-d 6d pose tracking with synthetic training data, 2023.

Hodan, T., Michel, F., Brachmann, E., Kehl, W., Buch, A. G., Kraft, D., Drost, B., Vidal, J., Ihrke, S., Zabulis, X., Sahin, C., Manhardt, F., Tombari, F., Kim, T.-K., Matas, J., and Rother, C. Bop: Benchmark for 6d object pose estimation, 2018.

Hu, Y., Fua, P., and Salzmann, M. Perspective flow aggregation for data-limited 6d object pose estimation, 2022.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955. doi: https://doi.org/10.1002/nav.3800020109. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109.

Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. DeepIM: Deep iterative matching for 6d pose estimation. *International Journal of Computer Vision*, 128(3):657–678, nov 2019. doi: 10.1007/s11263-019-01250-9. URL https://doi.org/10.1007%2Fs11263-019-01250-9.

Lipson, L., Teed, Z., Goyal, A., and Deng, J. Coupled iterative refinement for 6d multi-object pose estimation, 2022.

OpenAI. Gpt-4 technical report, 2023.

Pereira, N. and Alexandre, L. A. Maskedfusion: Mask-based 6d object pose estimation, 2020.

Shoemake, K. Animating rotation with quaternion curves. *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, 1985. URL https://api.semanticscholar.org/CorpusID:11290566.

Su, Y., Saleh, M., Fetzer, T., Rambach, J., Navab, N., Busam, B., Stricker, D., and Tombari, F. Zebrapose: Coarse to fine surface encoding for 6dof object pose estimation, 2022.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

Wang, C., Martín-Martín, R., Xu, D., Lv, J., Lu, C., Fei-Fei, L., Savarese, S., and Zhu, Y. 6-pack: Category-level 6d pose tracker with anchor-based keypoints, 2019a.

Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., and Savarese, S. Densefusion: 6d object pose estimation by iterative dense fusion, 2019b.

Wang, G., Manhardt, F., Tombari, F., and Ji, X. Gdr-net: Geometry-guided direct regression network for monocular 6d object pose estimation, 2021.

Wen, B. and Bekris, K. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models, 2021.

Wen, B., Mitash, C., Ren, B., and Bekris, K. E. se(3)-TrackNet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2020. doi: 10.1109/iros45743.2020.9341314. URL https://doi.org/10.1109%2Firos45743.2020.9341314.

Wu, Y., Javaheri, A., Zand, M., and Greenspan, M. Keypoint cascade voting for point cloud based 6dof pose estimation, 2022.

Xu, J., Xiong, Z., and Bhattacharyya, S. P. Pidnet: A real-time semantic segmentation network inspired by pid controllers, 2023.

Yin, Z. and Li, Y. Overview of robotic grasp detection from 2d to 3d. *Cognitive Robotics*, 2:73–82, 2022. ISSN 2667-2413. doi: https://doi.org/10.1016/j.cogr.2022.03.002. URL https://www.sciencedirect.com/science/article/pii/S2667241322000052.

Zakharov, S., Shugurov, I., and Ilic, S. Dpod: 6d pose object detector and refiner, 2019.

Zarchan and Musoff. *Fundamentals of Kalman Filtering: A Practical Approach, Fourth Edition*. American Institute of Aeronautics and Astronautics, Inc., 1801 Alexander Bell Drive, Reston, VA 20191-4344, 2015.