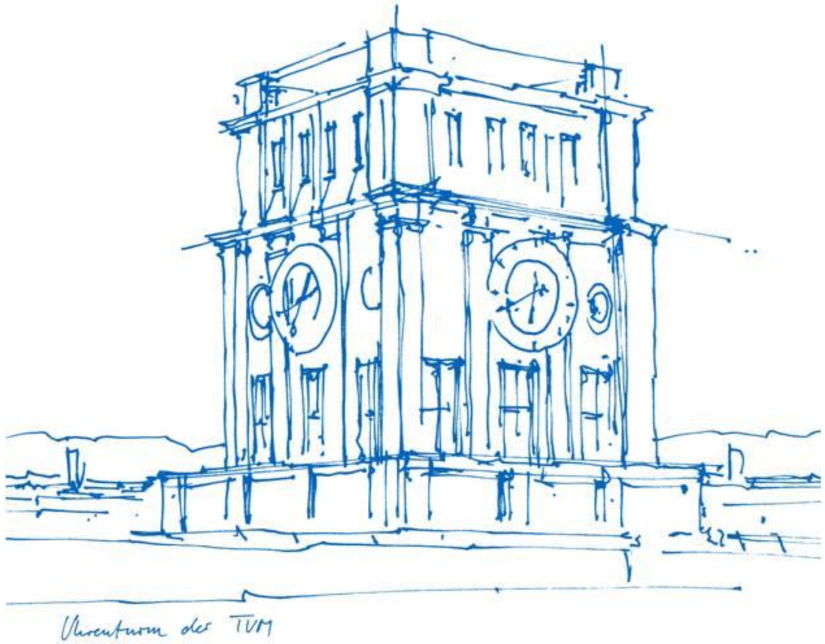
Dynamic Object SLAM with Dense Optical Flow

Dongyue Lu

Advisor: Dr.Xingxing Zuo, Weicai Ye(Zhejiang University)

Supervisor: Prof. Dr. Stefan Leutenegger

July 3, 2023



Motivation

- Static environment assumption can lead to severe performance degradation of SLAM systems in dynamic scenarios
- Dynamic object tracking is important in many applications, like autonomous driving, multi-robot collaboration and augmented/virtual reality

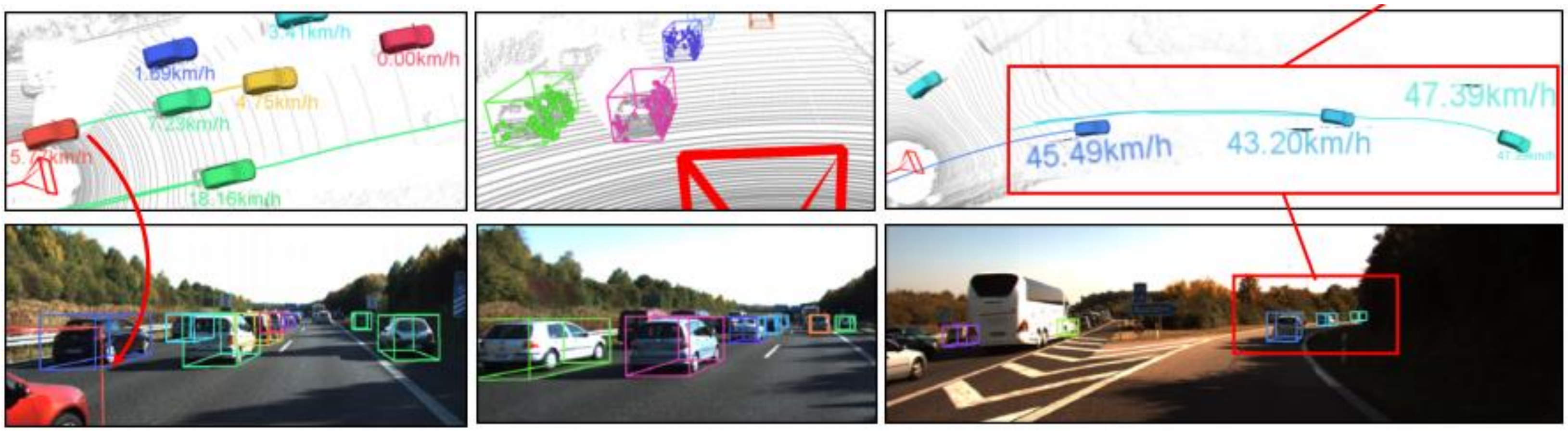


Figure 1: camera and object pose joint estimation

Related Literature

- Dynamic SLAM
  - **DynaSLAM II**[1] and **VDO-SLAM**[2] incorporate static and dynamic feature points and other motion constraints into a joint optimization problem to track the camera and dynamic objects jointly

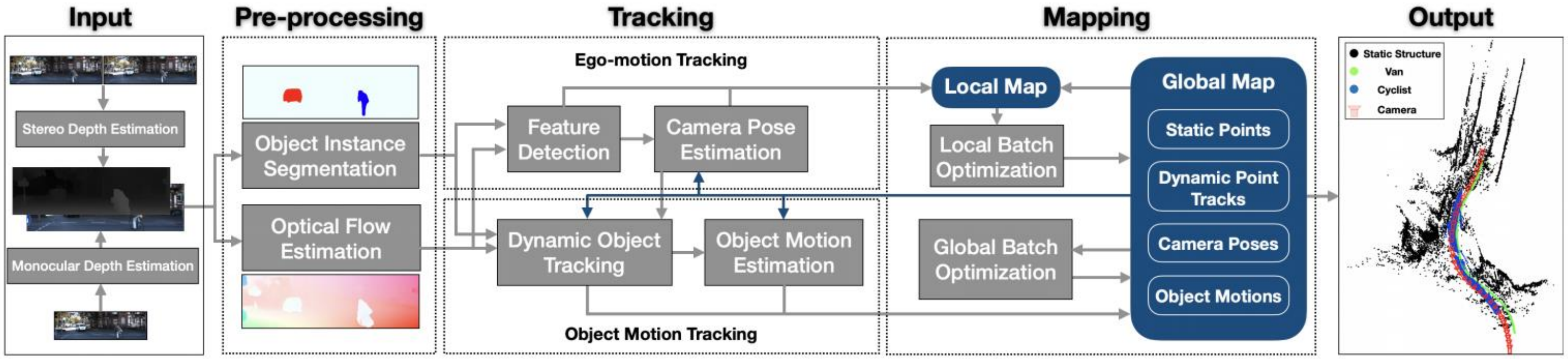


Figure 2: Overview of VDO-SLAM[2] system

Related Literature

- RAFT optical flow
  - Droid SLAM[3] and DeFlow SLAM[4] realize accurate camera tracking using camera flow provided by RAFT[5] optical flow estimator
  - Based on RAFT[5] , RAFT-3D[6] and Multi-Scale RAFT[7] provide better identification of rigidly moving regions and multi-resolution optical flow estimation

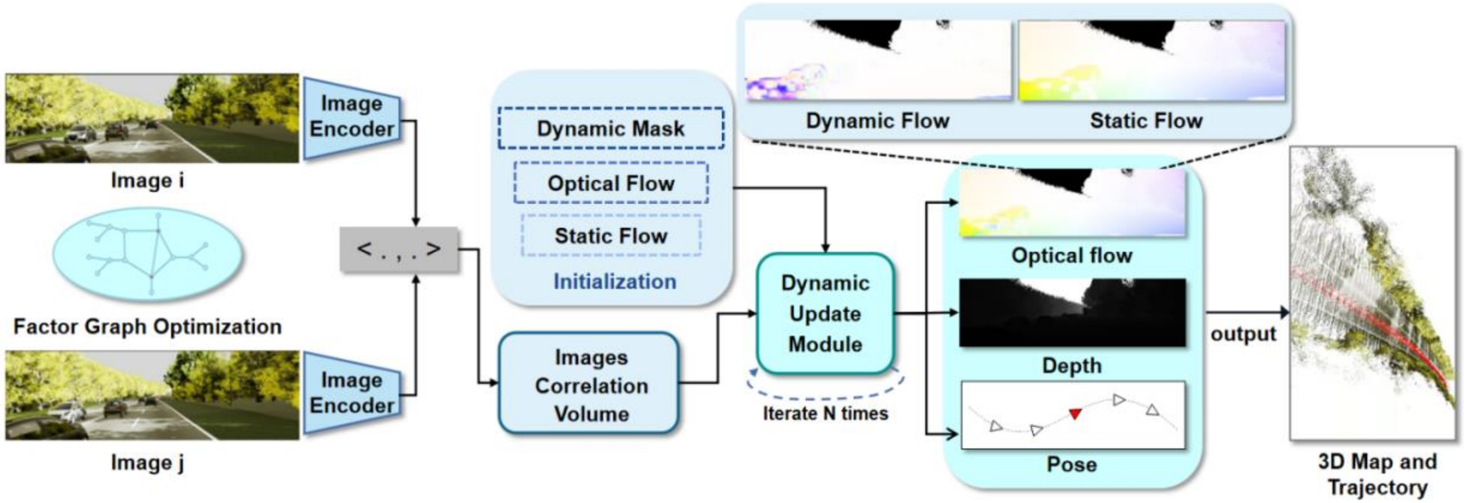


Figure 3: Overview of DeFlow SLAM[4]

Method

- Overview

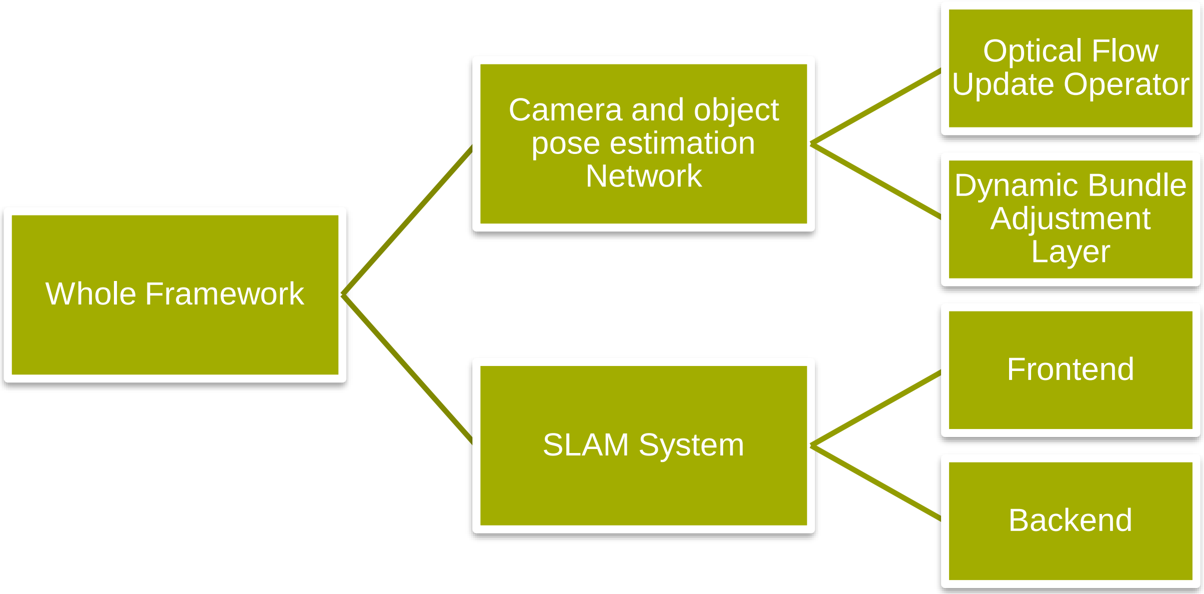


Figure 4: Overview of the entire framework

Method

- Camera and object pose estimation Network: Design 1

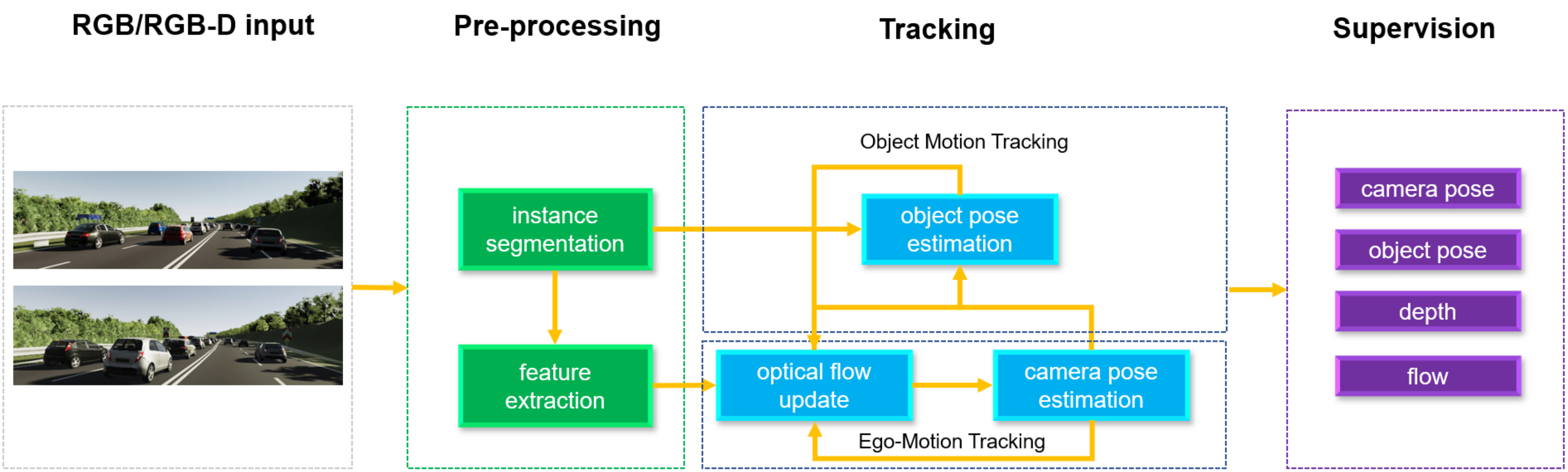


Figure 5: Pose estimation network design 1 overview

Method

- Dynamic Bundle adjustment
  - Cost function on static region

$$E({}^{c}_{w}\boldsymbol{T},\ {}^{c}\boldsymbol{d}) = \sum_{(i,j)\in\varepsilon} ||\boldsymbol{f}_{pred} - \Pi_c\,({}^{c_j}_{w}\boldsymbol{T}\,{}^{w}_{c_i}\boldsymbol{T} \circ \Pi_c^{-1}({}^{c_i}\boldsymbol{p},\ {}^{c_i}\boldsymbol{d}))||^2_{\Sigma_{i,j}}$$

  - Cost function on dynamic region

$$E({}^{o}_{w}\boldsymbol{T}, {}^{c}_{w}\boldsymbol{T},\ {}^{c}\boldsymbol{d}) = \sum_{(i,j)\in\varepsilon} ||\boldsymbol{f}_{pred} - \Pi_c\,({}^{c_j}_{w}\boldsymbol{T}\,{}^{w}_{o_j}\boldsymbol{T}\,{}^{o_i}_{w}\boldsymbol{T}\,{}^{w}_{c_i}\boldsymbol{T} \circ \Pi_c^{-1}({}^{c_i}\boldsymbol{p},\ {}^{c_i}\boldsymbol{d}))||^2_{\Sigma_{i,j}}$$

  - The system can be solved efficiently using Gauss-Newton algorithm and Schur complement

Method

- Training details
  - Dataset: 06,18,20 from virtual KITTI[8], each is a 6-frame video sequence
  - Object selection: select object within a certain distance(0.2-30m) and with sufficient constraints(>80pixel under 1/8 resolution)
  - Frame selection: Appropriate camera flow (8-96px) and object flow (20-50px) **OR** simply take a frame every two frames

Method

- Supervision
  - Camera pose loss

$$\mathcal{L}_c = \sum_i \gamma^{N-i} \| Log_{SE3}(\mathbf{G}_c{}^{-1} \cdot \hat{\mathbf{T}}_{ci}) \|_2$$

  - Object pose loss

$$\mathcal{L}_{ok} = \sum_i \gamma^{N-i} \| Log_{SE3}(\mathbf{G}_{ok}{}^{-1} \cdot \hat{\mathbf{T}}_{oki}) \|_2$$

  - Induced flow loss

$$\mathcal{L}_{induced} = \sum_i^N \gamma^{N-i} \| \mathbf{p}_{gt} - \hat{\mathbf{p}}_i \|_2$$

  - Depth loss

$$\mathcal{L}_{depth} = \sum_i^N \gamma^{N-i} \| d_{gt} - \hat{d} \|_1$$

  - Residual loss

$$\mathcal{L}_r = \sum_i^N \gamma^{N-i} \| r_i \|_1$$

Method

- Training Result
  - Validation result on VKitti[8] validation sequence(RGB-D, Clone from 06,18,20)

	Camera rotation error(RPE/°)	Camera translation error(RPE/m)	Object rotation error(RPE/°)	Object translation error(RPE/m)
Design 1	0.06	0.006	0.4	0.02

Table 1: Test result of design 1

- Resolution Experiment
  - Increasing the image resolution can effectively improve the accuracy of object pose estimation(RGB-D, Sequence20, 20frames)

Resolution	Camera pose error(ATE/m)	Object pose error(ATE/m)
30*101	0.00286	0.81
60*202	0.000793	0.33
120*404	0.000151	0.032
240*808	0.0000823	0.012
375*1242	0.0000384	0.00207

Table 2: Resolution experiment

Method

- Camera and object pose estimation Network: Design 2, coarse to fine optimization

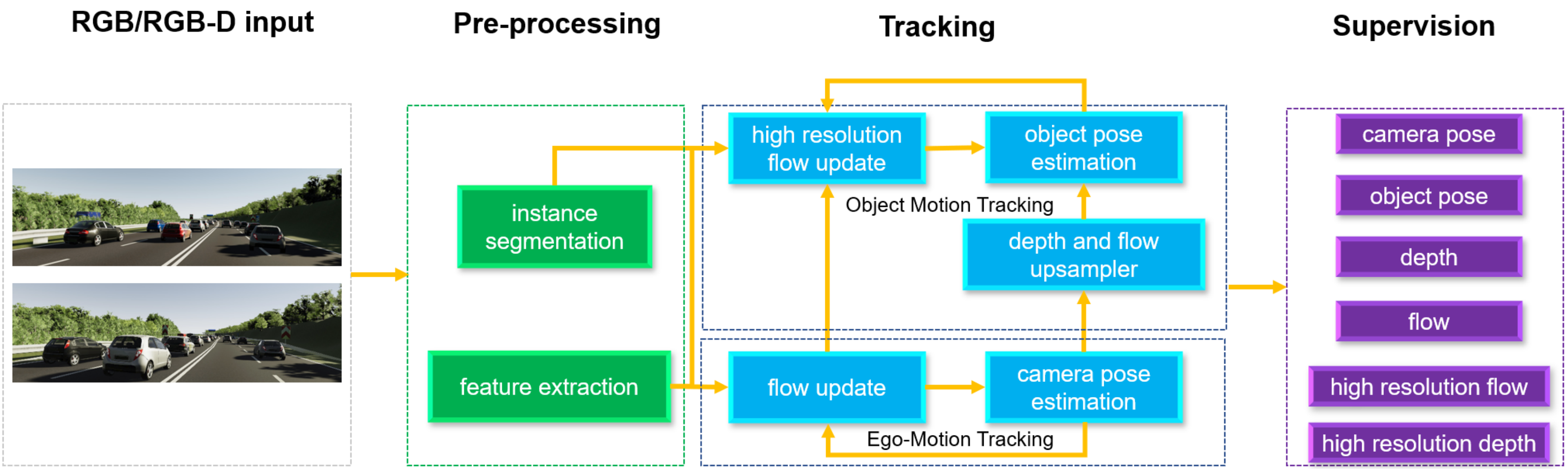


Figure 6: Pose estimation network design 2 overview

Method

- Coarse to fine optimization
  - Choose the size of the object patch according to the movement of each object in the sequence

Figure 7: Object patches

Method

- Training Result
  - Validation result on VKitti[8] validation sequence(RGB-D, clone from 06,18,20)

	Camera rotation error(RPE/°)	Camera translation error(RPE/m)	Object rotation error(RPE/°)	Object translation error(RPE/m)
Design 1	0.06	0.006	0.4	0.02
Design 2	0.04	0.006	0.2	0.005

Table 3: Test result of design 2

Method

- Camera and object pose estimation Network: Design 3

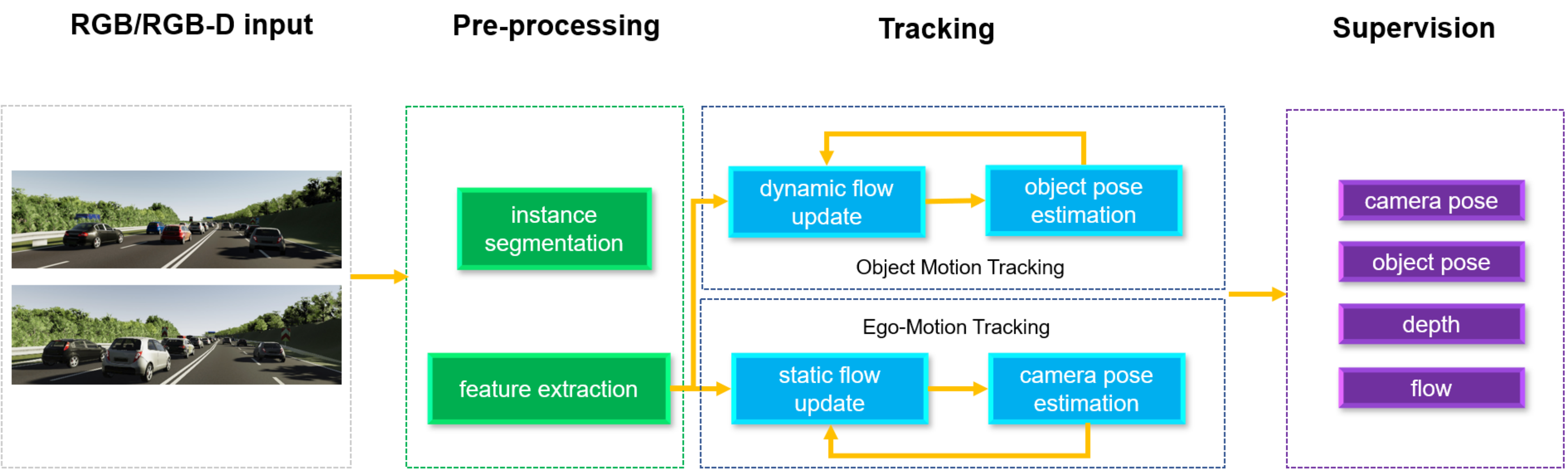


Figure 8: Pose estimation network design 3 overview

Method

- Training Result
  - Validation result on VKitti[8] validation sequence(RGB-D, clone from 06,18,20)

	Camera rotation error(RPE/°)	Camera translation error(RPE/m)	Object rotation error(RPE/°)	Object translation error(RPE/m)
Design 1	0.06	0.006	0.4	0.02
Design 2	0.04	0.006	0.2	0.005
Design 3	0.08	0.009	0.18	0.004

Table 4: Test result of design 3

Method

- ICP constraint
  - Incorporate the spatial point-to-plane error of two point clouds in addition to the optical flow reprojection error in RGB-D settings to improve the accuracy of object pose estimation
  - Cost function on static region

$$E({}^{c}_{w}T) = \boldsymbol{n}_i^T(\Pi_c^{-1}\left({}^{c_i}\boldsymbol{p}, {}^{c_i}\boldsymbol{d}\right) - {}^{c_i}_{w}T\,{}^{w}_{c_j}T \circ \Pi_c^{-1}\left({}^{c_j}\boldsymbol{p}, {}^{c_j}\boldsymbol{d}\right))$$

  - Cost function on dynamic region

$$E({}^{c}_{w}\boldsymbol{T}, {}^{o}_{w}\boldsymbol{T}) = \boldsymbol{n}_i^T(\Pi_c^{-1}\left({}^{c_i}\boldsymbol{p}, {}^{c_i}\boldsymbol{d}\right) - {}^{i}_{w}\boldsymbol{T}\,{}^{w}_{o_i}\boldsymbol{T}\,{}^{j}_{w}\boldsymbol{T}\,{}^{w}_{c_j}\boldsymbol{T} \circ \Pi_c^{-1}\left({}^{c_j}\boldsymbol{p}, {}^{c_j}\boldsymbol{d}\right))$$

Method

- Training Result
  - Validation result on VKitti[8] validation sequence(RGB-D, clone from 06,18,20)

	Camera rotation error(RPE/°)	Camera translation error(RPE/m)	Object rotation error(RPE/°)	Object translation error(RPE/m)
Design 1	0.06	0.006	0.4	0.02
Design 2	0.04	0.006	0.2	0.005
Design 3	0.08	0.009	0.18	0.004
Design 1 +ICP constraint	**0.014**	**0.0008**	**0.07**	**0.004**
Design 2 +ICP constraint	0.013	0.0007	0.06	0.003
Design 3 +ICP constraint	0.013	0.0008	0.08	0.004

Table 5: Test result of different designs and ICP

Method

- Training on other indoor datasets
    - Co-Fusion[9], dataset from Xu et al.[10], Self-rendered dataset

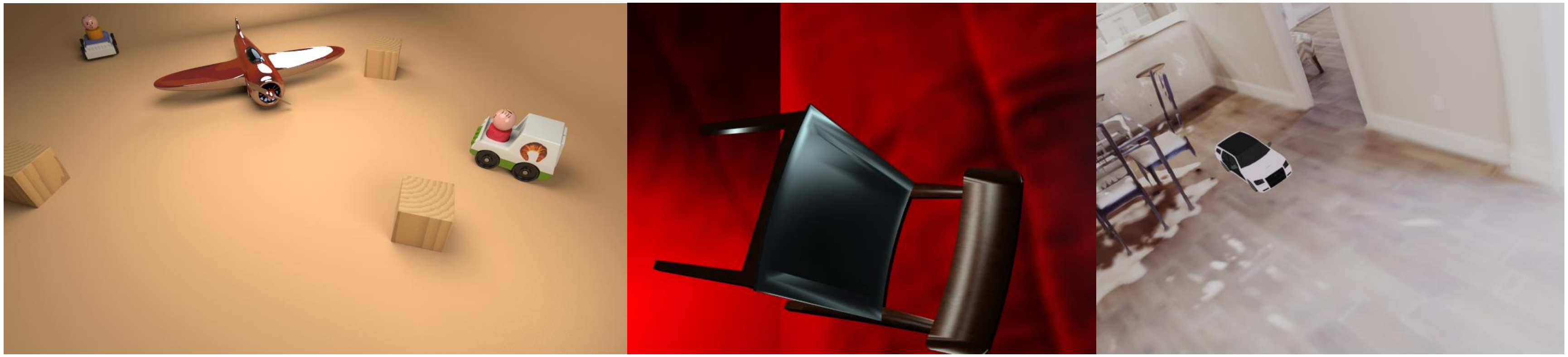


Figure 9: Example from other datasets

Method

- Training Result
  - Validation result on validation sequence

	Camera rotation error(RPE/°)	Camera translation error(RPE/m)	Object rotation error(RPE/°)	Object translation error(RPE/m)
Design 1 +ICP constraint	**0.014**	**0.0008**	**0.07**	**0.004**
Co-Fusion[11]	0.11	0.01	0.22	0.14
Dataset from Xu et al.[12]	0.16	0.05	0.34	0.23
Self-rendered dataset	0.43	0.02	0.83	1.2

Table 6: Test result on other datasets

Method

- SLAM system: Motion Filter
  - Instance Segmentation via Mask-RCNN, data association by IOU
  - Initialize a frame graph and object-frame graph representing the relationship between objects and keyframes and run the update operator once enough keyframes are accumulated

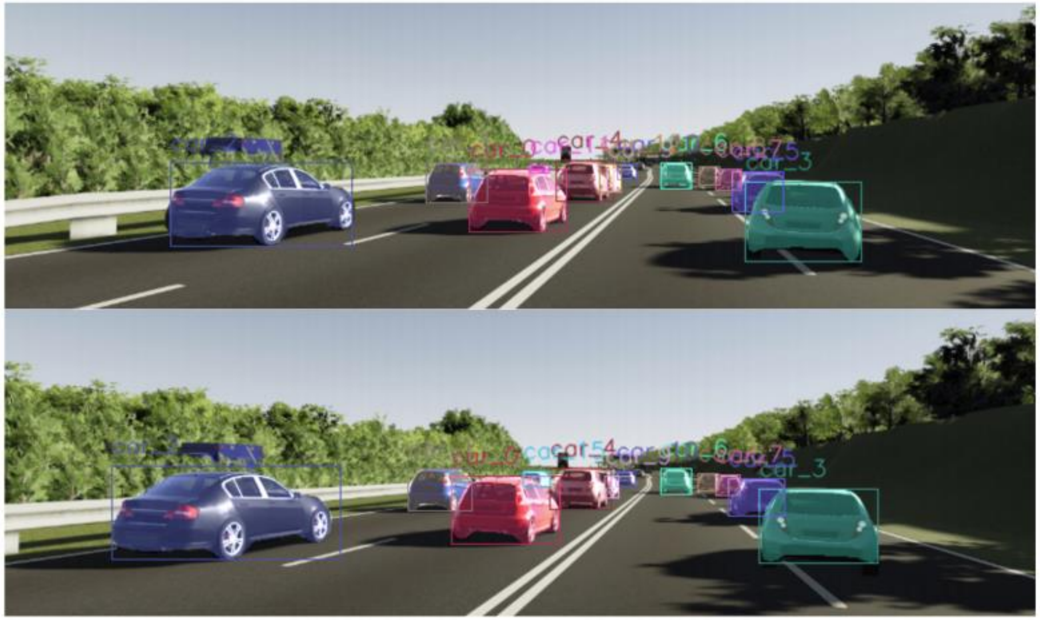


Figure 10: Data assocation

Method

- SLAM system
  - Frame graph Initialization: For each frame, in addition to adding the three nearest neighbors to the frame graph, the frame with the largest induced flow between each pair of frames is also added,
  - Frontend: New keyframe is added to the frame graph adding edges with its closest neighbors as measured by mean optical flow and timestamps
  - Backend: Rebuild frame graph using the flow between all pairs of keyframes in each iteration and perform global bundle adjustment
  - Perform motion-only bundle adjustment by iteratively estimating flow between each keyframe and its neighboring non-keyframes and evaluate on the full camera trajectory

Inference

- VKitti[8] Dataset
  - Clone from 18,20

	VK18 Camera pose error (ATE/m)	VK18 Object pose error (car1, ATE/m)	VK20 Camera pose error (ATE/m)	VK20 Object pose error (car7, ATE/m)
DynaSLAM	Fail	-	2.807	-
Droid SLAM	1.190	-	6.998	-
DeFlow SLAM	0.400	-	1.039	-
Ours(Monocular)	0.386	1.148	1.031	1.264
Ours(RGB-D)	**0.316**	0.626	**1.024**	0.478

Table 7: Inference result on Vkitti[10] dataset

Inference

- KITTI Tracking[11] Dataset
  - Camera pose estimation

Sequence	VDO-SLAM			DynaSLAM II			Ours		
	ATE[m]	RPE[m/f]	RPE[°/f]	ATE[m]	RPE[m/f]	RPE[°/f]	ATE[m]	RPE[m/f]	RPE[°/f]
0018	-	0.07	**0.02**	1.09	0.05	**0.02**	**0.72**	**0.04**	**0.02**
0020	-	0.16	**0.03**	**1.36**	**0.07**	0.04	1.38	0.08	0.04

Table 8: camera pose estimation result on KITTI Tracking[11] dataset

Inference

- KITTI Tracking[11] Dataset
  - Object pose estimation

Sequence	VDO-SLAM			DynaSLAM II			CubeSLAM			Ours		
	ATE[m]	RPE[m/f]	RPE[°/f]	ATE[m]	RPE[m/f]	RPE[°/f]	ATE[m]	RPE[m/f]	RPE[°/f]	ATE[m]	RPE[m/f]	RPE[°/f]
0018 car2	-	0.08	0.25	1.10	0.30	9.27	-	3.79	3.18	**0.45**	**0.07**	**0.23**
0018 car3	-	-	-	1.13	0.55	20.05	-	-	-	**0.62**	**0.22**	**0.47**
0020 car0	-	**0.08**	**0.37**	0.56	0.45	1.30	-	5.70	3.42	0.38	0.18	0.38
0020 car12	-	-	-	1.18	0.40	6.19	-	-	-	**0.72**	**0.24**	**0.36**

Table 9: object pose estimation result on KITTI Tracking[11] dataset

Ablation studies

- Ratio between the two constraints
  - Clone from 18

	Camera pose error(ATE/m)	Object pose error(car1, ATE/m)
Only 2D reprojection error	0.326	1.119
Only 3D ICP constraint	0.354	1.386
10:1	0.318	1.078
5:1	0.316	0.789
2:1	**0.316**	0.626
1:1	0.313	0.743

Table 10: Comparison of different configurations

Ablation studies

- Keyframe selection strategy
  - Clone from 18

	Camera pose error(ATE/m)	Object pose error(car1, ATE/m)
Take one frame every two frames	0.316	**0.626**
Appropriate camera flow and object flow	**0.289**	1.236

Table 11: Comparison of different keyframe strategy

Time Analysis

- Time test
  - NVIDIA GeForce RTX 3070 graphics card with 8GB of VRAM and an Intel i5-10600 CPU
  - Test on Vkitti18, Clone

Module	Time[ms]
Frontend	159
Backend	186

Table 12: Time Analysis

Future Work

- Monocular Depth Prior guided optimization
  - Initially proved that the system has better camera pose estimation accuracy with a depth prior from monocular depth prediction network
  - Vkitti, all scenes

	Camera rotation error(RPE/°)	Camera translation error(RPE/m)
With depth prior	**0.008**	**0.0005**
Without depth prior	0.011	0.0007

Table 13: Comparison of camera pose estimation with depth prior

Future Work

- Shape Reconstruction
  - Realize the shape reconstruction or novel view synthesis of dynamic objects using DeepSDF[12] or NeRF[13] based on accurate pose and depth, DSP SLAM[14] DiscoScene[15]etc.

Conclusion

- Propose a novel method for dynamic SLAM that combines classical optimization techniques with deep learning to achieve high accuracy in estimating camera poses, dynamic object poses in dynamic scenearios
- Employ a dynamic differentiable bundle adjustment layer that allows for the joint refinement of camera poses and dynamic object poses
- Different from the traditional practice of treating dynamic regions as noise in camera pose estimation, this method partially proves that dynamic region information can improve the accuracy of camera pose estimation
- This method can be further improved by using a more lightweight network, adding depth priors to optimization, and adding object shape reconstruction

Reference

[1] Bescos, B., Campos, C., Tardos, J., & Neira, J. (2021). DynaSLAM II: Tightly-Coupled Multi-Object Tracking and SLAM. *IEEE Robotics and Automation Letters, PP, 1-1.*

[2] J. Zhang, M. Henein, R. Mahony, and V. Ila. "VDO-SLAM: a visual dynamic object-aware SLAM system". In: *arXiv preprint arXiv:2005.11052* (2020).

[3] Teed, Z., & Deng, J. (2021). DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras. *Advances in neural information processing systems*.

[4] Ye, W., Yu, X., Lan, X., Ming, Y., Li, J., Bao, H., Cui, Z., & Zhang, G. (2022). DeFlowSLAM: Self-Supervised Scene Motion Decomposition for Dynamic Dense SLAM. *arXiv preprint arXiv:2207.08794*.

[5] Teed, Z., & Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16* (pp. 402–419).

[6] Teed, Z., & Deng, J. (2021). RAFT-3D: Scene Flow using Rigid-Motion Embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[7] Jahedi, A., Mehl, L., Rivinius, M., & Bruhn, A. (2022). Multi-Scale Raft: Combining Hierarchical Concepts for Learning-Based Optical Flow Estimation. In *2022 IEEE International Conference on Image Processing (ICIP)* (pp. 1236-1240).

[8] Gaidon, A., Wang, Q., Cabon, Y., & Vig, E. (2016). Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 4340–4349).

[9] M. Rünz and L. Agapito. (2017). Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. May 2017, pp. 4471–4478.

[10] B. Xu, A. J. Davison, and S. Leutenegger. Learning to Complete Object Shapes for Object-level Mapping in Dynamic Scenes. 2022. arXiv: 2208.05067 [cs.CV].

[11] Andreas Geiger, Philip Lenz, & Raquel Urtasun (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[12] Park, J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[13] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, & Ren Ng (2020). NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

[14] Wang, J., Runz, M., & Agapito, L. (2021). DSP-SLAM: object oriented SLAM with deep shape priors. In *2021 International Conference on 3D Vision (3DV)* (pp. 1362–1371).

[15] Xu, Y., Chai, M., Shi, Z., Peng, S., Skorokhodov Ivan, Siarohin Aliaksandr, Yang, C., Shen, Y., Lee, H.Y., Zhou, B., & Tulyakov Sergy (2022). DiscoScene: Spatially Disentangled Generative Radiance Field for Controllable 3D-aware Scene Synthesis. *arxiv: 2212.11984*.

Thanks!