Master Seminar Robot Perception & Intelligence

Differentiable SLAM

Presenting Student: Kameel Amareen

Advisor: Sebastián Barbas Laina

Prof. Dr. Stefan Leutenegger

WS 2023-2024

Table of Contents

Motivation - Why Deep Learning ?

- Current SLAM Systems lack robustness [1]:
  - Lost Feature Tracks
  - Divergence in the Optimization algorithm
  - Accumulation of Drift

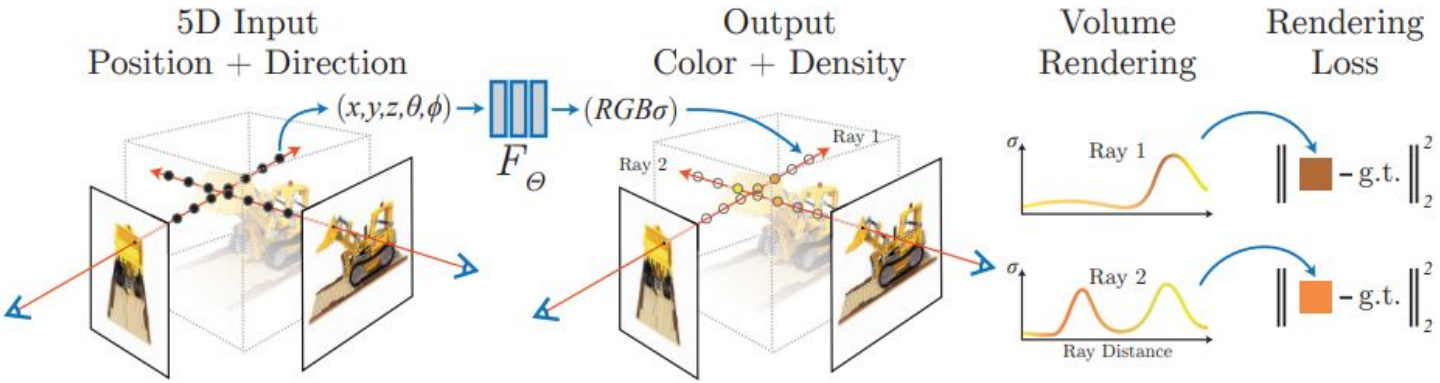
- Deep Learning can be leveraged in various forms:
  - **End-to-End Pose Updates Regression & Feature Estimation** [1]
  - **Using Neural Implicit Representations** [2][3]
  - Using learnt Features [4]

[1] Teed, Zachary, and Jia Deng. "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras." *Advances in neural information processing systems* 34 (2021): 16558-16569.
[2] Zhu, Zihan, et al. "Nice-slam: Neural implicit scalable encoding for slam." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.
[3] Sucar, Edgar, et al. "iMAP: Implicit mapping and positioning in real-time." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.
[4] Sarlin, Paul-Edouard, et al. "Superglue: Learning feature matching with graph neural networks." *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020.

SMART ROBOTICS LAB

Using Neural Implicit Representations

What are Neural Implicit Representations ?



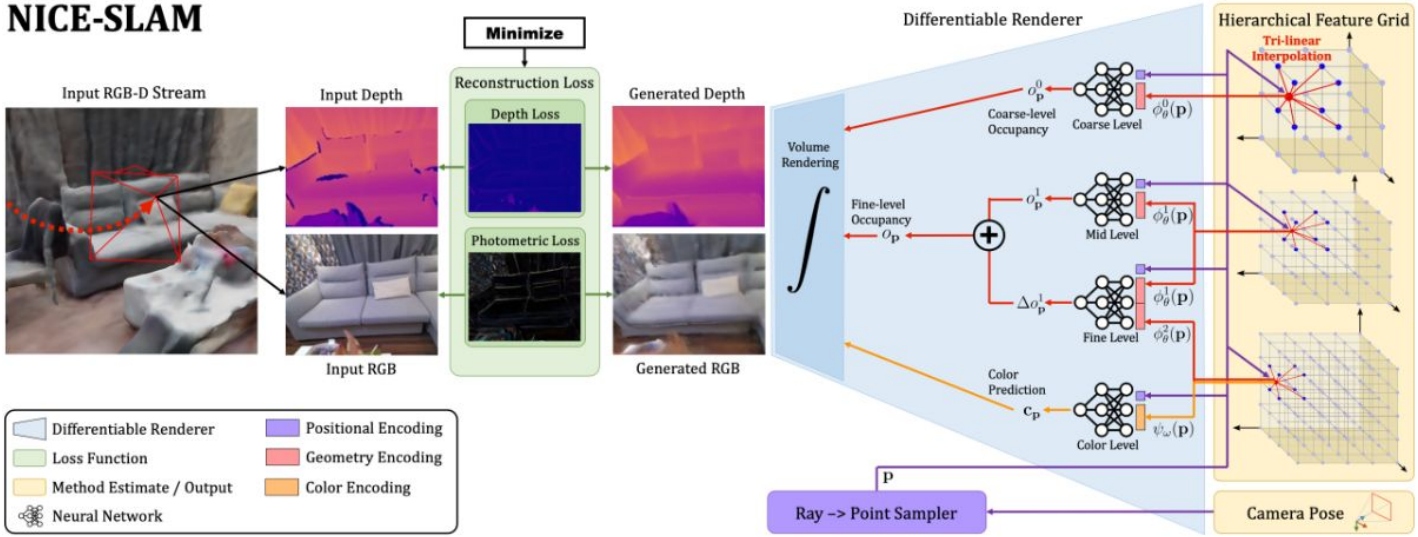
NeRF: Neural Radiance Fields [5]

- Encodes a dense continuous space representation
- Fills mapping gaps/Predicts unseen regions

[5] Mildenhall, Ben, et al. "Nerf: Representing scenes as neural radiance fields for view synthesis." *Communications of the ACM* 65.1 (2021): 99-106.

NICE-SLAM 2022:

Neural Implicit Scalable Encoding for SLAM [2]



- Input: RGB-D Stream
- Hierarchical Scene Representation
  - Large Scene Mapping & Efficient Local Feature Updates
- Parallel Processes for Mapping & Tracking

[2] Zhu, Zihan, et al. "Nice-slam: Neural implicit scalable encoding for slam." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

NICE-SLAM: Methodology

Hierarchical Scene Representation

Enables Local Feature Updates, vital for efficient Large Scene Mapping

Mid & Fine Level Geometric Representation

$$o^1_{\mathbf{p}} = f^1(\mathbf{p}, \phi^1_\theta(\mathbf{p})), \qquad \Delta o^1_{\mathbf{p}} = f^2(\mathbf{p}, \phi^1_\theta(\mathbf{p}), \phi^2_\theta(\mathbf{p}))$$

$$o_{\mathbf{p}} = o^1_{\mathbf{p}} + \Delta o^1_{\mathbf{p}}$$

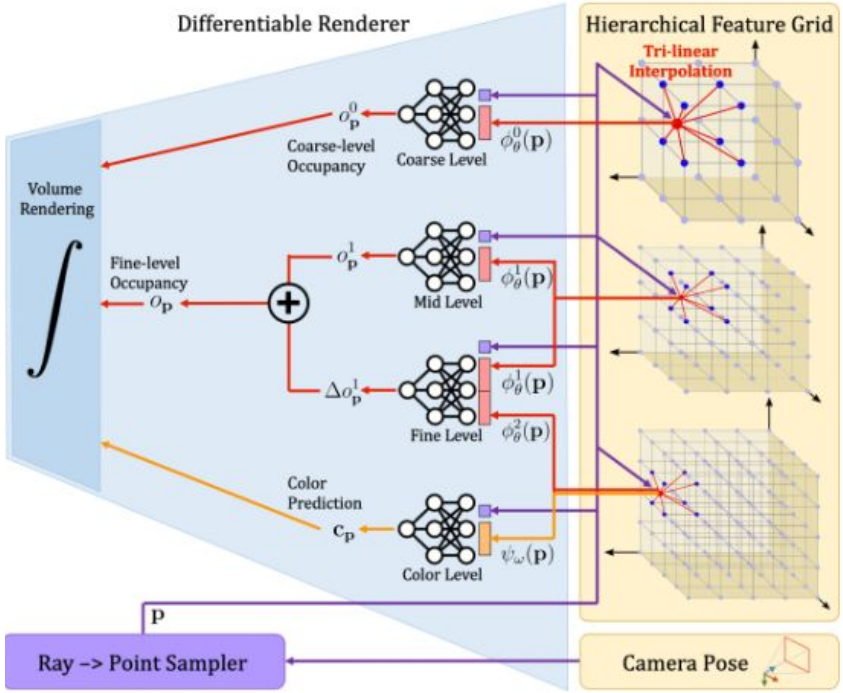
Coarse Level Geometric Representation

$$o^0_{\mathbf{p}} = f^0(\mathbf{p}, \phi^0_\theta(\mathbf{p}))$$

Color Representation

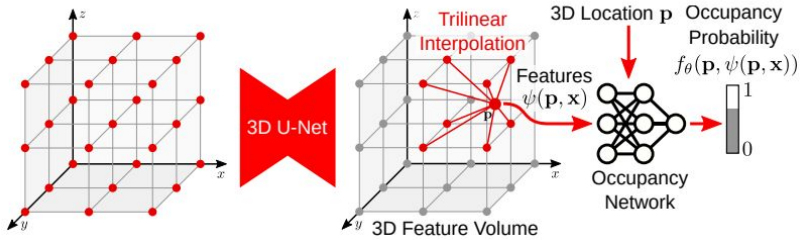
$$\mathbf{c}_{\mathbf{p}} = \mathbf{g}_\omega(\mathbf{p}, \psi_\omega(\mathbf{p}))$$

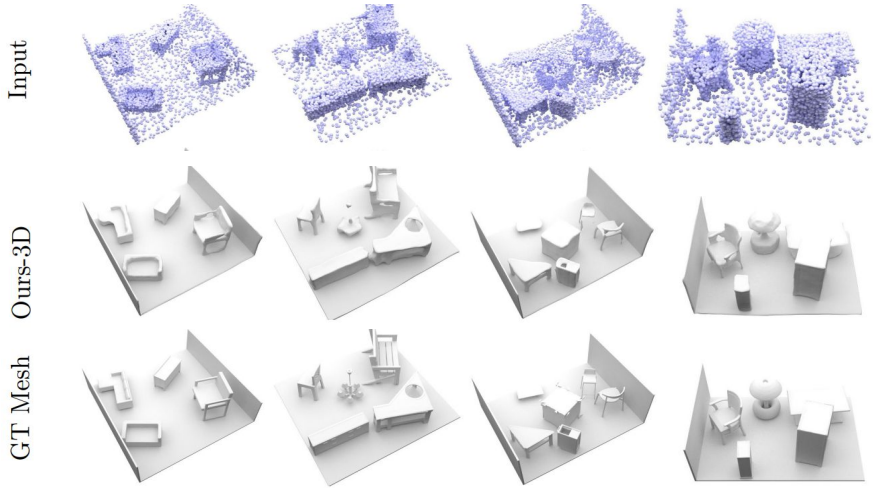
$\phi^0_\theta(\mathbf{p})$	$\phi^1_\theta(\mathbf{p})$	$\phi^2_\theta(\mathbf{p})$	Coarse, Mid & Fine Level Features
$o^0_{\mathbf{p}}$	$o^1_{\mathbf{p}}$	$\Delta o^1_{\mathbf{p}}$	Encoded Coarse, Mid and Residual Occupancies
f^0	f^1	f^2	Pretrained Occupancy Decoder MLPs
$\psi_\omega(\mathbf{p})$	$\mathbf{g}_\omega$	$\mathbf{c}_{\mathbf{p}}$	Color Features, Color Decoder & Color Value



NICE-SLAM: Methodology

ConvONET - Occupancy Feature Decoders [6]





[6] Peng, Songyou, et al. "Convolutional occupancy networks." *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*. Springer International Publishing, 2020.

NICE-SLAM: Methodology

Depth and Color Rendering

Ray-termination Probability Modeling

$$w_i^c = o^0_{\mathbf{p}_i} \prod_{j=1}^{i-1} (1 - o^0_{\mathbf{p}_j}) \qquad w_i^f = o_{\mathbf{p}_i} \prod_{j=1}^{i-1} (1 - o_{\mathbf{p}_j})$$

w_i^c Coarse Ray Termination Probability at point i

w_i^f Fine Ray Termination Probability at point i

Depth & Color Rendering

$$\hat{D}^c = \sum_{i=1}^{N} w_i^c d_i, \quad \hat{D}^f = \sum_{i=1}^{N} w_i^f d_i, \quad \hat{I} = \sum_{i=1}^{N} w_i^f \mathbf{c}_i.$$

$\hat{D}^c$ $\hat{D}^f$ Depth of Rendered Coarse and Fine Rays

d_i $\mathbf{c}_i$ Depth and Color of point i along Ray

$\hat{I}$ RGB value of Rendered Fine Ray

Depth Variance along the Ray

$$\hat{D}^c_{var} = \sum_{i=1}^{N} w_i^c (\hat{D}^c - d_i)^2 \quad \hat{D}^f_{var} = \sum_{i=1}^{N} w_i^f (\hat{D}^f - d_i)^2$$

$\hat{D}^c_{var}$ Depth variance along Rendered Coarse Ray

$\hat{D}^f_{var}$ Depth variance along Rendered Fine Ray

NICE-SLAM: Methodology

Scene Mapping

Mapping Geometric Loss:

$$\mathcal{L}_g^l = \frac{1}{M}\sum_{m=1}^{M}\left|D_m - \hat{D}_m^l\right|, \quad l \in \{c, f\}$$

Mapping Photometric Loss:

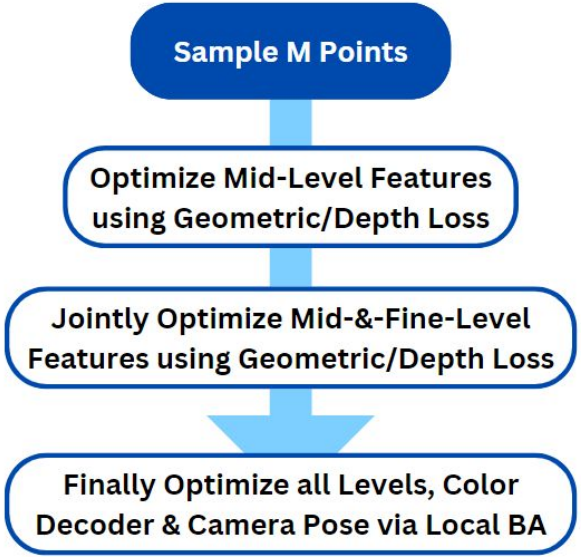
$$\mathcal{L}_p = \frac{1}{M}\sum_{m=1}^{M}\left|I_m - \hat{I}_m\right|$$

Mapping Loss Function:

$$\min_{\theta,\omega,\{\mathbf{R}_i,\mathbf{t}_i\}} (\mathcal{L}_g^c + \mathcal{L}_g^f + \lambda_p \mathcal{L}_p)$$

D_m $\hat{D}_m^l$ Measured and Rendered Depth images (L is either Coarse or Fine Level)

I_m $\hat{I}_m$ Measured and Rendered RGB images



Due to Hierarchical Feature Grid local feature updates is feasible:

- Geometry outside of View remain static
- Very Efficient

NICE-SLAM: Methodology

Camera Tracking

Tracking Geometric Loss:

$$\mathcal{L}_{g\_var} = \frac{1}{M_t} \sum_{m=1}^{M_t} \frac{\left| D_m - \hat{D}_m^c \right|}{\sqrt{\hat{D}_{var}^c}} + \frac{\left| D_m - \hat{D}_m^f \right|}{\sqrt{\hat{D}_{var}^f}}.$$

Tracking Loss Function:

$$\min_{\mathbf{R},\mathbf{t}} \left(\mathcal{L}_{g\_var} + \lambda_{pt} \mathcal{L}_p \right)$$

- Coarse features: short-range predictions of the scene geometry helpful in unobserved areas
- Robustness to Dynamic Objects: neglect pixels with high loss values

[2] Zhu, Zihan, et al. "Nice-slam: Neural implicit scalable encoding for slam." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.

NICE-SLAM: Evaluation & Results

Results on Replica Dataset:

	iMAP* [3]	NICE-SLAM
Mem. (MB) ↓	**1.04**	12.02
Depth L1 ↓	7.64	**3.53**
Acc. ↓	6.95	2.85
Comp. ↓	5.33	**3.00**
Comp. Ratio ↑	66.60	**89.33**

Results on Scannet Dataset (ATE cm):

Scene ID	0000	0059	0106	0169	0181	0207	Avg.
iMAP* [3]	55.95	32.06	17.50	70.51	32.10	11.91	36.67
NICE-SLAM	**8.64**	**12.25**	**8.09**	**10.28**	**12.93**	**5.59**	**9.63**

Results on TUM-RGBD Dataset (ATE cm):

	fr1/desk	fr2/xyz	fr3/office
iMAP [3]	4.9	2.0	5.8
iMAP* [3]	7.2	2.1	9.0
NICE-SLAM	2.7	1.8	3.0
ORB-SLAM2 [7]	**1.6**	**0.4**	**1.0**

Computation & Runtime (Small Scenes):

	FLOPs [$\times 10^3$]↓	Tracking [ms]↓	Mapping [ms]↓
iMAP [3]	443.91	101	448
NICE-SLAM	**104.16**	**47**	**130**

- Desktop PC with a 3.80GHz Intel i7-10700K CPU and an NVIDIA RTX 3090 GPU

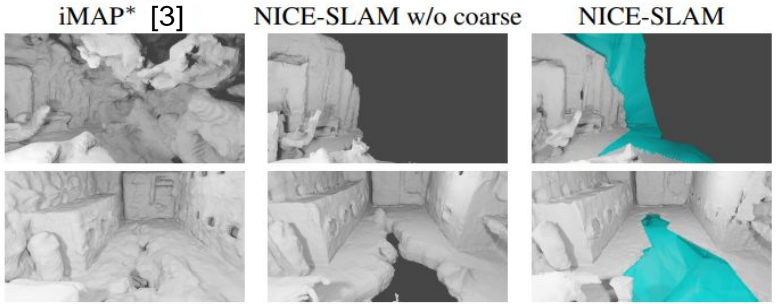
[2] Zhu, Zihan, et al. "Nice-slam: Neural implicit scalable encoding for slam." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.
[7] Mur-Artal, Raul, and Juan D. Tardós. "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras." *IEEE transactions on robotics* 33.5 (2017): 1255-1262.

NICE-SLAM: Evaluation & Results

Results on Replica Dataset:

Robustness to Dynamic Objects:

Geometry Forecasting & Hole Filling:



[2] Zhu, Zihan, et al. "Nice-slam: Neural implicit scalable encoding for slam." *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022.
[3] Sucar, Edgar, et al. "iMAP: Implicit mapping and positioning in real-time." *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021.

NICE-SLAM: Limitations & Future Work

- Feature grid dimensions need to be tuned for good results
- Method does not perform Loop Closures
- Traditional Methods are still better performing at tracking
- Lacks Real-Time capability with large scenes, if lots of Optimization Iterations are required [8]

Other works building on NICE-SLAM [8]:

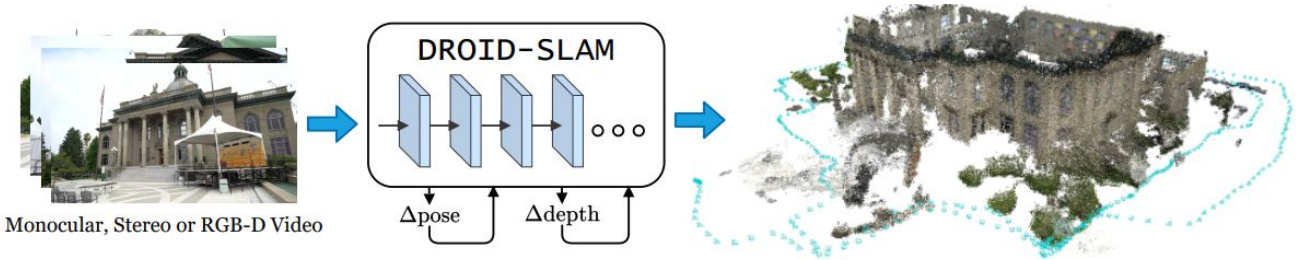
- Includes Motion Data via IMU into optimization
- Introduces a sphere bounded background model

[8] Lisus, Daniil, Connor Holmes, and Steven Waslander. "Towards open world nerf-based slam." *2023 20th Conference on Robots and Vision (CRV)*. IEEE, 2023.

End-to-End Pose Updates Regression & Feature Estimation

DROID-SLAM 2021:

Deep Visual SLAM for Monocular, Stereo & RGB-D Cameras [1]



Inputs Supported:

- Monocular, RGB-D & Stereo Streams

Main Idea:

- Regresses optical flow to define geometrical residuals for pose refinement

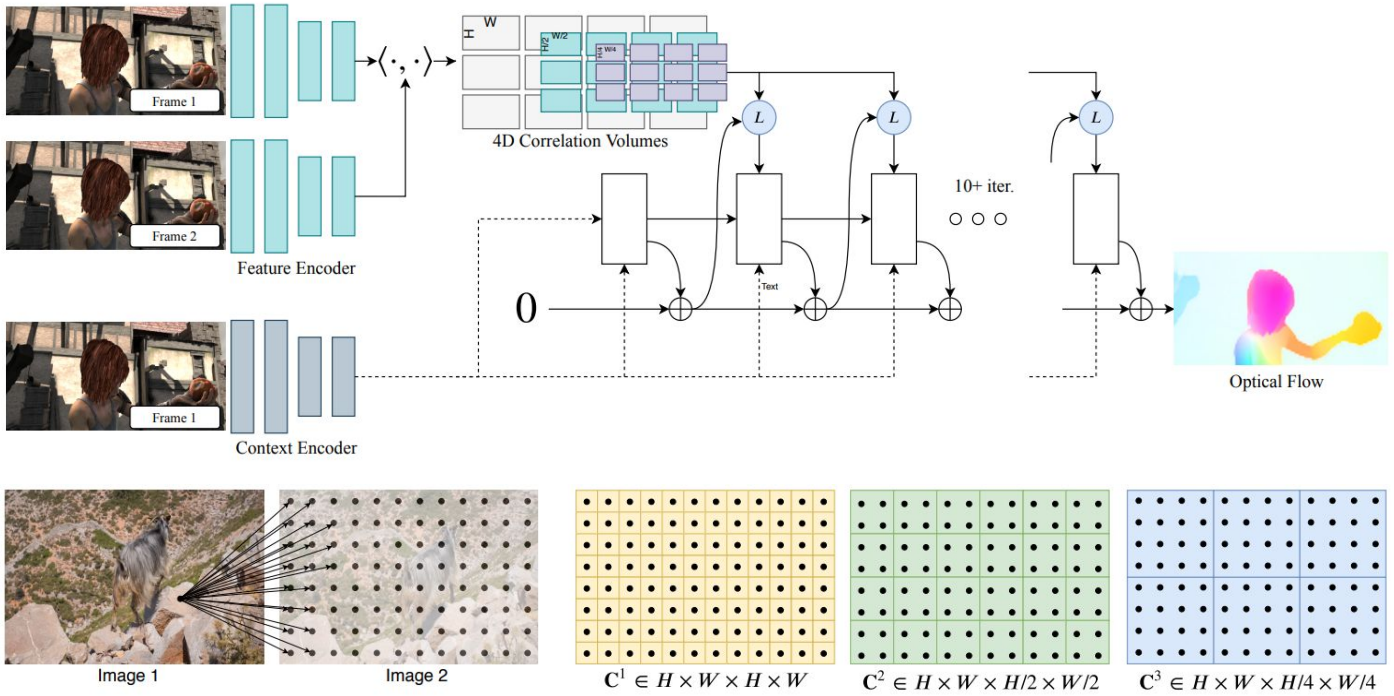
Optical Flow Estimation based on:

- RAFT [8] : Recurrent All-Pairs Field Transforms for Optical Flow

[1] Teed, Zachary, and Jia Deng. "Droid-slam: Deep visual slam for monocular, stereo, and rgb-d cameras." *Advances in neural information processing systems* 34 (2021): 16558-16569.
[8] Teed, Zachary, and Jia Deng. "Raft: Recurrent all-pairs field transforms for optical flow." *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer International Publishing, 2020.

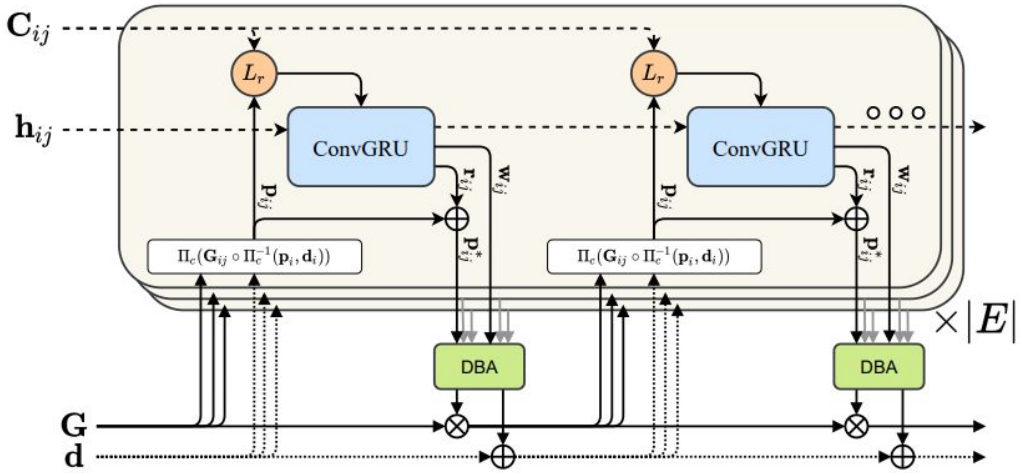
DROID-SLAM Methodology:

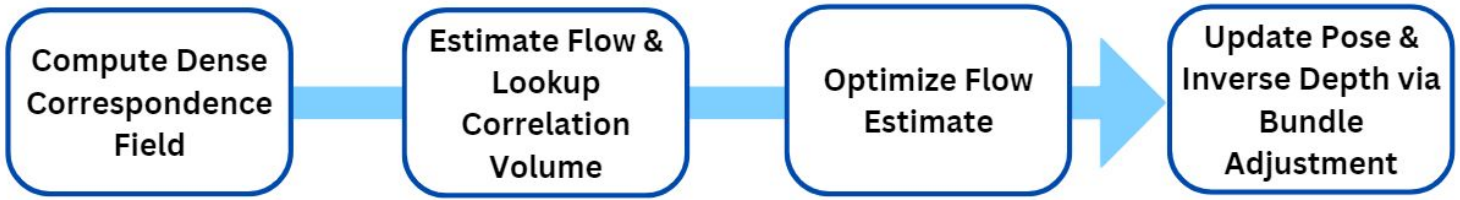
RAFT - Optical Flow Regression [8]



DROID-SLAM: Methodology

General Concept



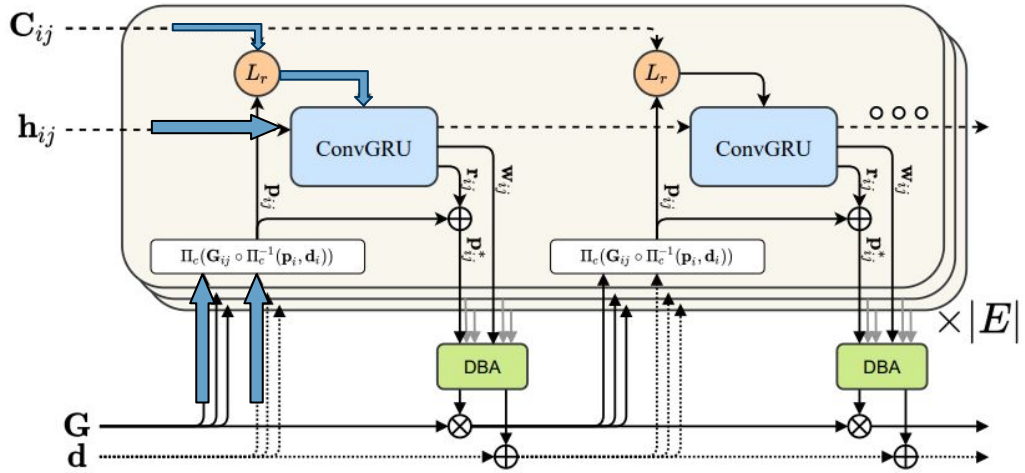


DROID-SLAM: Methodology

Flow Field Update Operator

Correspondence Field between Frames i and j

Hidden state



Dense Correspondence Field

$$\mathbf{p}_{ij} = \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}_i))$$

$$\mathbf{G}_{ij} = \mathbf{G}_j \circ \mathbf{G}_i^{-1}$$

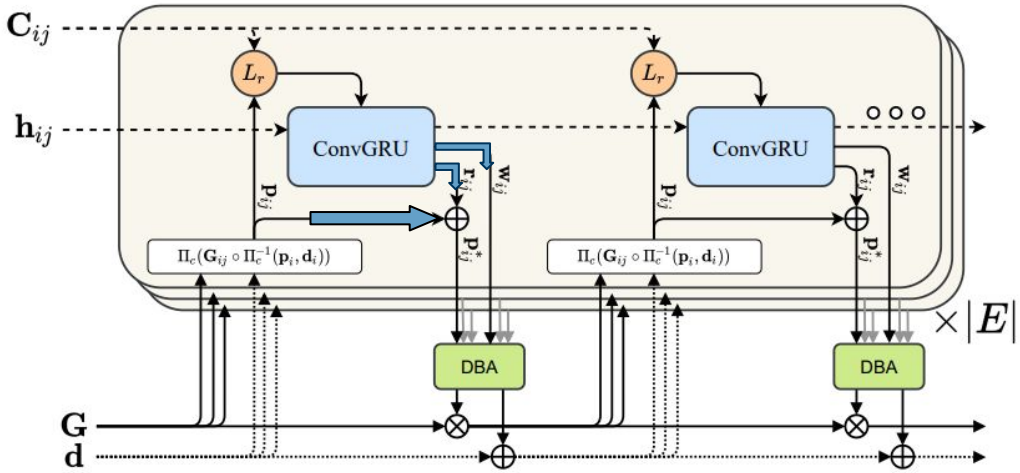
Flow Estimate:

$$\mathbf{p}_{ij} - \mathbf{p}_j$$

Π_c	Camera Intrinsics
$\mathbf{G}_i$	Camera pose at frame i
$\mathbf{p}_i$	Grid of Pixel Coordinates
$\mathbf{d}_i$	Inverse depth at frame i

DROID-SLAM: Methodology

Flow Field Update Operator



Input to ConvGRU:

- Flow Field
- Lookup Features
- Context Features

Outputs:

$$\mathbf{p}^*_{ij} = \mathbf{r}_{ij} + \mathbf{p}_{ij}$$

$$\mathbf{w}_{ij} \in \mathbb{R}^{H \times W \times 2}_{+}$$

$\mathbf{p}^*_{ij}$	Updated Dense Correspondence Field
$\mathbf{r}_{ij}$	Revision Flow Field
$\mathbf{w}_{ij}$	Confidence Map

DROID-SLAM: Methodology

Dense Bundle Adjustment

Maps the set of flow revisions into a set of pose and pixel wise depth updates

$$\mathbf{E}(\mathbf{G}', \mathbf{d}') = \sum_{(i,j)\in\mathcal{E}} \left\| \mathbf{p}_{ij}^{*} - \Pi_c(\mathbf{G}'_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}'_i)) \right\|_{\Sigma_{ij}}^2 \qquad \Sigma_{ij} = \operatorname{diag} \mathbf{w}_{ij} \qquad \mathbf{G}'_{ij} \;\; \mathbf{d}'_i$$

Updated relative pose & inverse depth

- Norm is Mahalanobis Distance weighted by Confidence Map

Use Gauss Newton to solve for

$$\Delta\boldsymbol{\xi}, \Delta\mathbf{d}$$

$\Delta\boldsymbol{\xi}$ Pose Update in Log Space

$\Delta\mathbf{d}$ Inverse Depth Update

Apply Update

$$\mathbf{G}^{(k+1)} = \operatorname{Exp}(\Delta\boldsymbol{\xi}^{(k)}) \circ \mathbf{G}^{(k)}, \qquad \mathbf{d}^{(k+1)} = \Delta\mathbf{d}^{(k)} + \mathbf{d}^{(k)}$$

DROID-SLAM: Methodology

Training

Removing Gauge Freedom

- Fix first two poses to the ground-truth poses of each training sequence

Training Videos (TartainAir Dataset)

- Optical Flow between covisible frame precomputed
- Sampling paths between 8px and 96px

Supervision

- L2 Optical Flow Loss
- Pose Loss

$$\mathcal{L}_{pose} = \sum_i || \text{Log}_{SE3}(\mathbf{T}_i^{-1} \cdot \mathbf{G}_i)||_2$$

$\mathbf{T}_i^{-1}$ Ground Truth Pose

DROID-SLAM: Methodology

SLAM System

Two Threads are running asynchronously

- Frontend:
  - New Frames in
  - Feature Extraction
  - Local Bundle Adjustment
- Backend:
  - Global Bundle Adjustment over all frames

RGB-D Compatibility:

- Penalizes squared distance between Predicted & GT Depth

Stereo Compatibility:

- Same system but doubles the frames and fixes relative pose between right and left frames in DBA

DROID: Evaluation & Results

TartanAir Hard Sequences - Showcases Robustness & Accuracy (ATE m)

Monocular	MH000	MH001	MH002	MH003	MH004	MH005	MH006	MH007	Avg
ORB-SLAM	1.30	**0.04**	2.37	2.45	X	X	21.47	2.73	-
DeepV2D	6.15	2.12	4.54	3.89	2.71	11.55	5.53	3.76	5.03
TartanVO	4.88	0.26	2.00	0.94	1.07	3.19	1.00	2.04	1.92
Ours	**0.08**	0.05	**0.04**	**0.02**	**0.01**	**1.31**	**0.30**	**0.07**	**0.24**

EuRoC - Generalizability to New Cameras (ATE m)

		MH01	MH02	MH03	MH04	MH05	V101	V102	V103	V201	V202	V203	Avg
Deep/Hyb.	DeepFactors	1.587	1.479	3.139	5.331	4.002	1.520	0.679	0.900	0.876	1.905	1.021	2.040
	DeepV2D	0.739	1.144	0.752	1.492	1.567	0.981	0.801	1.570	0.290	2.202	2.743	1.298
	DeepV2D (Tartan Air)	1.614	1.492	1.635	1.775	1.013	0.717	0.695	1.483	0.839	1.052	0.591	1.173
	TartanVO[1]	0.639	0.325	0.550	1.153	1.021	0.447	0.389	0.622	0.433	0.749	1.152	0.680
	D3VO + DSO	-	-	0.08	-	0.09	-	-	0.11	-	0.05	0.19	-
Classical	ORB-SLAM	0.071	0.067	0.071	0.082	0.060	**0.015**	0.020	X	0.021	0.018	X	-
	DSO	0.046	0.046	0.172	3.810	0.110	0.089	0.107	0.903	0.044	0.132	1.152	0.601
	SVO	0.100	0.120	0.410	0.430	0.300	0.070	0.210	X	0.110	0.110	1.080	-
	DSM	0.039	0.036	0.055	0.057	0.067	0.095	0.059	0.076	0.056	0.057	0.784	0.126
	ORB-SLAM3	0.016	0.027	0.028	0.138	0.072	0.033	0.015	0.033	0.023	0.029	X	-
	Ours (odometry only)	0.163	0.121	0.242	0.399	0.270	0.103	0.165	0.158	0.102	0.115	0.204	0.186
	Ours	**0.013**	**0.014**	**0.022**	**0.043**	**0.043**	0.037	**0.012**	**0.020**	**0.017**	**0.013**	**0.014**	**0.022**

DROID: Evaluation & Results

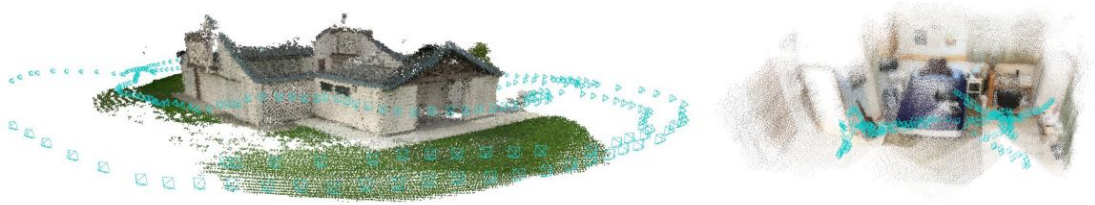
TUM-RGBD - Robustness to Rolling Shutter Artifacts, Motion Blur & Heavy Rotation (ATE m)

	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	avg
ORB-SLAM2	X	0.071	X	0.023	X	X	X	X	0.010	-
ORB-SLAM3	X	**0.017**	0.210	X	0.034	X	X	X	**0.009**	-
DeepTAM	0.111	0.053	0.103	0.206	0.064	0.239	0.093	0.144	0.036	0.116
TartanVO	0.178	0.125	0.122	0.349	0.297	0.333	0.049	0.339	0.062	0.206
DeepV2D	0.243	0.166	0.379	1.653	0.203	0.246	0.105	0.316	0.064	0.375
DeepV2D (TartanAir)	0.182	0.652	0.633	0.579	0.582	0.776	0.053	0.602	0.150	0.468
DeepFactors	0.159	0.170	0.253	0.169	0.305	0.364	0.043	0.601	0.035	0.233
Ours	**0.111**	0.018	**0.042**	**0.021**	**0.016**	**0.049**	**0.026**	**0.048**	0.012	**0.038**

ETH3D-SLAM - RGB-D Performance

Method	AUC (train)	AUC (test)
BundleFusion	84.10	33.84
ElasticFusion	89.06	34.02
RFusion	17.37	51.94
DVO-SLAM	193.89	71.83
ORB-SLAM2	156.10	104.28
BAD-SLAM	280.05	153.47
Ours	**340.42**	**207.79**

DROID: Limitations & Future Work



- Compute Intensive & Low Resolution
  - Real-Time Operation with 2 3090 GPUs
    - EuRoC: 20 FPS @ 320x512
    - TUM-RGBD: 30 FPS @ 240x320
    - Both skipping every other frame
    - TartanAir: 8 FPS hence not realtime
- Did not study robustness against dynamic objects
- Incorporate Scene Semantics while Mapping

Final Comparison & Remarks

NICE-SLAM vs DROID on TUM-RGBD Dataset

Common Scenes (ATE cm):

Method	desk	xyz
NICE-SLAM	2.7	1.8
DROID-SLAM	**1.8**	**1.2**

NICE-SLAM

- Neural Implicit Representation
- Tracking not as good as Classical SLAM
- More-Efficient than predecessor
- Robustness to Dynamic Objects & Performs Hole Filling
- Not Real Time & Requires Feature Grid Tuning
- No Loop Closure

DROID

- Learns to Optimize
- Provides an Explicit Map Representation
- Tracking overperforms Classical SLAM
- High Computation Power needed

[9] Keetha, Nikhil, et al. "Splatam: Splat, track & map 3d gaussians for dense rgb-d slam." *arXiv preprint arXiv:2312.02126* (2023).

Deep Learning + SLAM

Final Remarks:

- On-going Hot Research Field
- Deep Learning Methods provide better Maps
- Doesn’t ensure better tracking accuracy than Classical Methods

New Interesting Paper Suggestion:

- SplaTAM: Splat, Track & Map 3D Gaussians for Dense RGB-D SLAM [9]

[9] Keetha, Nikhil, et al. "Splatam: Splat, track & map 3d gaussians for dense rgb-d slam." *arXiv preprint arXiv:2312.02126* (2023).