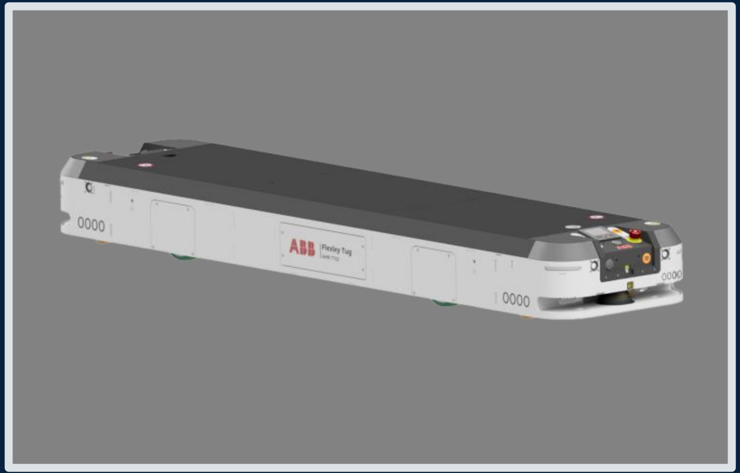
Motivation

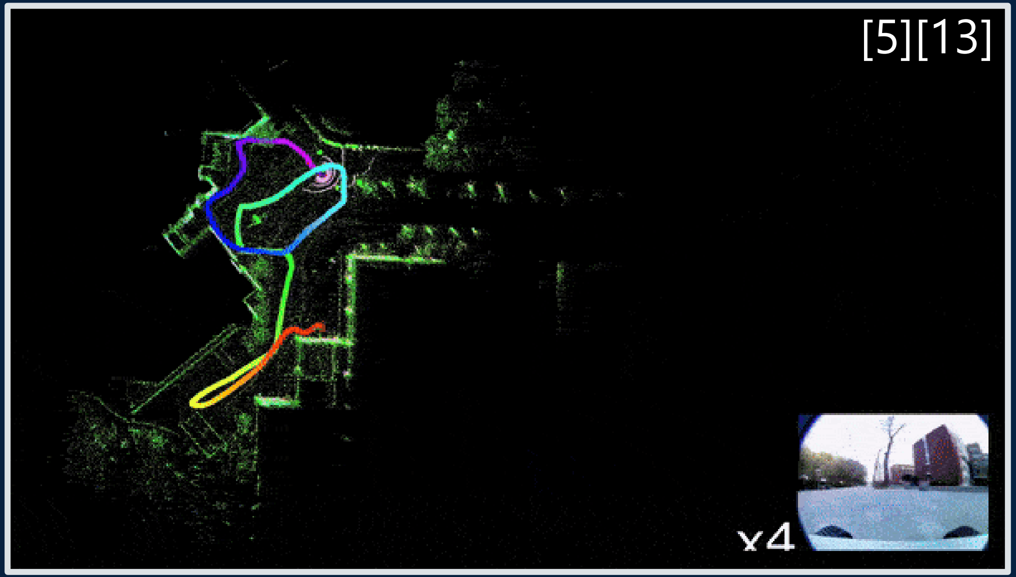


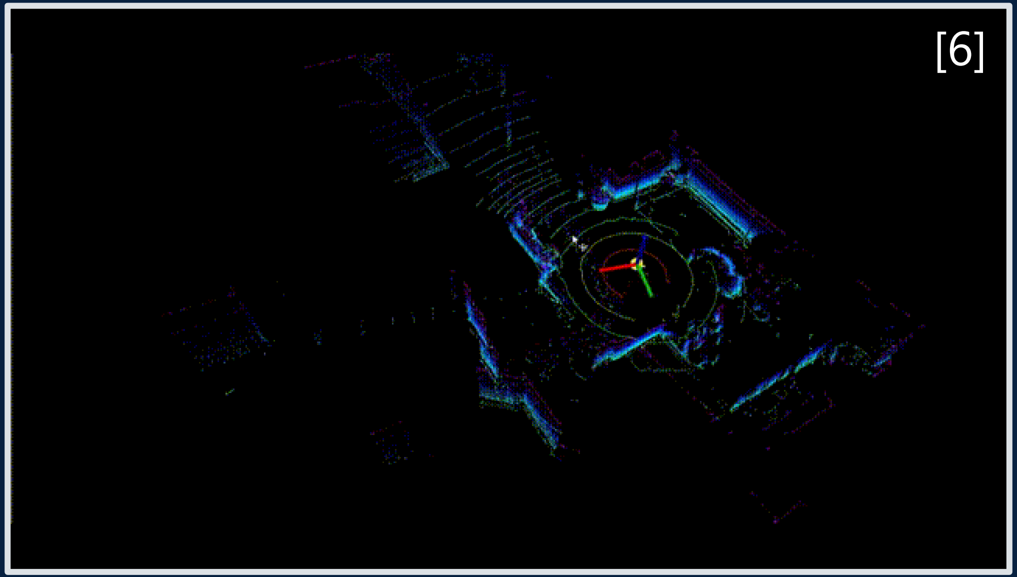
industrial robotics [1]

exploration, search & rescue [2]

autonomous driving [4]

"SLAM is the process by which a mobile robot can build a map of an environment and at the same time use this map to compute it's own location." [15]



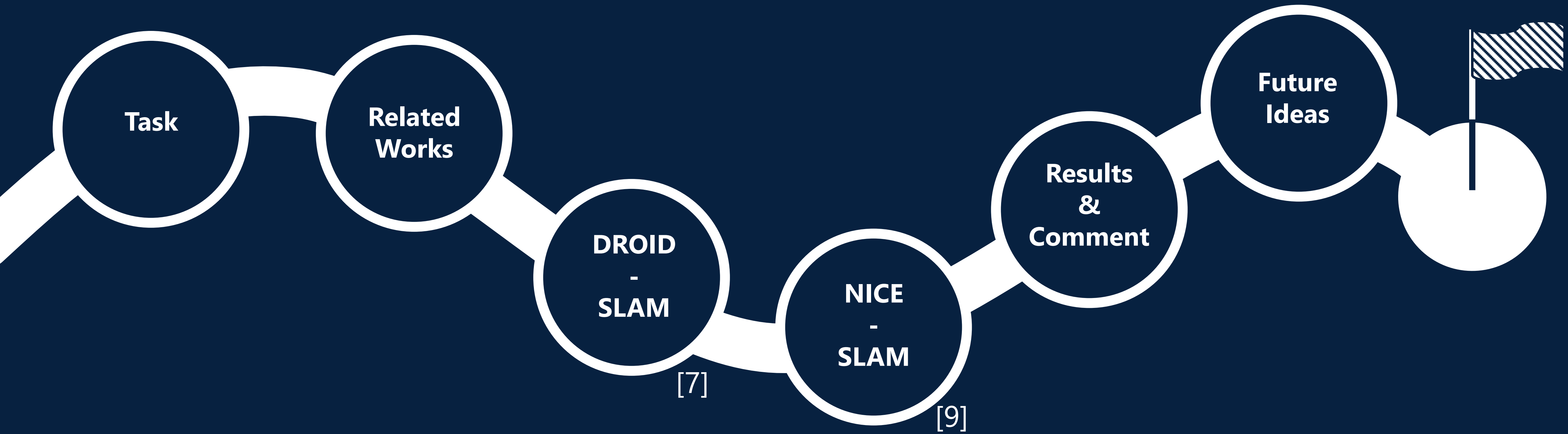


Learning-Based Differentiable SLAM

Seminar: Robot Perception & Intelligence

Alexander Vieres, Advisor: Sebastián Barbas Laina

Structure


Task
Related Works
DROID - SLAM
[7]
NICE - SLAM
[9]
Results & Comment
Future Ideas

Overview on the Task at Hand

unknown environment

models

(noisy) sensors

(approx.) static environment

dynamic position

situation

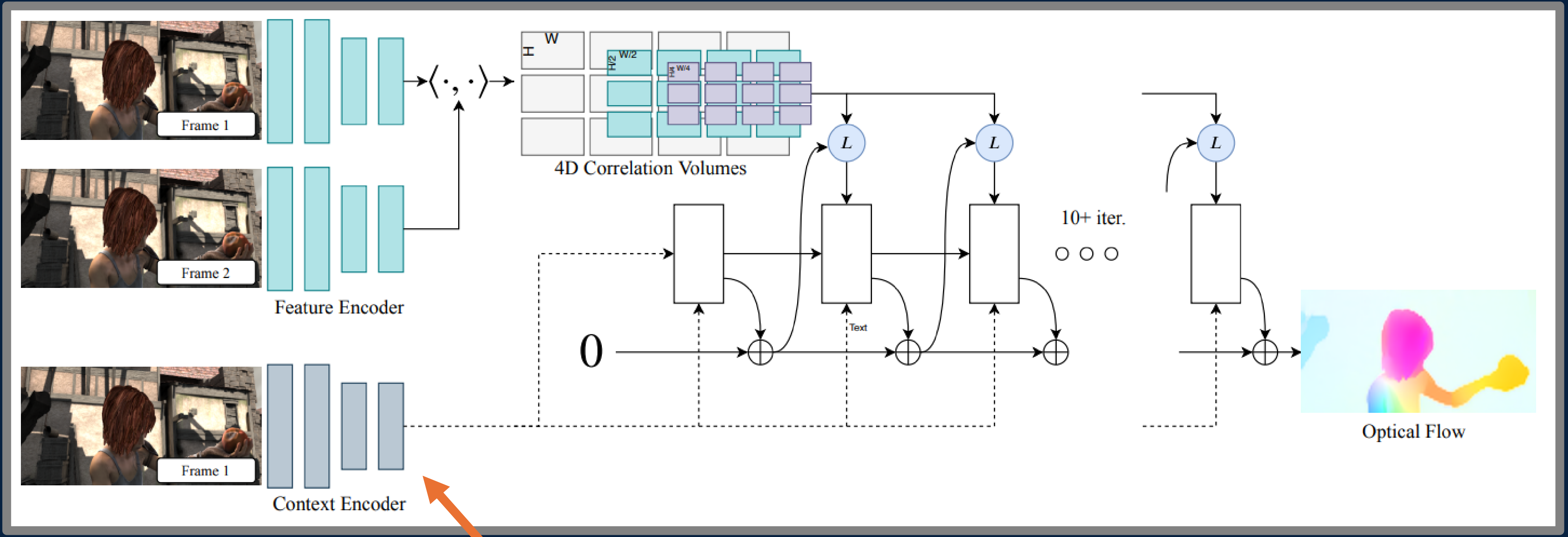
consistent map of the environment

accurate position within

objective

Related Works – RAFT [14] Overview

optical flow estimation

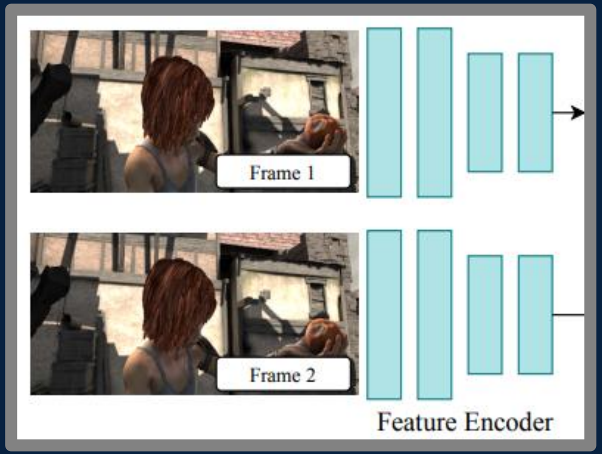


[14]

remember for later

Related Works – RAFT [14] Correlation

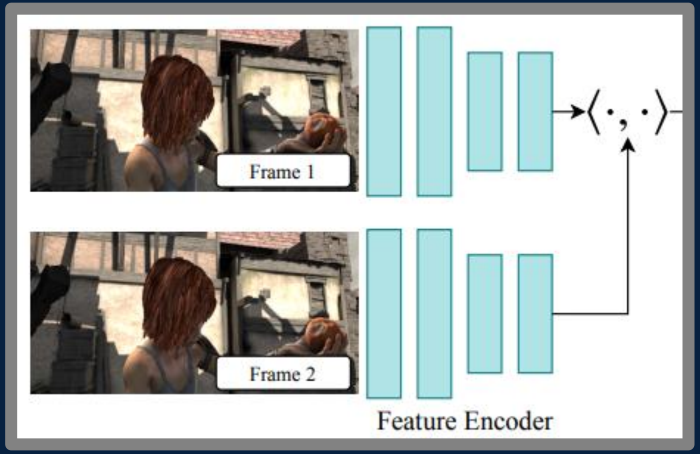
1. neural network encoder network: (HxWx3) → (HxWxD)

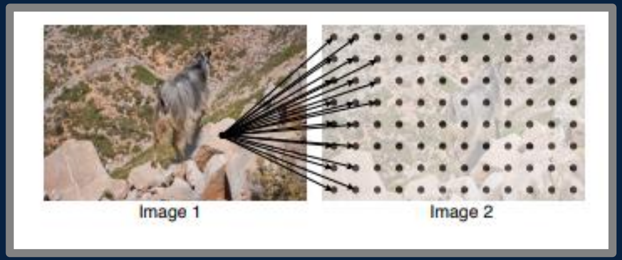


[14]

Related Works – RAFT [14] Correlation

1. neural network encoder network: (HxWx3) → (HxWxD)
2. correlate each feature of each pixel of image 1 to those of image 2



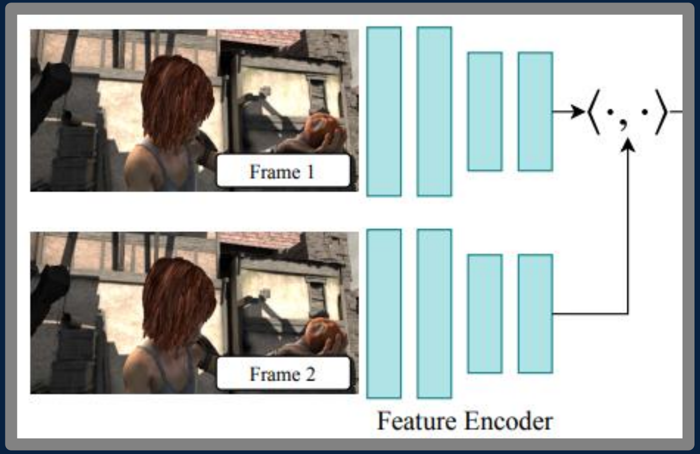


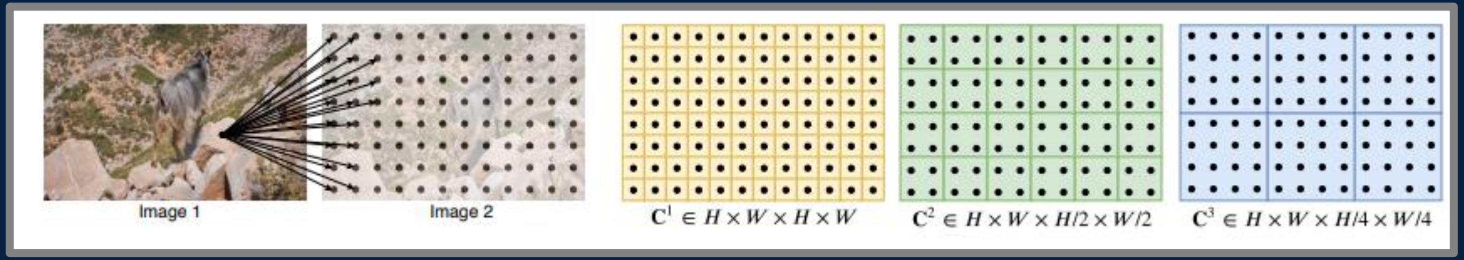
just a scalar product of two neural network outputs

[14]

Related Works – RAFT [14] Correlation

1. neural network encoder network: (HxWx3) → (HxWxD)
2. correlate each feature of each pixel of image 1 to those of image 2
3. pool the feature map, keep features maps before pooling

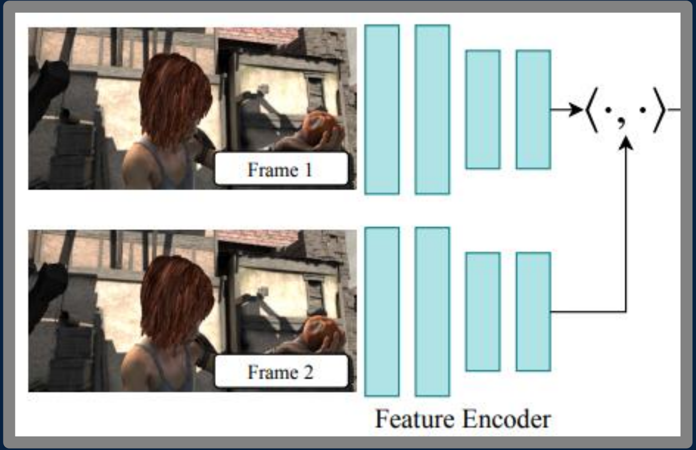


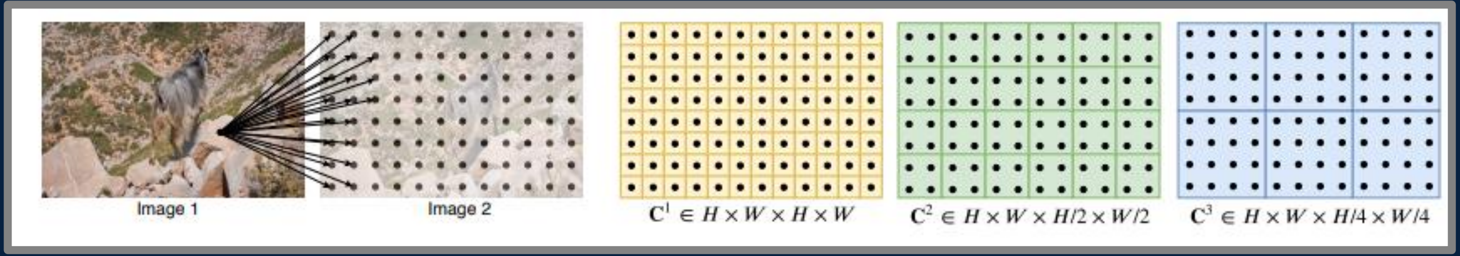


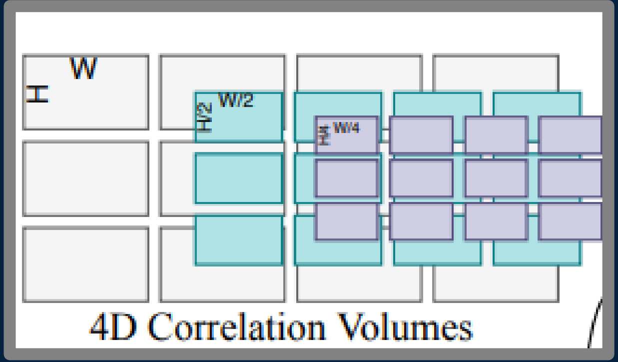
[U]

Related Works – RAFT [14] Correlation

1. neural network encoder network: (HxWx3) → (HxWxD)
2. correlate each feature of each pixel of image 1 to those of image 2
3. pool the feature map, keep features maps before pooling → a **correlation pyramid**

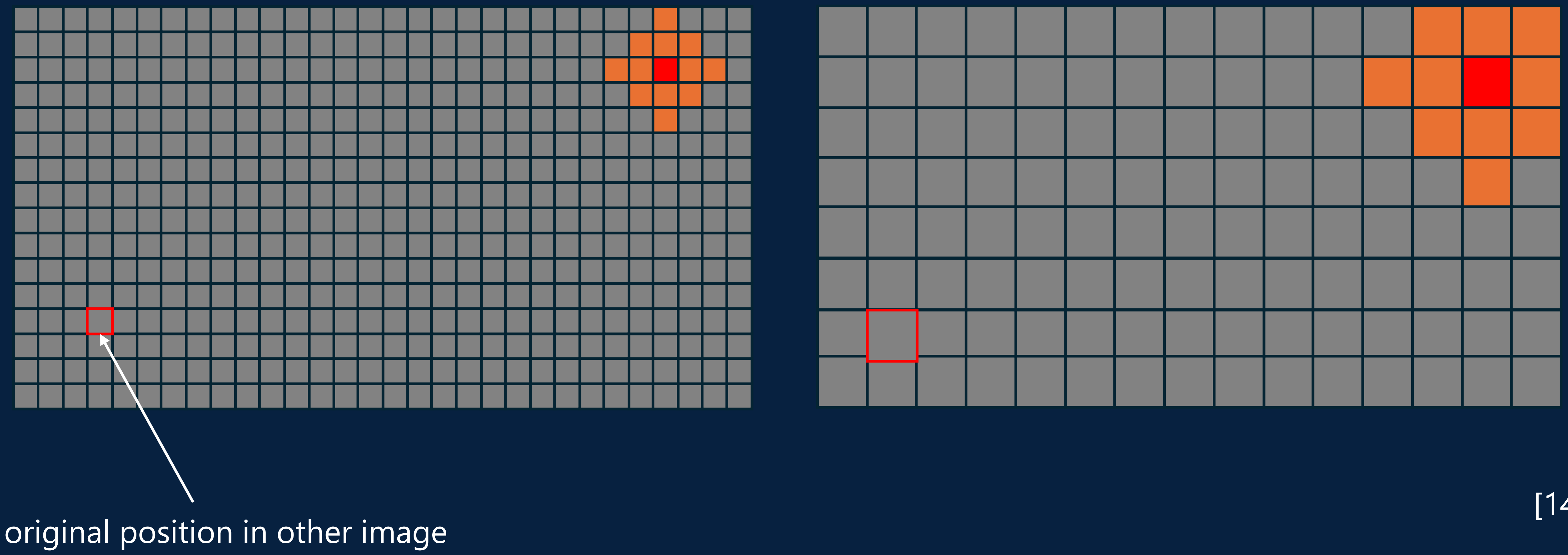






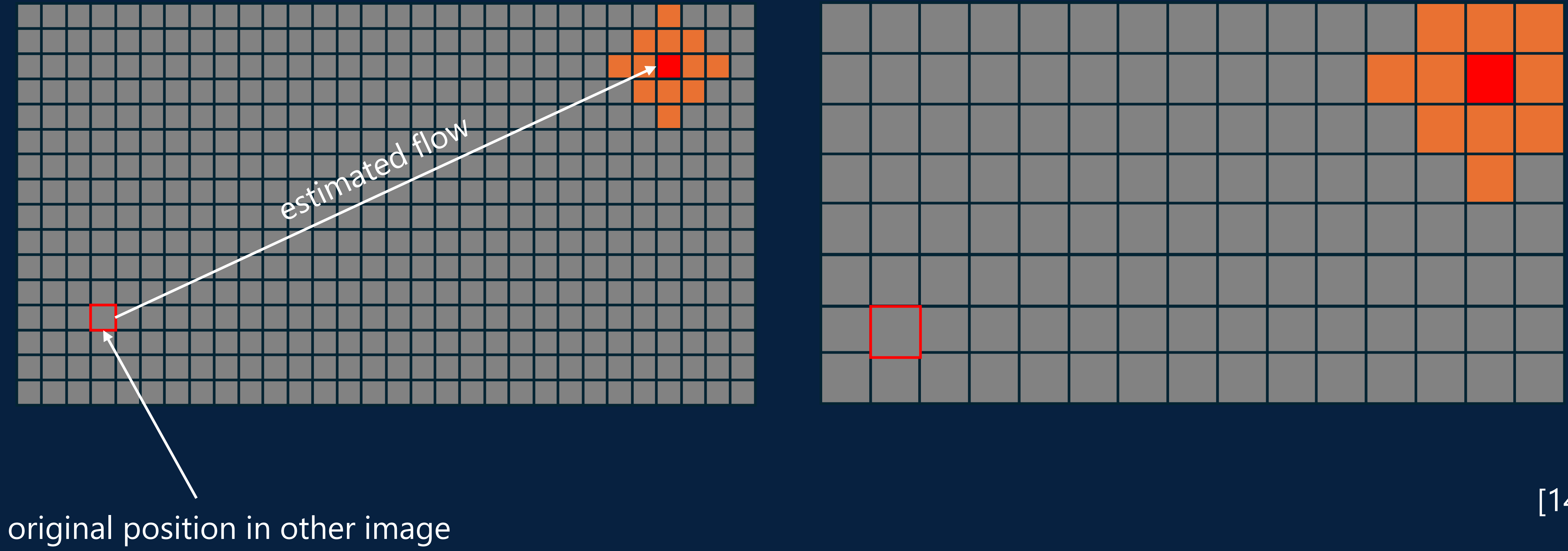
[14]

Related Works – RAFT [14] Lookup



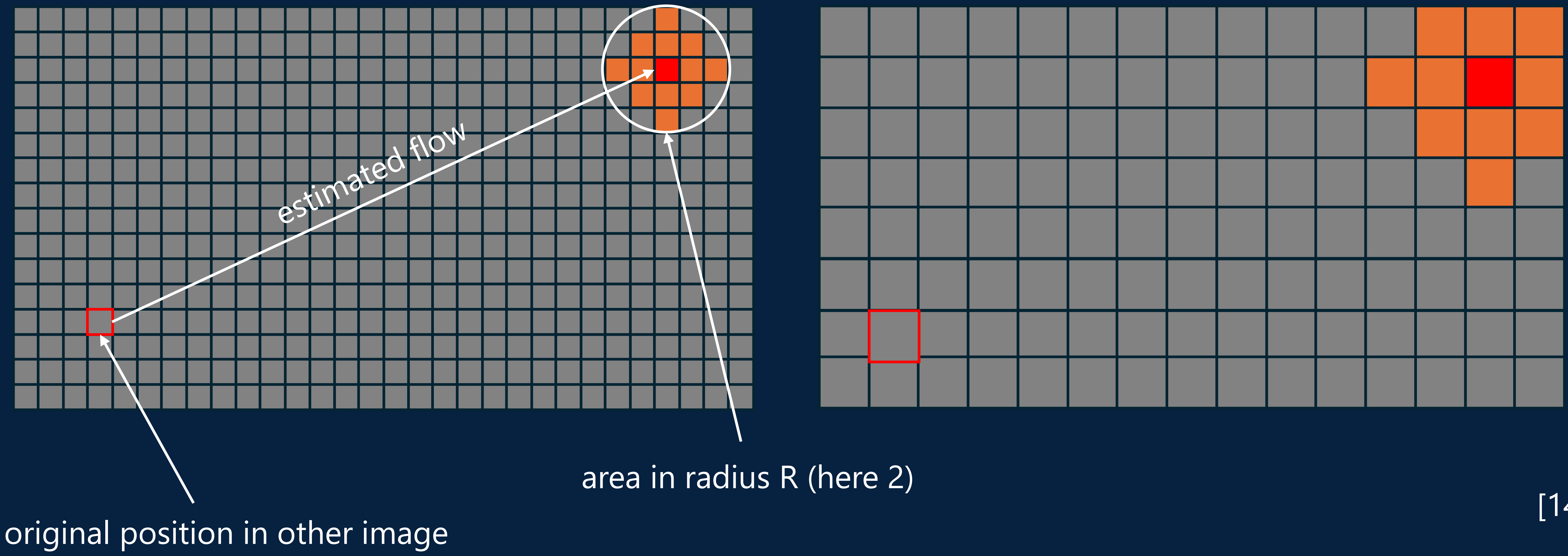
[14]

Related Works – RAFT [14] Lookup



[14]

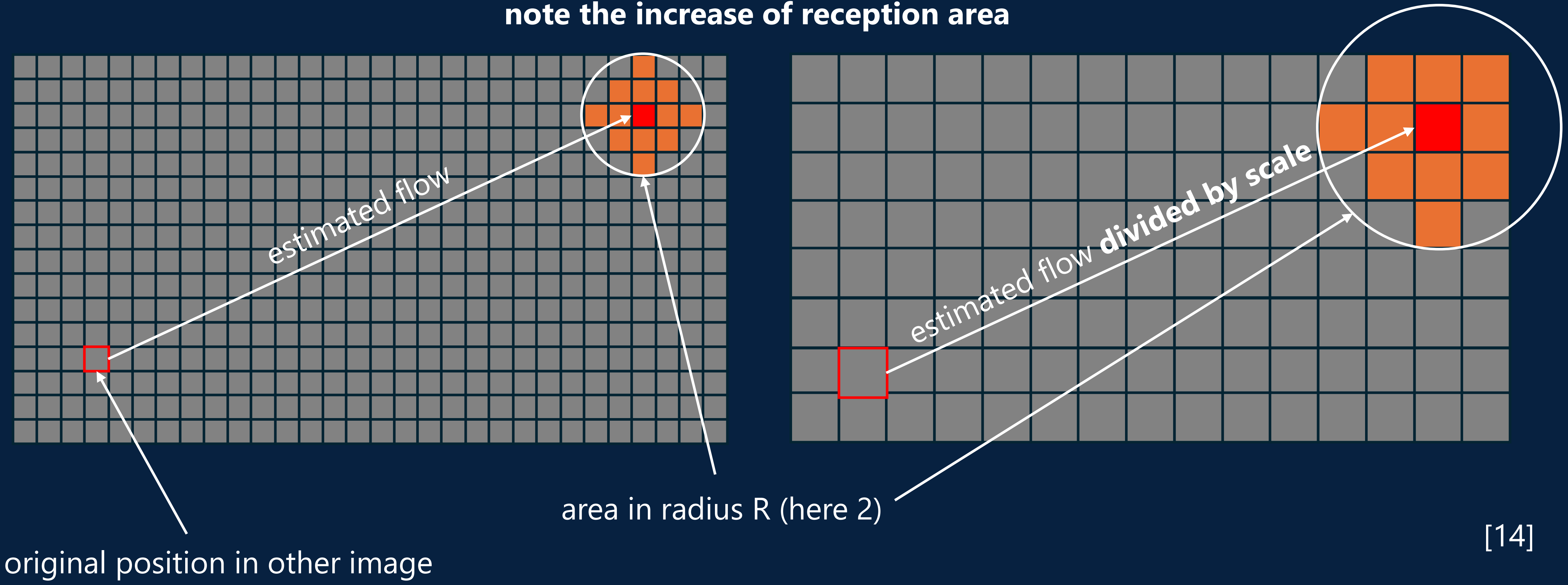
Related Works – RAFT [14] Lookup



[14]

Related Works – RAFT [14] Lookup

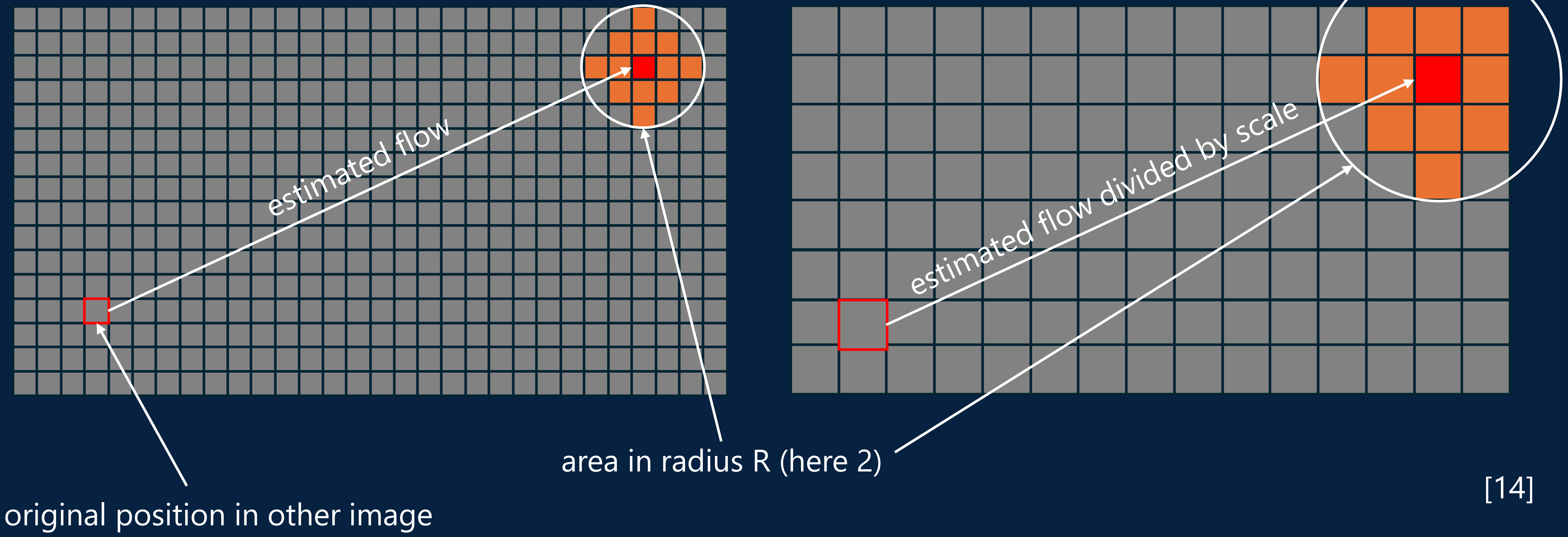
note the increase of reception area



[14]

Related Works – RAFT [14] Lookup

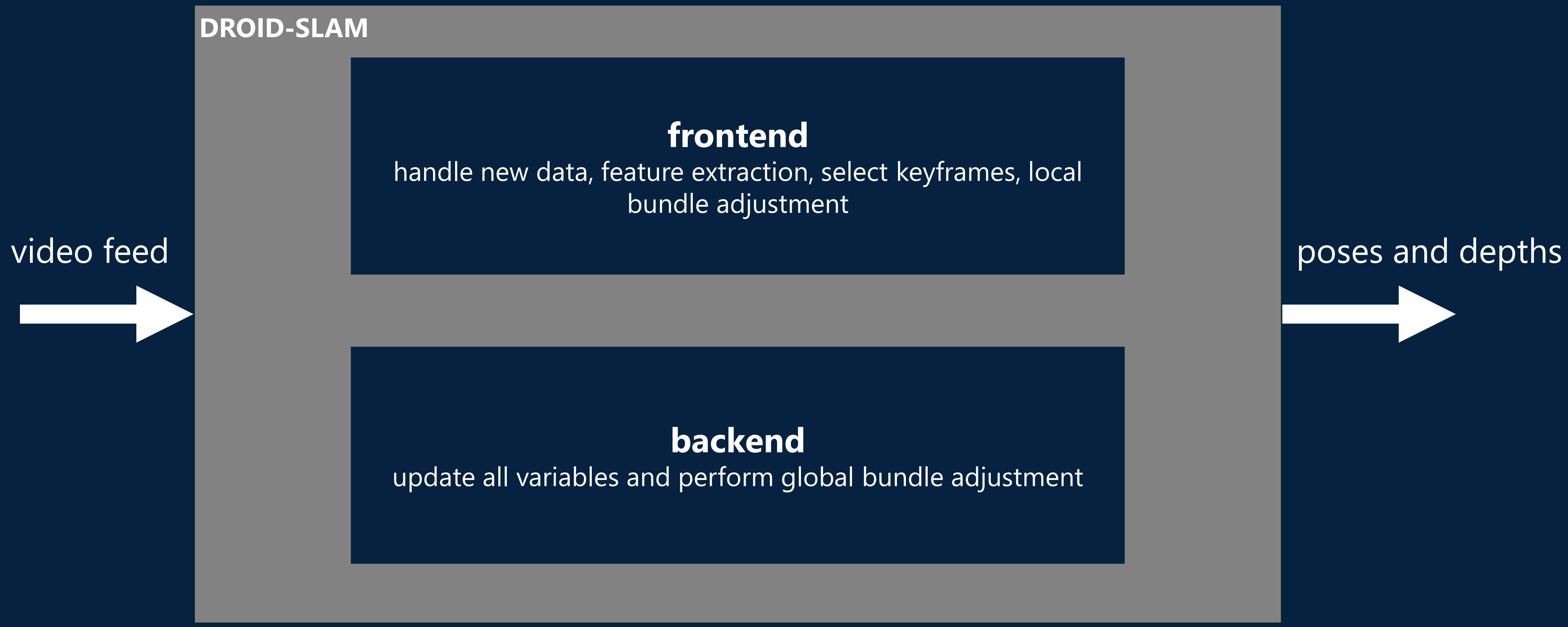
note the increase of reception area



[14]

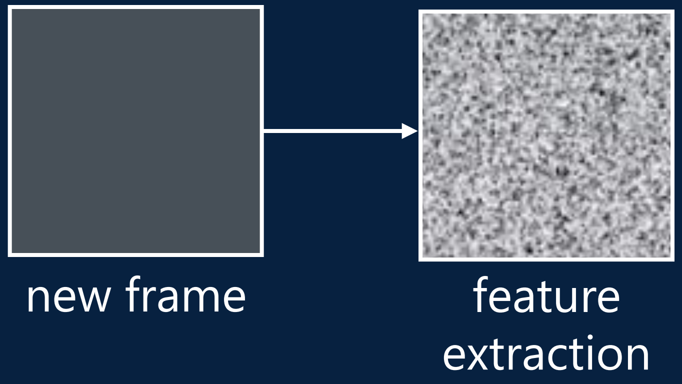
output: concatenate all correlation features

DROID-SLAM [7] – Overview



[7]

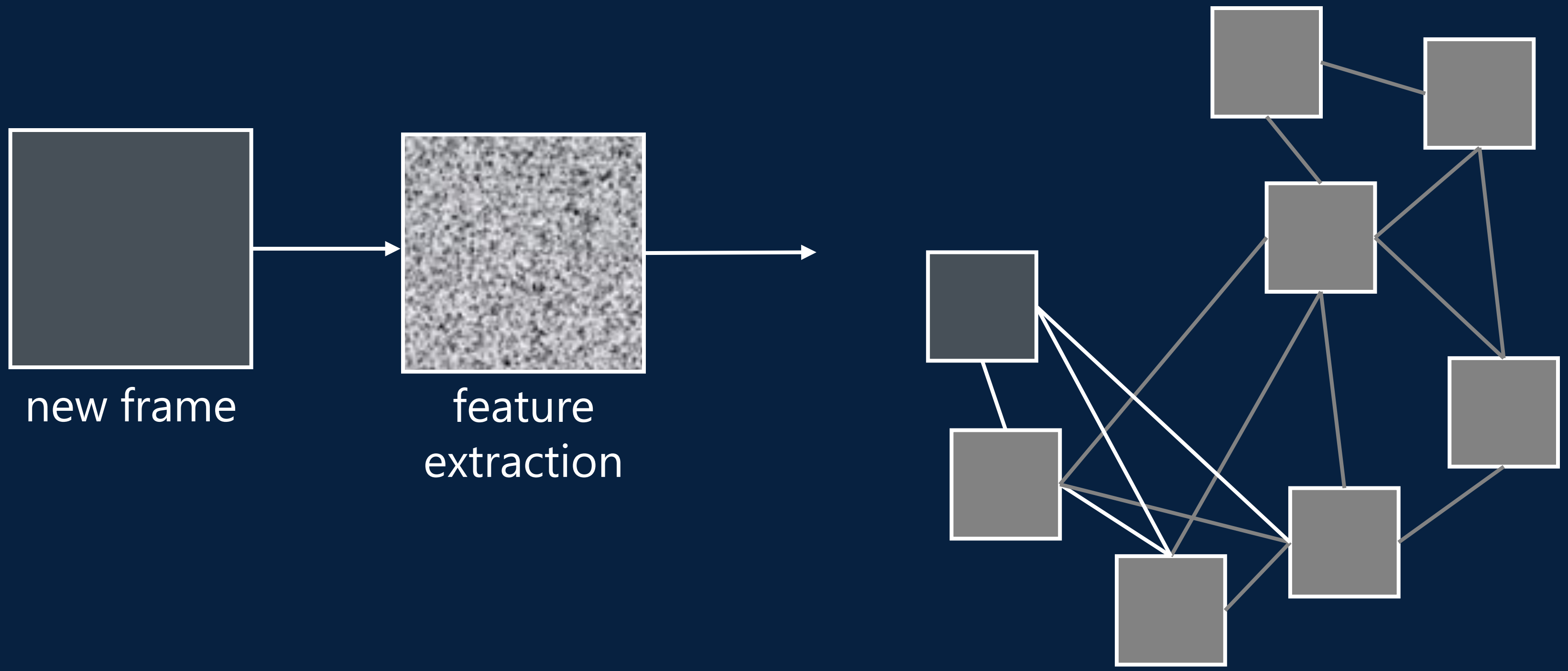
DROID-SLAM [7] – frontend



[7]

DROID-SLAM [7] – frontend

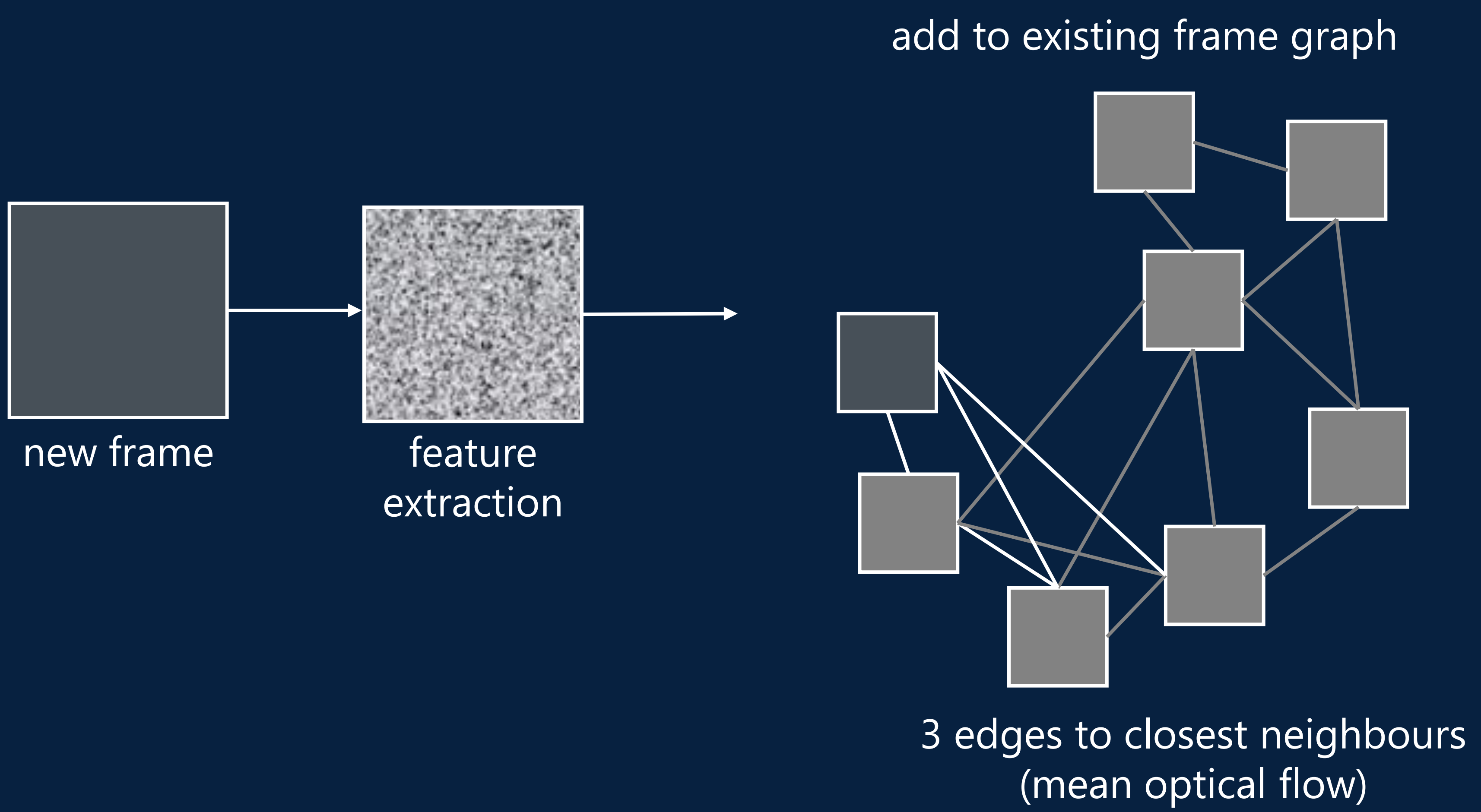
add to existing frame graph



3 edges to closest neighbours
(mean optical flow)

[7]

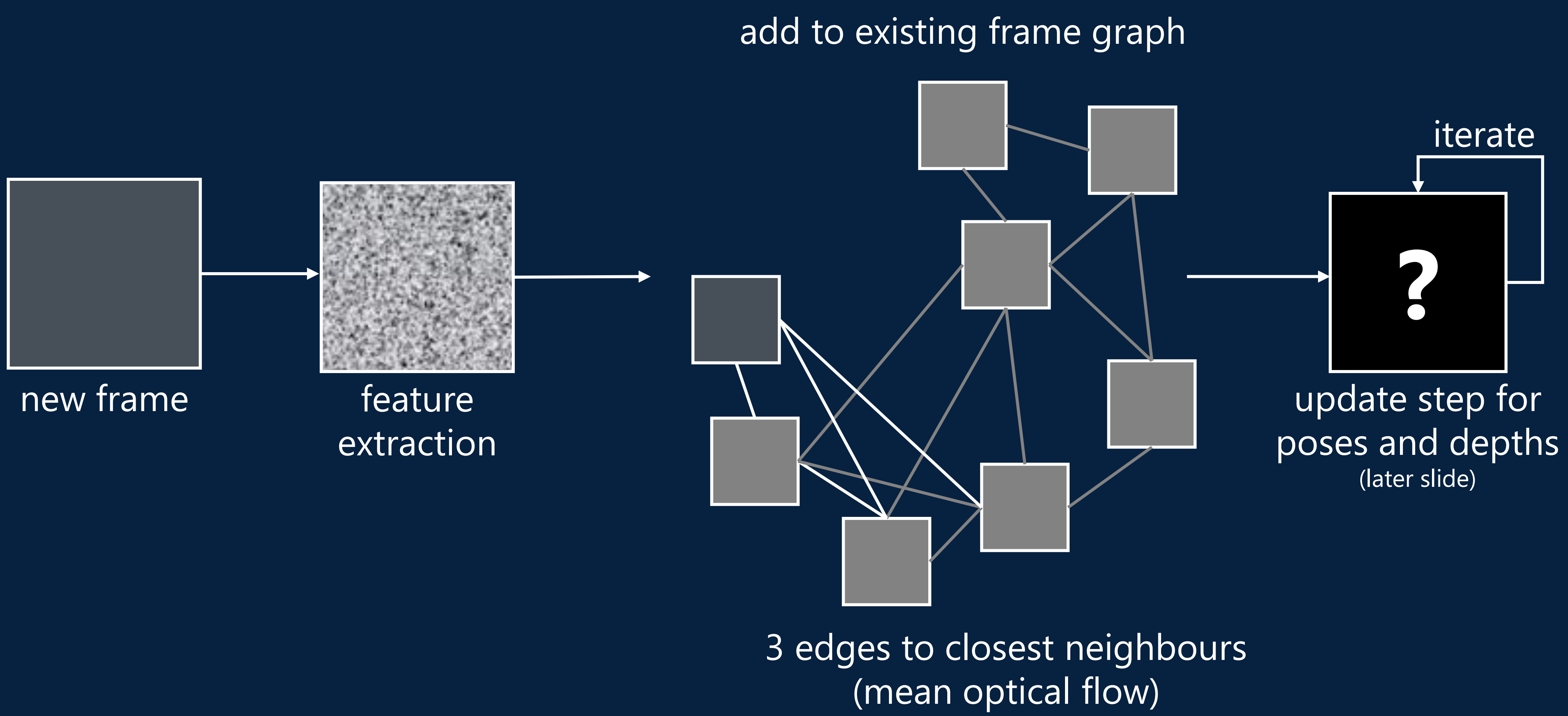
DROID-SLAM [7] – frontend



frame graph
connect images which have covisible features by edges

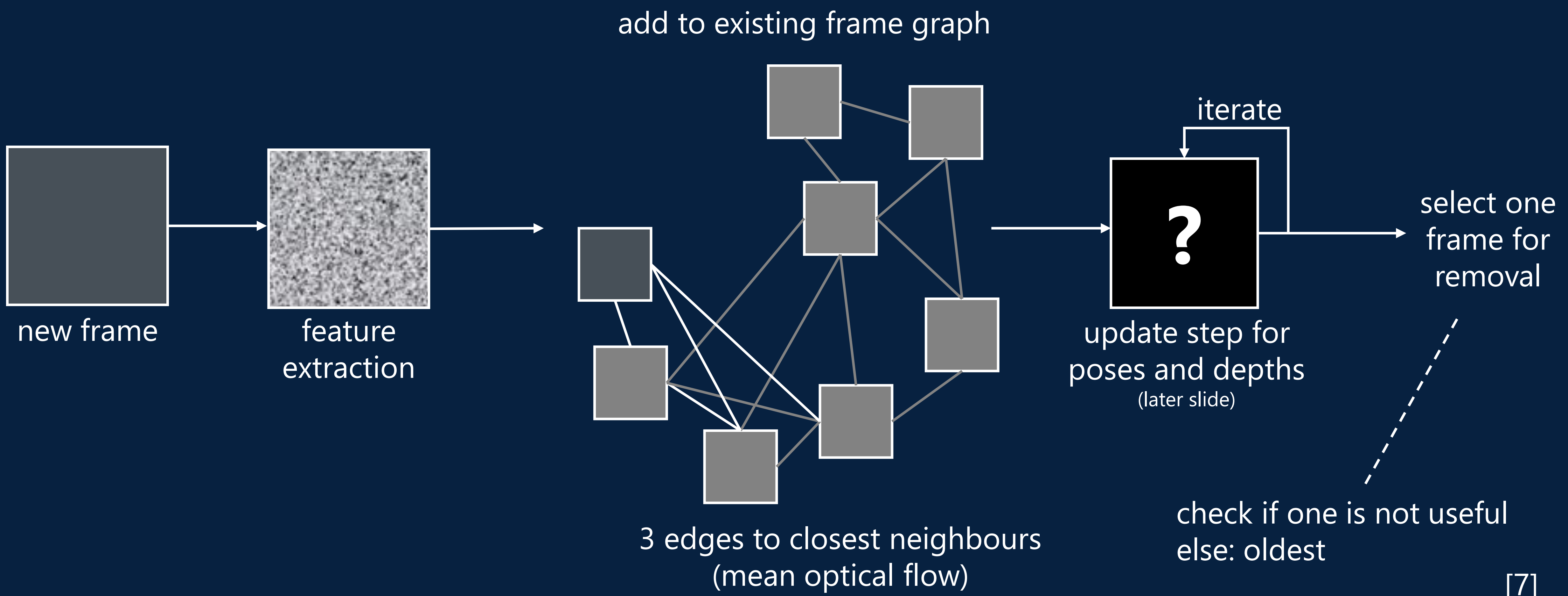
[7]

DROID-SLAM [7] – frontend



[7]

DROID-SLAM [7] – frontend



[7]

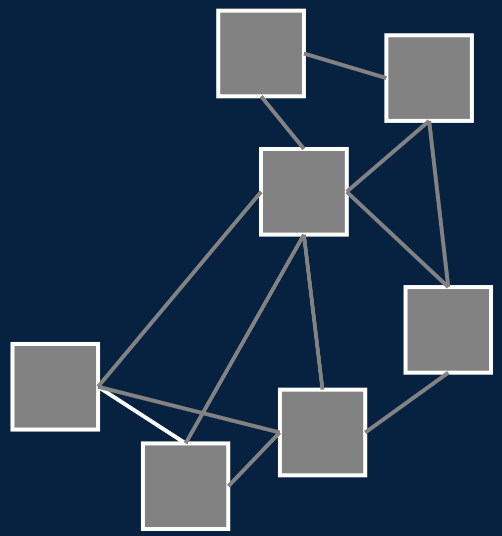
DROID-SLAM [7] – backend

1. stores history of keyframes
2. maintains and updates frame graph
3. applies update operator on whole frame graph

updating the frame graph

first add edges for temporally adjacent keyframes

sample edges from a distance matrix (least first) and prevent connections from edges within a distance of two (in optical flow) of each new connection → sparse graph → **computation!**



Chebyshev distance: $\|(i,j)-(k,l)\|_\infty = \max(|i-k|,|j-l|)$
where i, j, k, l are indexes

[7]

DROID-SLAM [7] – Update Operator

correlation pyramid
remember RAFT [14]

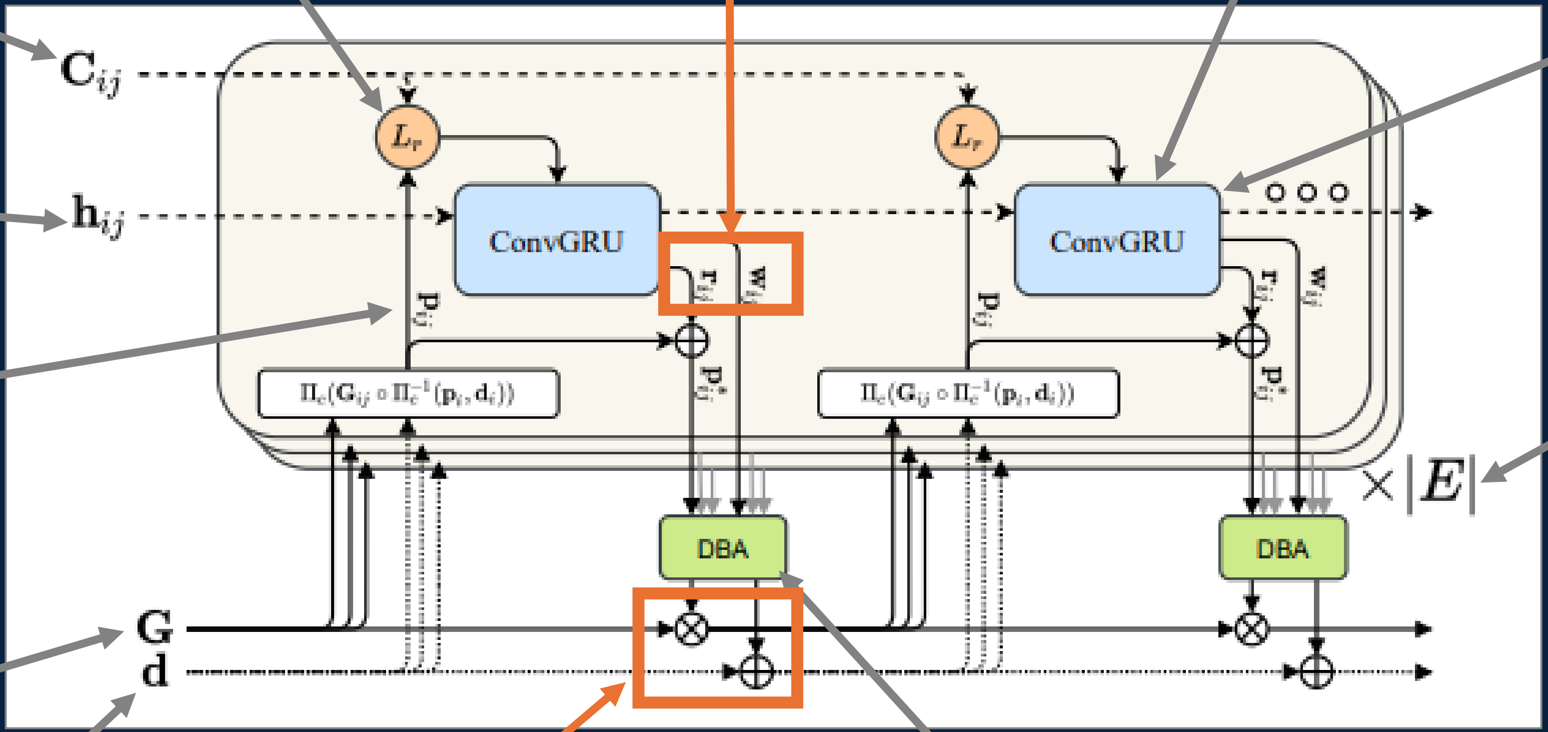
lookup operator
remember RAFT [14]

flow correction and confidence
for each pixel

Gated Recurrent Unit
(next slides)

+ context embedding

hidden state
of the recurrent network



for each edge in the frame graph

correspondence field
(next slide)

reprojection
(next slide)

poses

inverse depths

update
(in two slides)

dense bundle adjustmend
(in two slides)

[7]

DROID-SLAM [7] – Exlanations

reprojection

projection function of camera

pose tranform from frame i to frame j

pixel coordinates $\in \mathbb{R}^{H \times W \times 2}$

inverse depths $\in \mathbb{R}^{H \times W}$

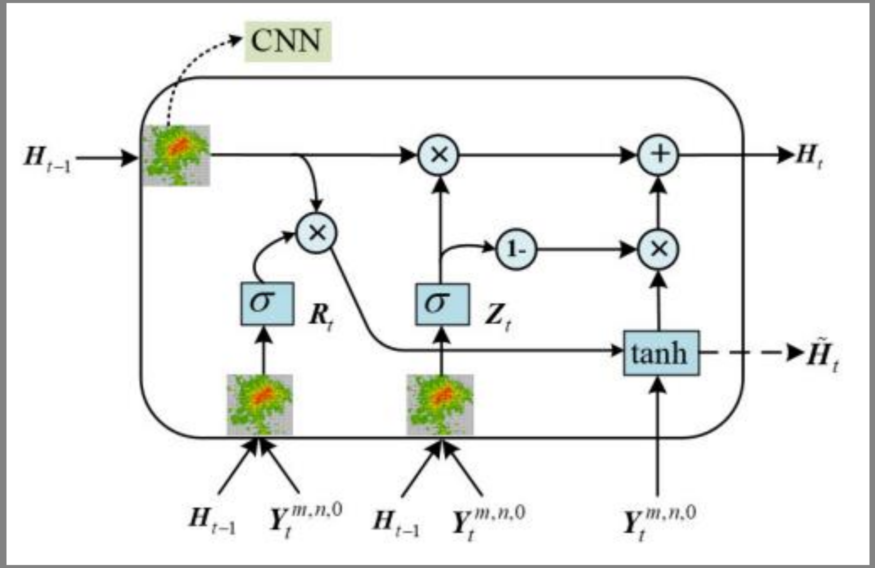
$$\boldsymbol{p_{ij} = \Pi_c(G_{ij} \circ \Pi_c^{-1}(p_i, d_i))}$$

correspondence field

„where would each pixel of image i with it's depth be located when projected into image j"

→ **used for the lookup!**

Gated Recurrent Unit



[8]

$\widetilde{H} = hidden\ state\ candidate, H = hidden\ state, Y = input$

[7]

DROID-SLAM [7] – DBA Layer

Dense **B**undle **A**djustment

$$E(G', d') = \sum_{(i,j)\epsilon\varepsilon} \left\| p_{ij}^{*} - \Pi_c \left(G'_{ij} \circ \Pi_c^{-1}(p_i, d'_i) \right) \right\|_{\Sigma_{ij}}^{2} \quad with\, \Sigma_{ij} = diag(w_{ij})$$

[7]

DROID-SLAM [7] – DBA Layer

Dense **B**undle **A**djustment

minimize the loss

$$E(G', d') = \sum_{(i,j)\epsilon\varepsilon} \left\| p_{ij}^{*} - \Pi_c \left(G'_{ij} \circ \Pi_c^{-1}(p_i, d'_i) \right) \right\|_{\Sigma_{ij}}^{2} \quad with\ \Sigma_{ij} = diag(w_{ij})$$

optimize G and d

all edges of the frame graph (front vs backend!)

corrected position prediction of each pixel

projection, function of depth and camera pose

squared error weighted by confidence of the ConvGRU

solved with local parametrization, linearzization and a set of mathematical tricks (special matrix structure, Schurs complement, …)

[7]

DROID-SLAM [G] – DBA Layer

Dense **B**undle **A**djustment

local parametrization and solving yields $\Delta\xi$ and Δd

G and d are then updated via retraction on the SE3 manifold

$$\mathbf{G}^{(\mathbf{k+1})} = \mathbf{Exp}\left(\boldsymbol{\Delta\xi}^{(\boldsymbol{k})}\right) \circ \boldsymbol{G}^{\boldsymbol{k}} \quad \textbf{and} \quad \mathbf{d}^{\mathbf{k+1}} = \mathbf{d}^{\mathbf{k}} + \boldsymbol{\Delta d}^{\boldsymbol{k}}$$

where G = poses, d = depths, k = current itation step, $\Delta\xi$ = pose change in tangent space

Special Euclidean 3 Group (SE3)
group of transformations that conists of rotations representable by a 3x3 rotation matrix and a transralation represented by a 3D vector
highly non-linear but we can go through the tangent space for small updates and transform them onto SE3 via Lie Algebra

NICE-SLAM [9] – Idea

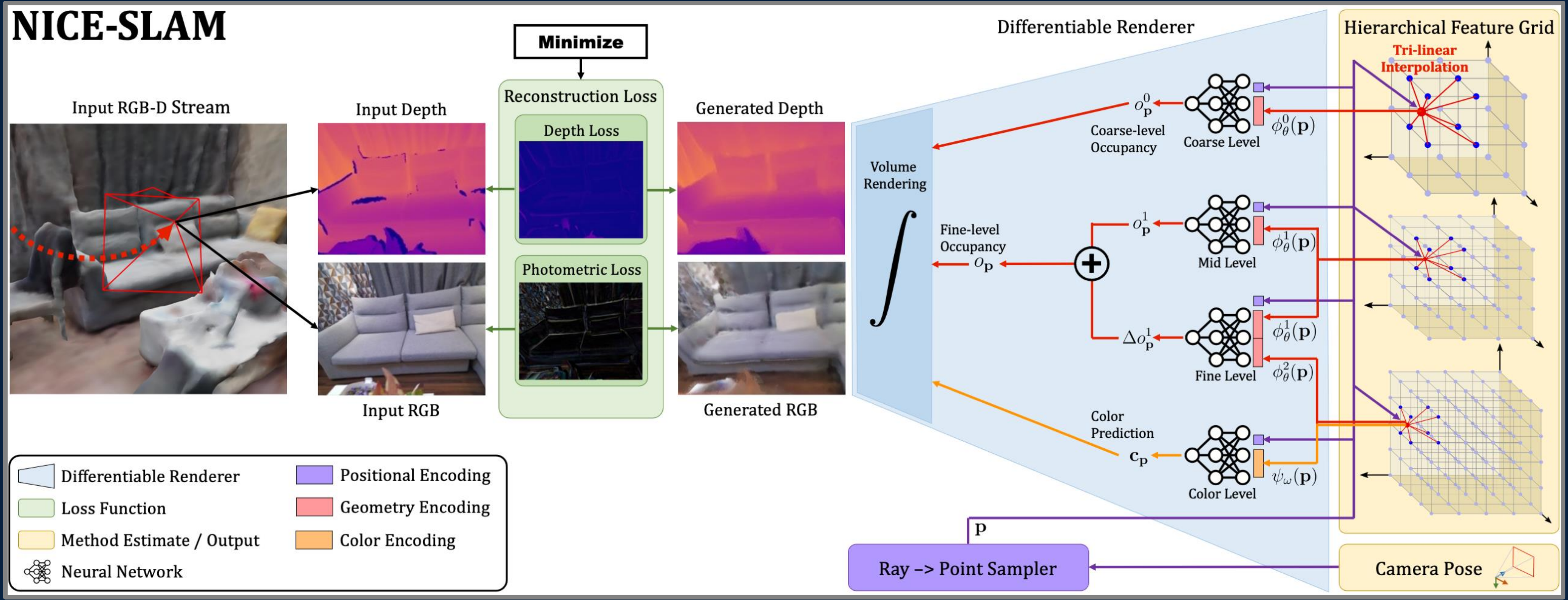
using differentiable rendering and the learning capabilities of neural networks for SLAM

[10]

- use neural implicit surface
- discretize space hierarchically
- distribute the responsibilities
- avoid forgetting learned areas

[9]

NICE-SLAM [9] – Pipeline



[10]

[9]

NICE-SLAM [9] – Pipeline

effectively just a storage

default

coarse geometry → $(2m)^3$

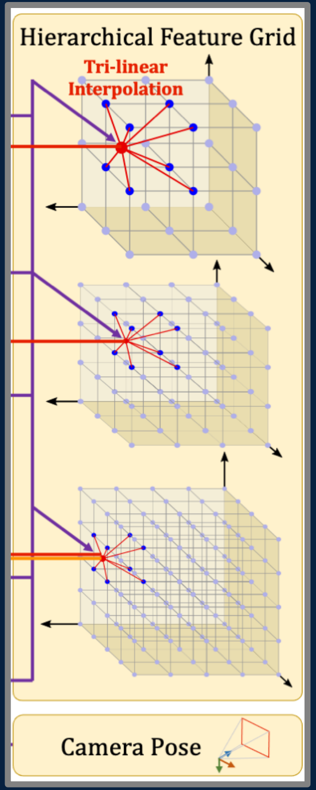
medium geometry → $(32cm)^3$

fine geometry → $(16cm)^3$

colors (second grid) →

position & direction →

voxel grid



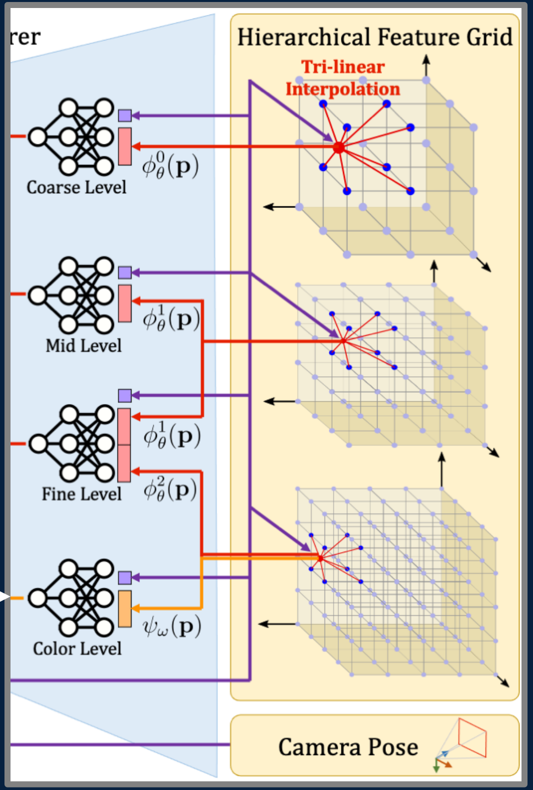
[10]

[9]

NICE-SLAM [9] – Pipeline

coarse and mid level pretrained MLP decoder networks on reconstruction

not pretrained, jointly optimized with color grid



[10]

[9]

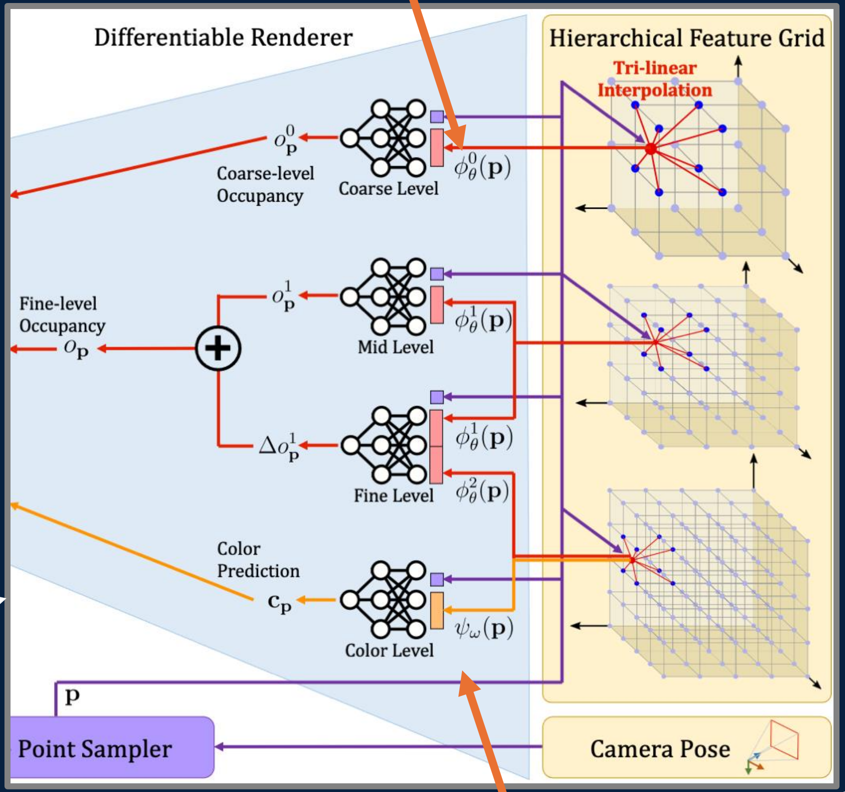
NICE-SLAM [9] – Pipeline

coarse and mid level pretrained MLP decoder networks on reconstruction

fine level network trained to complete/fix mid level

not pretrained, jointly optimized with color grid

neural network



neural network

[10]

[9]

NICE-SLAM [9] – Pipeline

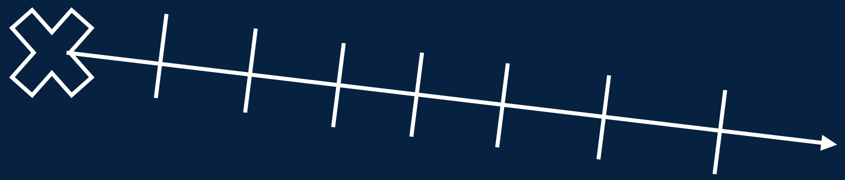
Volume Rendering
is something there?

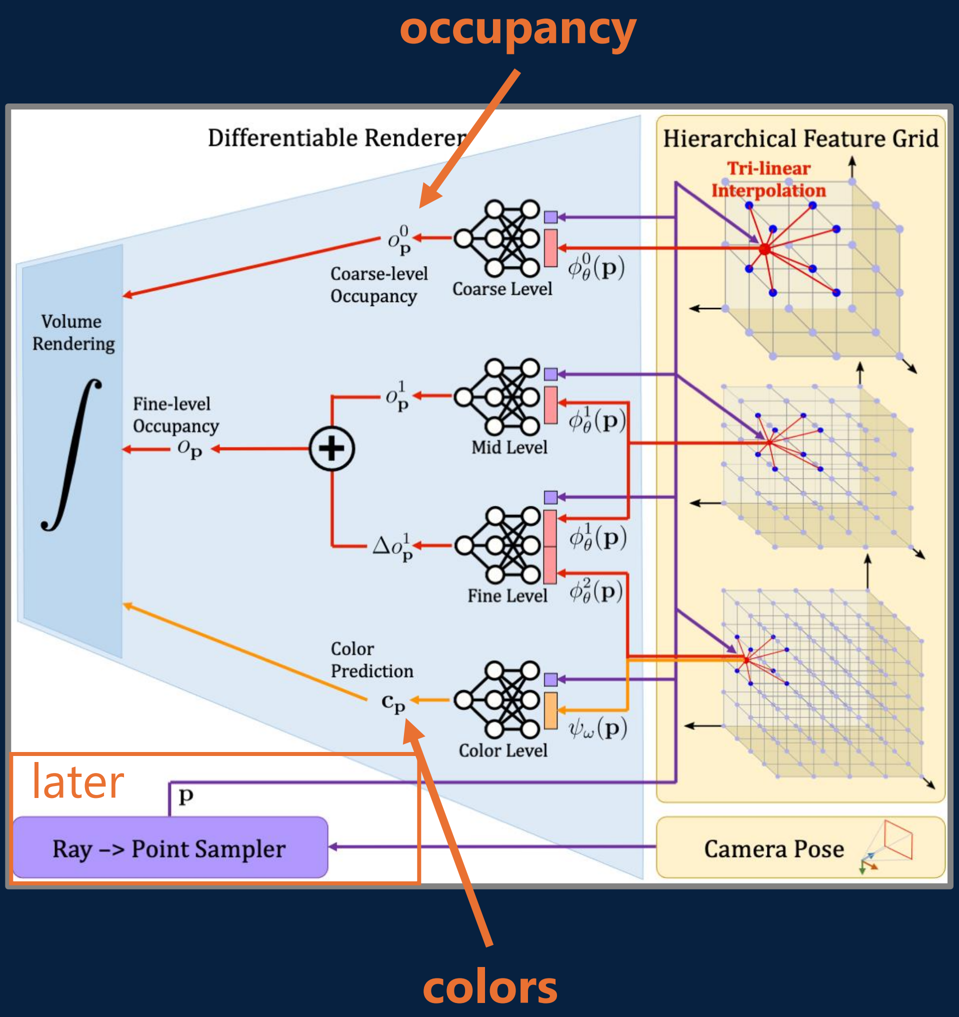
1. shoot a ray camera origin → pixel direction

$$p_i = o + d_i * r$$

p = point on line, o = camera origin, d = distance from origin, r = direction vector

2. sample at N distances





[10]

[9]

NICE-SLAM [9] – Pipeline

Volume Rendering
is something there?

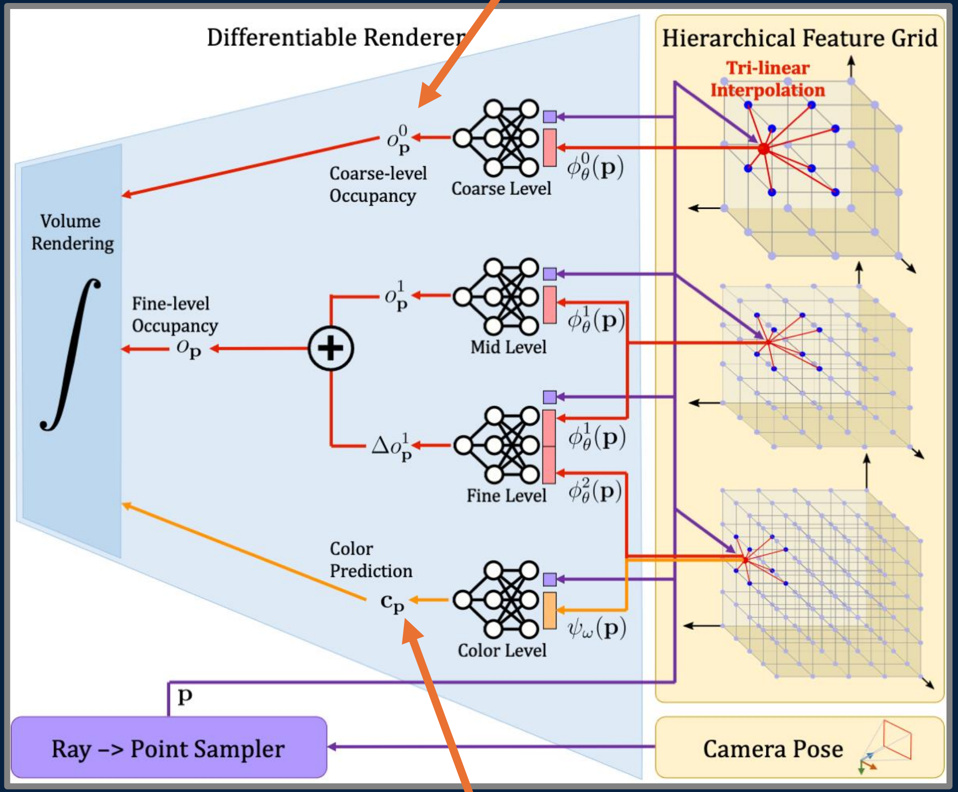
1. shoot a ray camera origin → pixel direction
   $p_i = c + d_i * r$
   p = point on line, c = camera origin, d = distance from origin, r = direction vector

2. sample at N distances

3. survival probability $w_i = o_{p_i} \prod_{j=1}^{i-1}(1 - o_{p_j})$

4. depth and color can be rendered as
   $\widehat{D} = \sum_{i=1}^{N} w_i * d_i$ (depth)
   $\hat{I} = \sum_{i=1}^{N} w_i^f * c_i$ (color)

occupancy



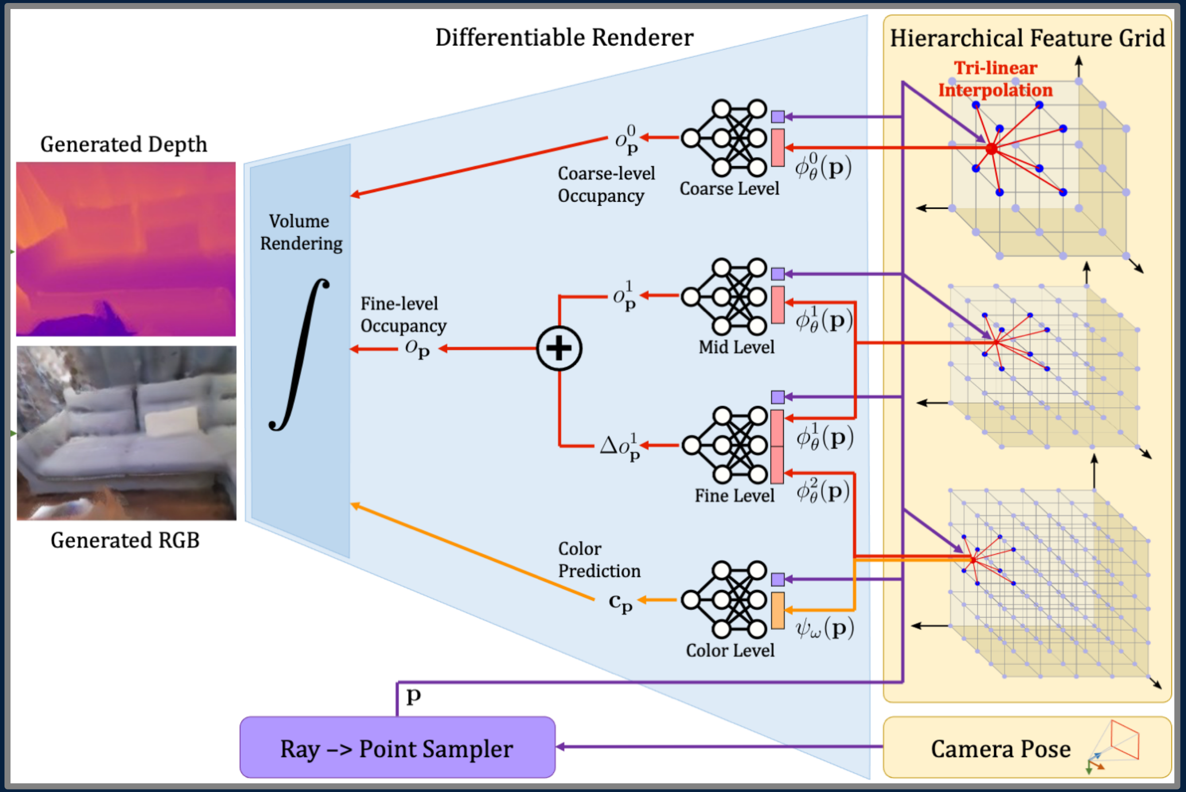
[10]

colors

[9]

NICE-SLAM [9] – Pipeline

Done (?)



[10]

[9]

NICE-SLAM [9] – Mapping

building a loss function (L1)

mapping loss function

coarse geometric loss

$$\mathcal{L}_g^c = \frac{1}{M}\sum_{m=1}^{M} |D_m - \widehat{D_m^c}|$$

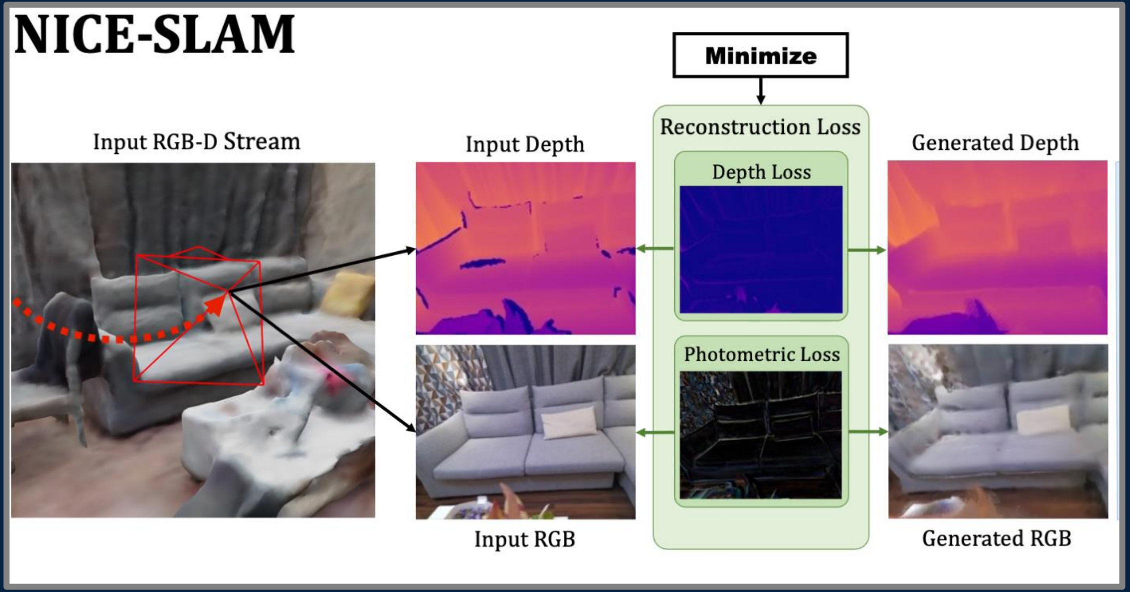
fine geometric loss

$$\mathcal{L}_g^f = \frac{1}{M}\sum_{m=1}^{M} |D_m - \widehat{D_m^f}|$$

photometric loss

$$\mathcal{L}_p = \frac{1}{M}\sum_{m=1}^{M} |I_m - \widehat{I_m}|$$

M = number of samples



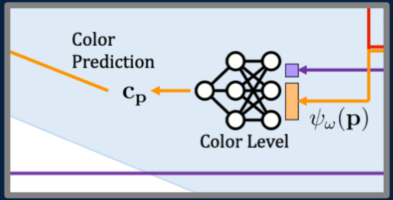
[10]

[9]

NICE-SLAM [9] – Mapping

optimize

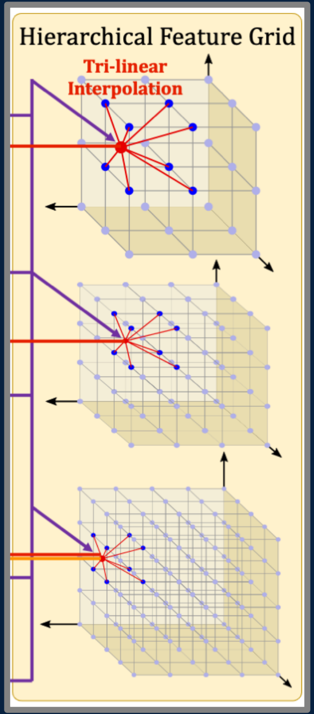
backpropagate loss to grid parameters and the learnable parameters of the color network



[10]

Training Schedule

1. optimize coarse and mid-level features on $\mathcal{L}_g^{c/f}$
2. optimize mid and fine-level features together on $\mathcal{L}_g^{f}$
3. perform local bundle adjustment (see earlier) to jointly optimize all grids, color decoder and camera extrinsic parameters on $\lambda_p\mathcal{L}_p + \mathcal{L}_g^f + \mathcal{L}_g^c$, λ_p = weighting factor



[10]

[9]

NICE-SLAM [9] – Tracking

building another loss function

Geometric Loss weighted by inverse variance

$$L_{g\,var} = \frac{1}{M_t}\sum_{m=1}^{M_t} \frac{|D_m - \widehat{D_m^c}|}{\sqrt{\widehat{D_{var}^c}}} + \frac{|D_m - \widehat{D_m^f}|}{\sqrt{\widehat{D_{var}^f}}}$$

M_t = number ob samples for tracking

+

Photometric Loss with weighting factor

$$\lambda_{pt}\mathcal{L}_p = \frac{\lambda_{pt}}{M}\sum_{m=1}^{M} |I_m - \breve{I_m}|$$

λ_{pt} = weighting factor for photometric loss

Depth Variance

$$\widehat{D_{var}} = \sum_{i=1}^{N} w_i * \left(\widehat{D} - d_i\right)^2$$

→ **backpropagate**

with regards to translation and rotation of the camera

[9]

NICE-SLAM [9] – task split

coarse grid – give some info on occupancy, some info on partially unseen areas

medium grid – focus on basic structure, provide general shape of environment

fine grid – focus on high level details, improve the medium grid

color grid – provide additional signals for tracking

[9]

Results – NICE-SLAM [9]

	fr1/desk	fr2/xyz	fr3/office
iMAP [47]	4.9	2.0	5.8
iMAP* [47]	7.2	2.1	9.0
DI-Fusion [16]	4.4	2.3	15.6
NICE-SLAM	2.7	1.8	3.0
BAD-SLAM [43]	1.7	1.1	1.7
Kintinuous [60]	3.7	2.9	3.0
ORB-SLAM2 [27]	**1.6**	**0.4**	**1.0**

TUM-RGB-D

[9] + table references at [9]

[10]

- evaluated on five datasets
- very good performance for techniques using neural implicit representation
- far from state of the art
- no failures reported
- appears very selective in the experiments (indoor)

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

Runtime Comparison

[9] + table references at [9]

[9]

Results – DROID-SLAM [7]

	360	desk	desk2	floor	plant	room	rpy	teddy	xyz	avg
ORB-SLAM2 [32]	X	0.071	X	0.023	X	X	X	X	0.010	-
ORB-SLAM3 [5]	X	**0.017**	0.210	X	0.034	X	X	X	**0.009**	-
DeepTAM[1] [60]	0.111	0.053	0.103	0.206	0.064	0.239	0.093	0.144	0.036	0.116
TartanVO[2] [54]	0.178	0.125	0.122	0.349	0.297	0.333	0.049	0.339	0.062	0.206
DeepV2D [48]	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 [9]	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**

TUM-RGB-D

[7] + table references at [7]

Tanks and Temples

[11]

8-30 fps

reported depending on camera speed

- evaluated on in- and outdoor datasets
- very robust, even on noisy inputs, bad lighting etc.
- outperformed state of the art, today still in top 3 of ETH3D-SLAM Benchmark [12]
- depending on application real-time capable

[7]

Personal Thoughts

DROID

runtime

ressources (electricity, memory)

great performance

determinism and explainability

dependency on training data

scaling

NICE

runtime

ressources

dependent on noisy RGB-D

determinism and explainability

quality

anything big scale problematic

Future Work

1. discard non-essential images after loop closure → memory efficiency
2. use depth as a 4th dimension when RGB-D images are available → leverage the information available
3. combine with a classic approach to provide explainability → practical usage
4. denoising diffusion nets as powerful prior to estimate (guess) the area not yet seen → increase chance of finding track again if lost
5. use non linear motion model for better tracking of highly dynamic objects

Key Take-Aways

- learning based differentiable SLAM can have outstanding performance
- issues with resource requirements
- neural network based feature extraction has a lot of potential for SLAM

Discussion

Thank you for your Attention

References

[1] https://www.synaos.com/integrations/mobile-robot-finder/abb-amr-t702 (01.12.2024)

[2] Tranzatto, Marco, et al. "Cerberus: Autonomous legged and aerial robotic exploration in the tunnel and urban circuits of the darpa subterranean challenge." arXiv preprint arXiv:2201.07067 (2022): 3

[3] https://commons.wikimedia.org/wiki/File:Stanley2.JPG (01.12.2024)

[4] Dewan, Abhishek, et al. "Advancement in SLAM Techniques and Their Diverse Applications." 2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART). IEEE, 2023.

[5] https://learnopencv.com/monocular-slam-in-python/ (01.12.2024)

[6] https://wiki.seeedstudio.com/a_loam/

[7] 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] Xu, Miao, Hongfei Liu, and Hongbo Yang. "A deep learning based multi-block hybrid model for bike-sharing supply-demand prediction." IEEE Access 8 (2020): 85826-85838.

[9] 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.

[10] https://pengsongyou.github.io/nice-slam (01.12.2024)

[11] https://www.youtube.com/watch?v=GG78CSlSHSA (01.12.2024)

[12] https://www.eth3d.net/slam_benchmark (01.12.2024)

[13] https://www.youtube.com/watch?v=3bkjse1keSA&ab_channel=RcLab

[14] 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.

[15] Whyte, H. Durrant. "Simultaneous localisation and mapping (SLAM): Part I the essential algorithms." *Robotics and Automation Magazine* (2006).