NYU TANDON SCHOOL OF ENGINEERING

AI4CE

Robot Perception

SLAM

Dr. Chen Feng

cfeng@nyu.edu

ROB-GY 6203, Fall 2023

Overview

++ Graph-based SLAM

+ SLAM formalism (state, observation, error/observation model)

* 1D SLAM example

+ A probabilistic perspective of SLAM

+ Parallel Tracking And Mapping (PTAM)

* Visual Place Recognition

+ Superpoint

*: know how to code (or how to use tools)
++: know how to derive (more than just the concept)
+: know the concept

References

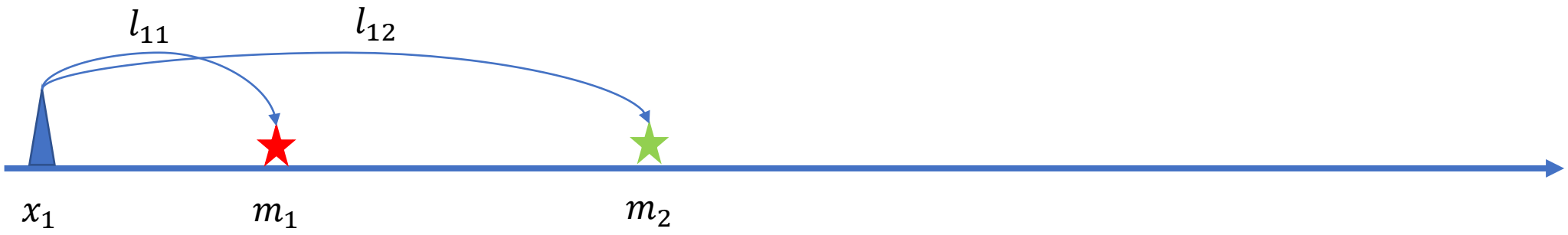
- Visual SLAM Tutorial, CVPR'14
- Dellaert, Frank. "Visual SLAM Tutorial: Bundle Adjustment." (2014).
- Grisetti, Giorgio, et al. "A tutorial on graph-based SLAM." IEEE Intelligent Transportation Systems Magazine 2.4 (2010): 31-43.
- Klein, G. and Murray, D., 2007, November. Parallel tracking and mapping for small AR workspaces. In 2007 6th IEEE and ACM international symposium on mixed and augmented reality (pp. 225-234). IEEE.
- Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R. and Ng, R., 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM, 65(1), pp.99-106.
- https://sites.google.com/view/lsvpr2019/home

Let's look at a 1D SLAM example

- A robot start at the location x_1

Let's look at a 1D SLAM example: observing

- A robot start at the location x_1, it observes
  - a landmark m_1 and find it is l_{11} away from the robot
  - a landmark m_2 and find it is l_{12} away from the robot

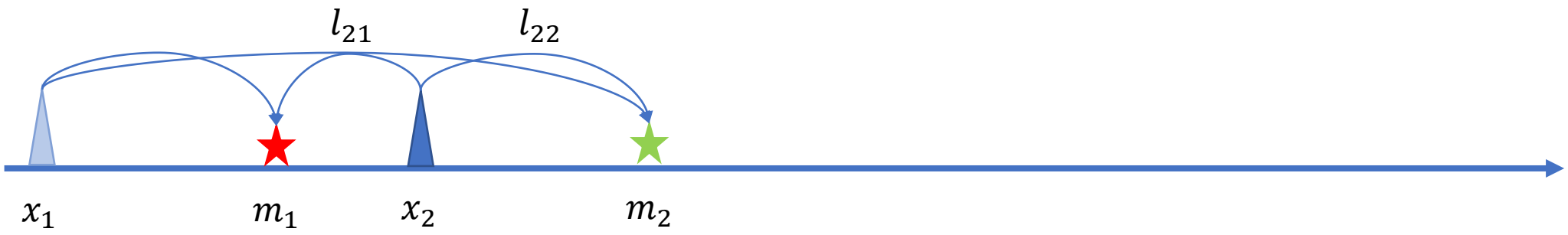


Let's look at a 1D SLAM example: moving

- A robot start at the location x_1
- Now the robot moves to another location x_2

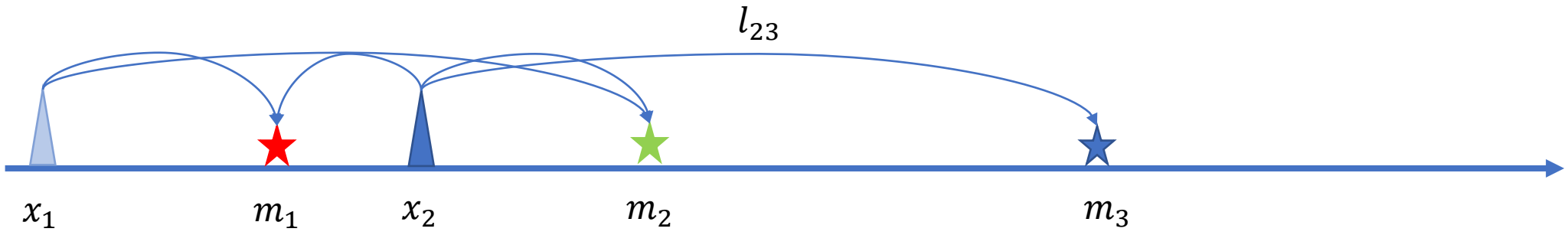
Let's look at a 1D SLAM example: re-observing

- Now the robot moves to another location x_2, it re-observes
  - The landmark m_1 and find it is l_{21} away from the robot
  - The landmark m_2 and find it is l_{22} away from the robot



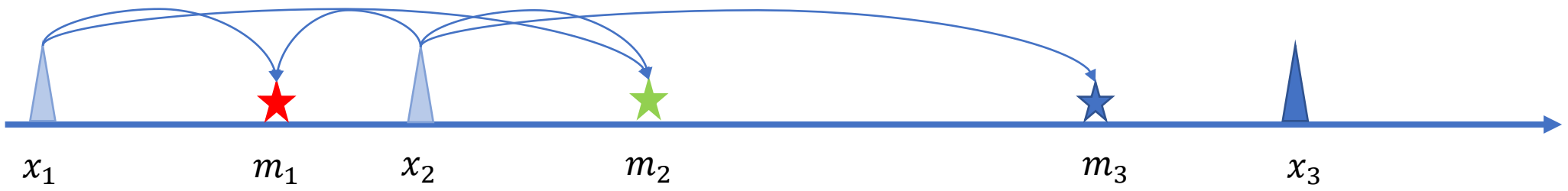
Let's look at a 1D SLAM example: new observations

- Now the robot moves to another location x_2, it discovers a new observation
  - A landmark m_3 and find it is l_{23} away from the robot



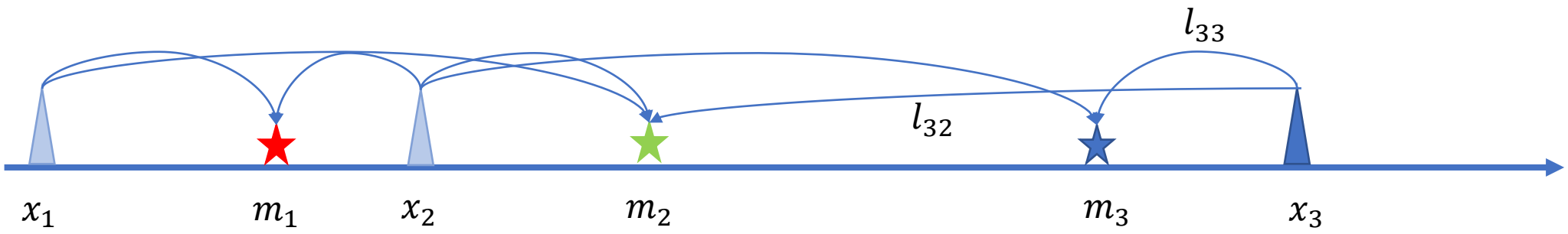
Let's look at a 1D SLAM example: moving again

- Now the robot moves to a new location x_3



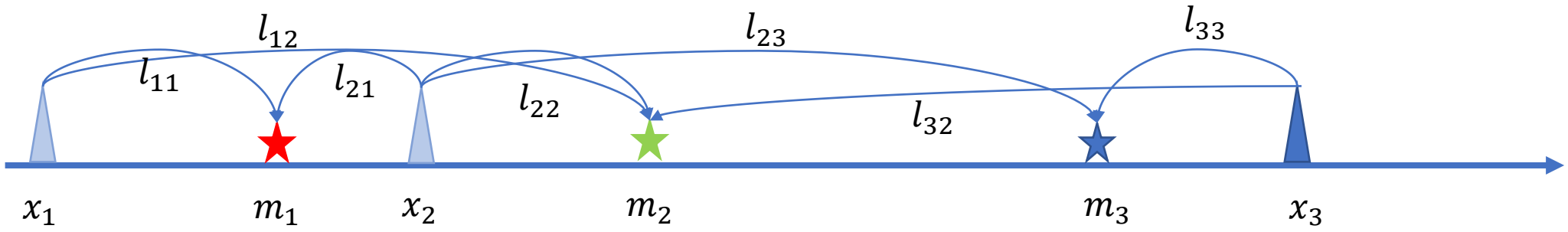
Let's look at a 1D SLAM example: re-observing

- Now the robot moves to a new location x_3, and it re-observes
  - The landmark m_2 and find it is l_{32} away from the robot
  - The landmark m_3 and find it is l_{33} away from the robot



Let's look at a 1D SLAM example

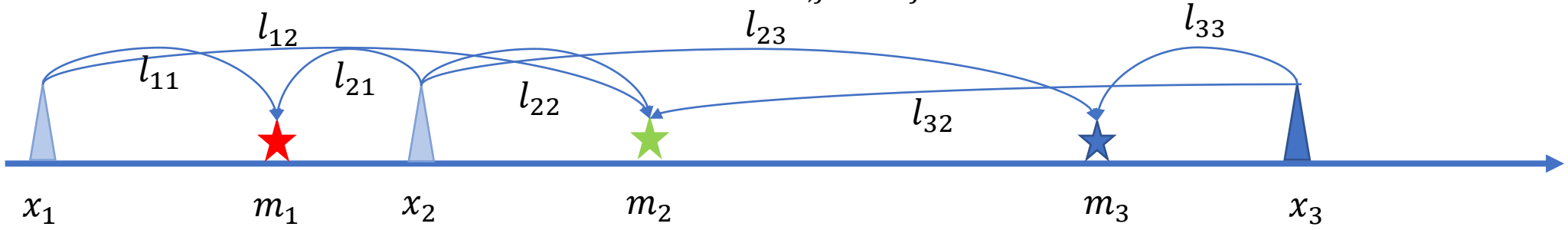
- We can formulate this as a SLAM problem by defining the following terms
  - States
  - Observations
  - Observation/Prediction/Measurement function



  - This would allow us to estimate states based on
    - the observations
    - the measurement function

Let's look at a 1D SLAM example

- We can formulate this as a SLAM problem by defining the following terms
  - States: $[x_1, x_2, x_3, m_1, m_2, m_3]$, Observations: $l_{i,j}$
  - Observation/Prediction/Measurement function: $l_{i,j} = m_j - x_i, \ \forall i, j$

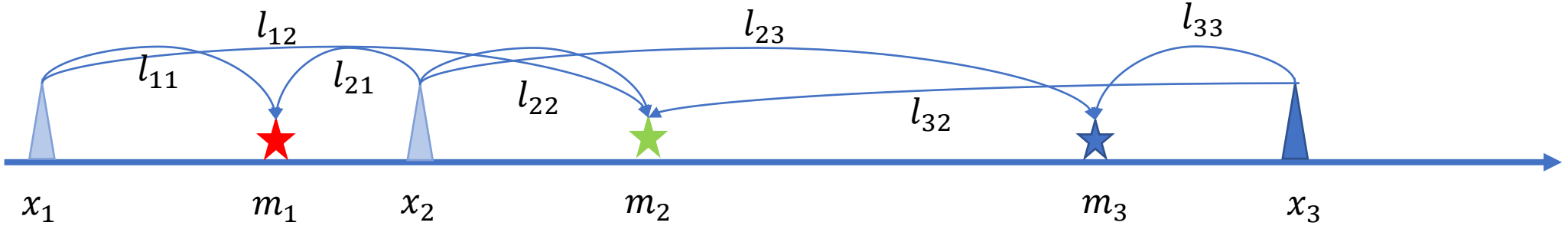


- $$\begin{bmatrix} l_{11} \\ l_{12} \\ l_{21} \\ l_{22} \\ l_{23} \\ l_{32} \\ l_{33} \end{bmatrix} = \begin{bmatrix} -1 & & & 1 & & \\ -1 & & & & 1 & \\ & -1 & & 1 & & \\ & -1 & & & 1 & \\ & -1 & & & & 1 \\ & & -1 & & 1 & \\ & & -1 & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ m_1 \\ m_2 \\ m_3 \end{bmatrix}$$
  $$x_1 \quad x_2 \quad x_3 \quad m_1 \quad m_2 \quad m_3$$

- This is an $\boldsymbol{Ax} = \boldsymbol{b}$ problem! $\boldsymbol{A}$ is the Jacobian of the measurement function

Let's look at a 1D SLAM example

- We can formulate this as a SLAM problem by defining the following terms
  - States: $[x_1, x_2, x_3, m_1, m_2, m_3]$, Observations: $l_{i,j}$
  - Observation/Prediction/Measurement function: $l_{i,j} = m_j - x_i, \ \forall i, j$

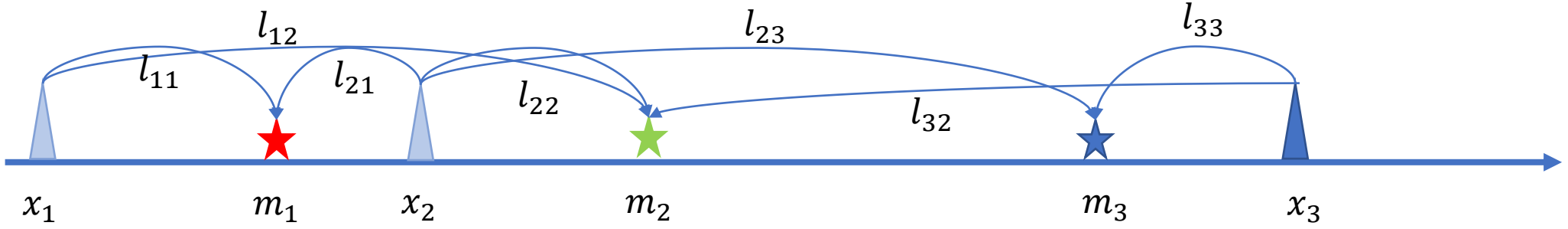


- We can solve it by $\boldsymbol{A'Ax} = \boldsymbol{A'b}$, and $\boldsymbol{A'A} = \begin{bmatrix} 2 & & & -1 & -1 & \\ & 3 & & -1 & -1 & -1 \\ & & 2 & & -1 & -1 \\ -1 & -1 & & 2 & & \\ -1 & -1 & -1 & & 3 & \\ & -1 & -1 & & & 2 \end{bmatrix}$

- But wait, this normal matrix $\boldsymbol{A'A}$ is singular, why?

Let's look at a 1D SLAM example

- We can formulate this as a SLAM problem by defining the following terms
  - States: $[x_1, x_2, x_3, m_1, m_2, m_3]$, Observations: $l_{i,j}$
  - Observation/Prediction/Measurement function: $l_{i,j} = m_j - x_i, \ \forall i, j$

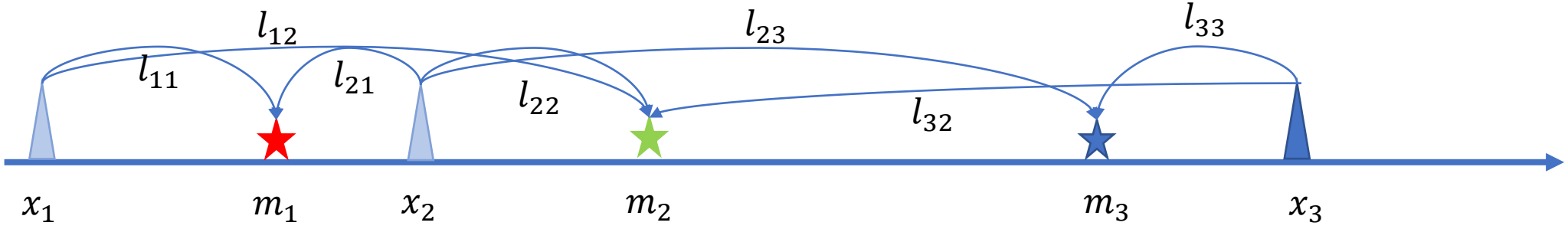


- $$\begin{bmatrix} l_{11} \\ l_{12} \\ l_{21} \\ l_{22} \\ l_{23} \\ l_{32} \\ l_{33} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} -1 & & & 1 & & \\ -1 & & & & 1 & \\ & -1 & & 1 & & \\ & -1 & & & 1 & \\ & -1 & & & & 1 \\ & & -1 & & 1 & \\ & & -1 & & & 1 \\ \mathbf{1} & & & & & \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ m_1 \\ m_2 \\ m_3 \end{bmatrix}$$, We must fix the initial condition!

$x_1 \quad x_2 \quad x_3 \quad m_1 \quad m_2 \quad m_3$

Let's look at a 1D SLAM example

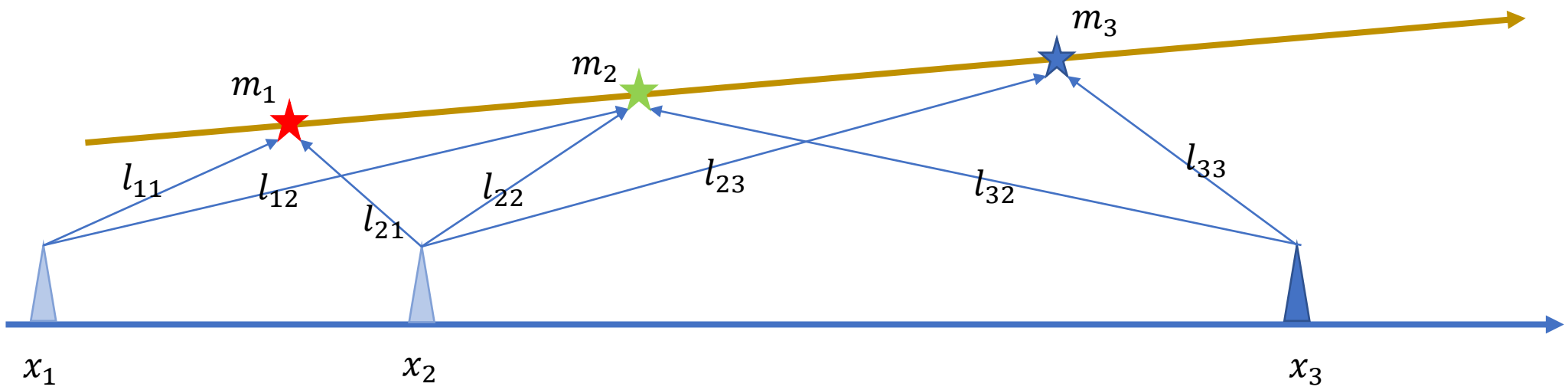
- We can formulate this as a SLAM problem by defining the following terms
  - States: $[x_1, x_2, x_3, m_1, m_2, m_3]$, Observations: $l_{i,j}$
  - Observation/Prediction/Measurement function: $l_{i,j} = m_j - x_i, \ \forall i, j$



- Now we can really solve it by $\boldsymbol{A'Ax} = \boldsymbol{A'b}$, and $\boldsymbol{A'A} = \begin{bmatrix} \mathbf{3} & & & -1 & -1 & \\ & 3 & & -1 & -1 & -1 \\ & & 2 & & -1 & -1 \\ -1 & -1 & & 2 & & \\ -1 & -1 & -1 & & 3 & \\ & -1 & -1 & & & 2 \end{bmatrix}$
  - A'A is also known as the Hessian matrix of the measurement function

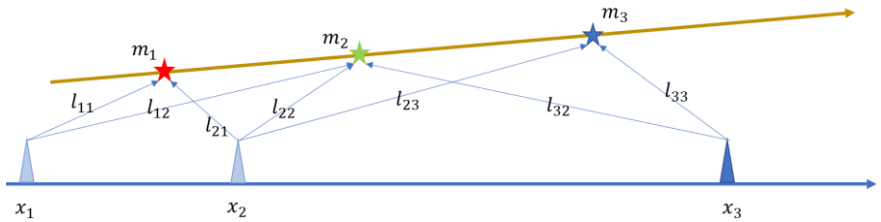
Another Example

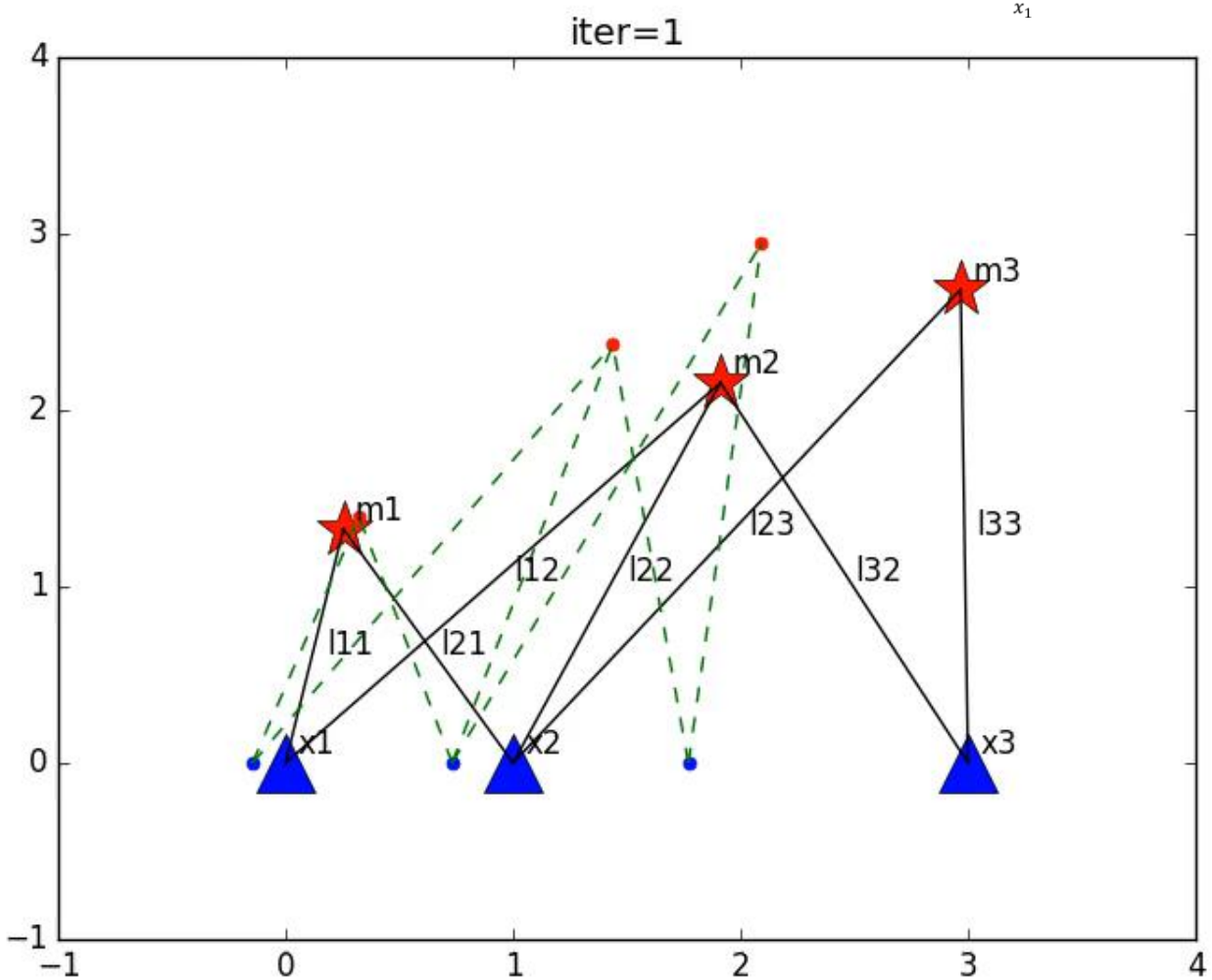
- Can you formulate this as a SLAM problem by the language we just discussed?
  - States? Observations $l_{i,j}$
  - Observation/Prediction/Measurement function?



  - Linearization of measurement function – need to be solved iteratively (more on this later)
  - Parameterization of States
    - Minimal parameterization
    - Over parameterization (Unit quaternions for 3D rotations) leads to a constrained optimization

Animation of SLAM Optimization





https://colab.research.google.com/drive/13fSbe9b2hlE7h63UToAMvEPTCvXYa7Cz?usp=sharing

SLAM as Estimation

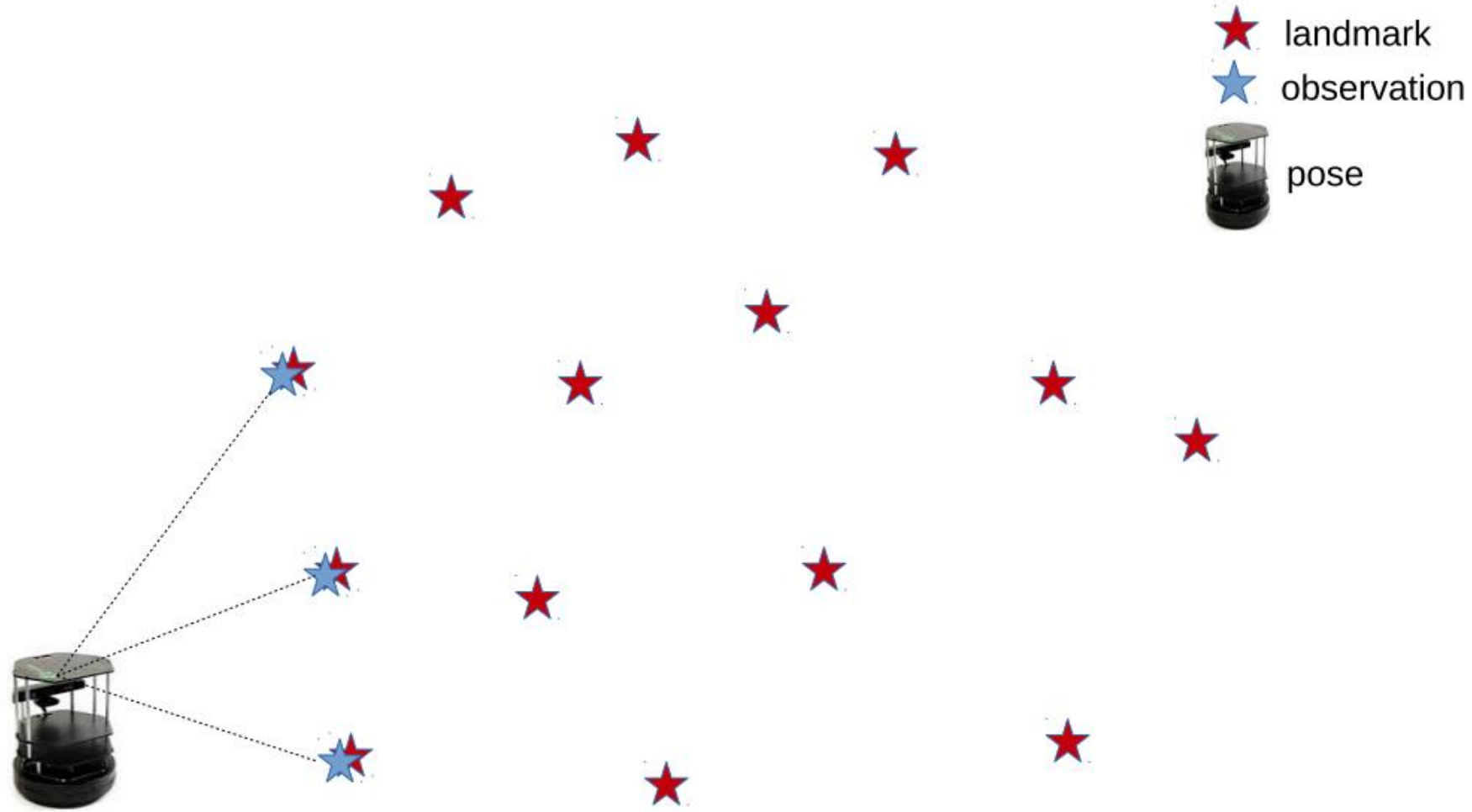


Image from Giorgio Grisetti 2016

SLAM as Estimation

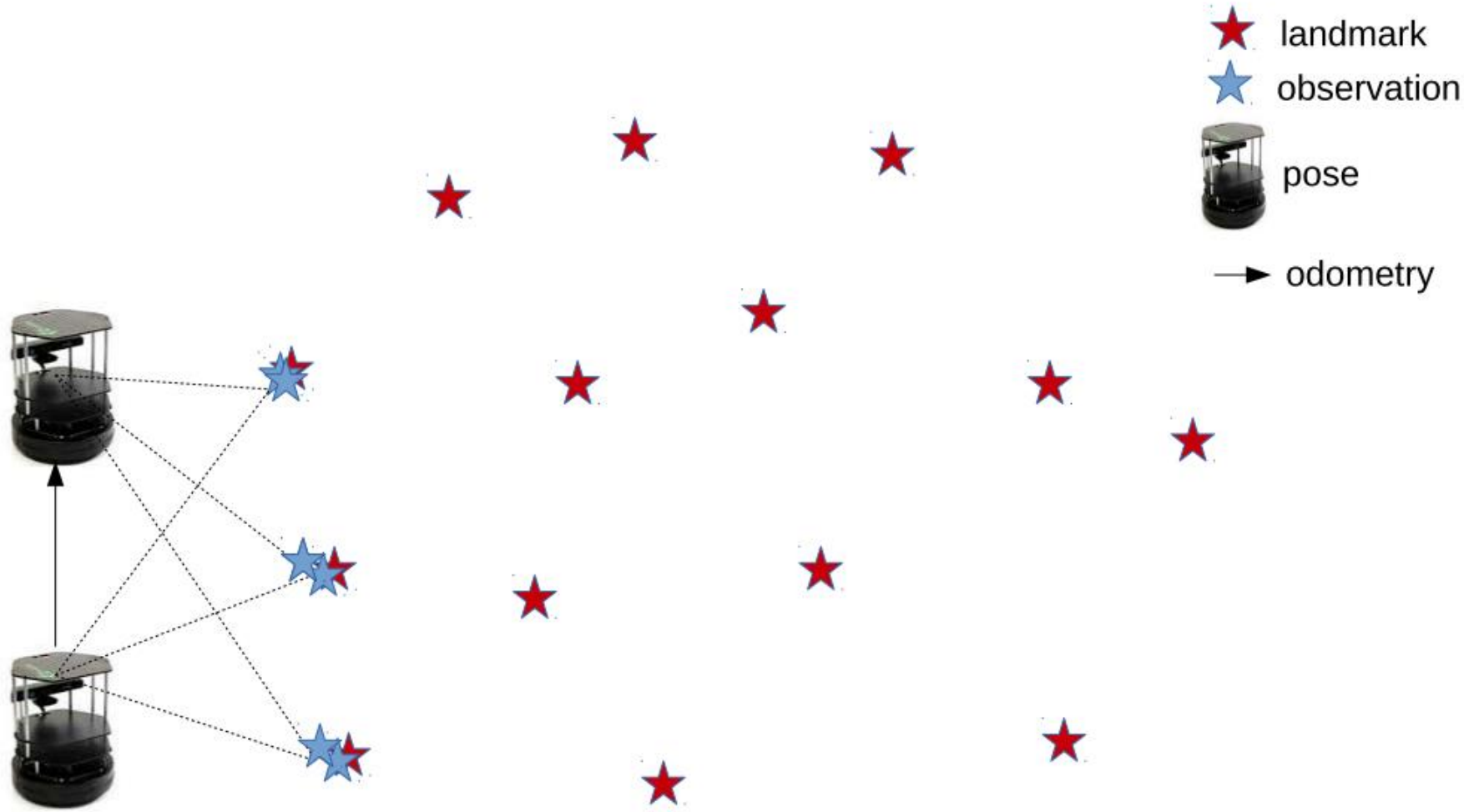


Image from Giorgio Grisetti 2016

SLAM as Estimation

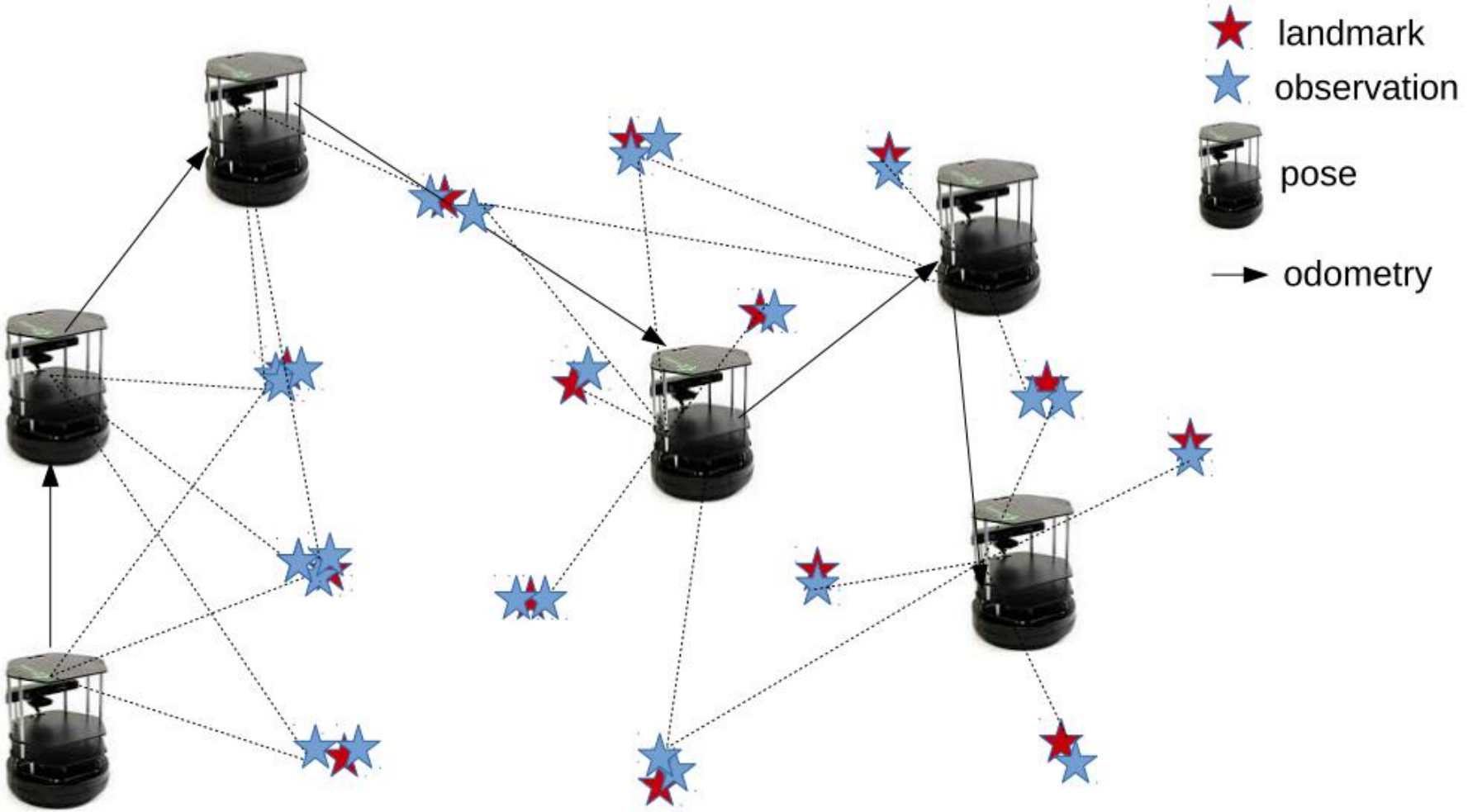


Image from Giorgio Grisetti 2016

SLAM as Estimation

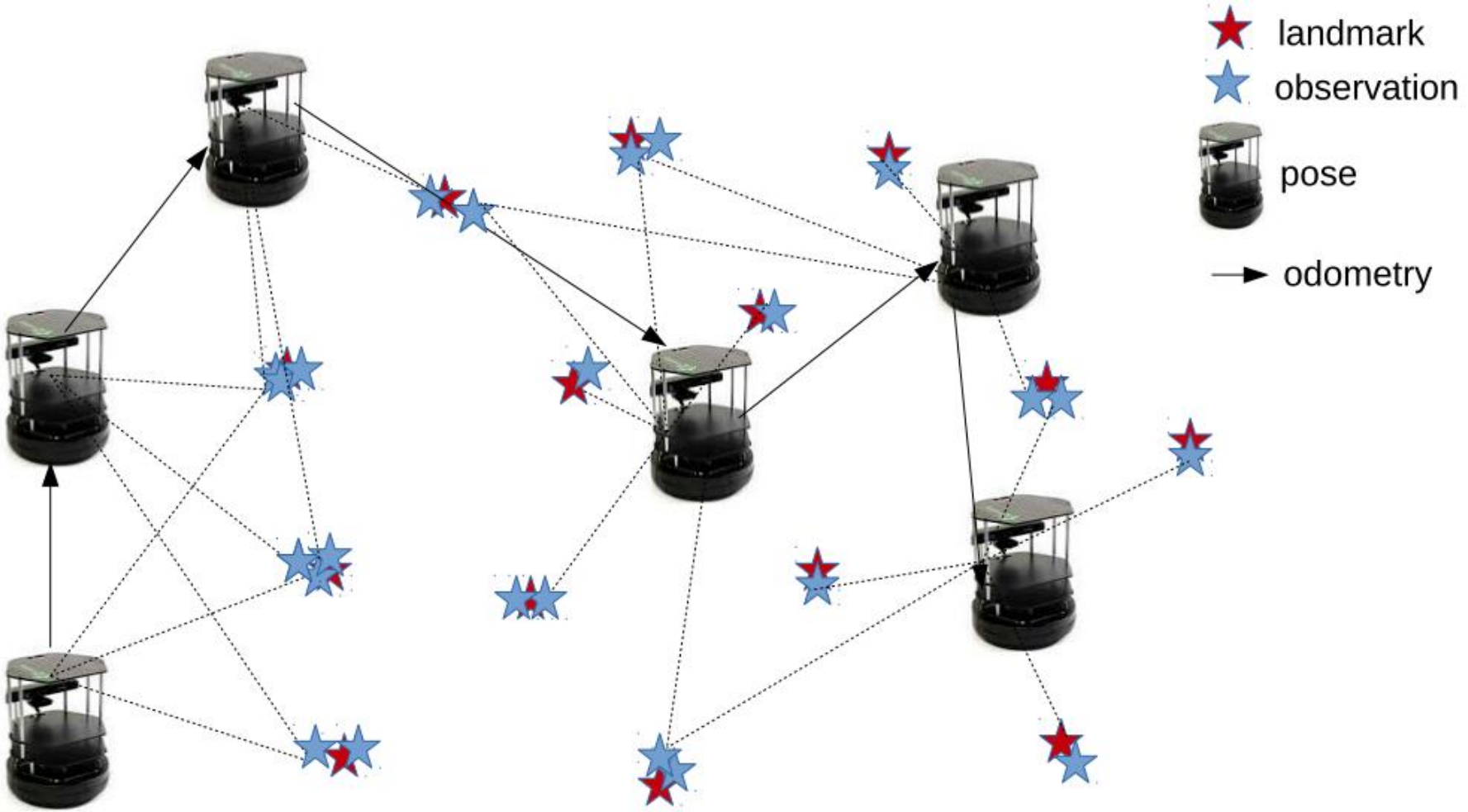


Image from Giorgio Grisetti 2016

SLAM as Estimation

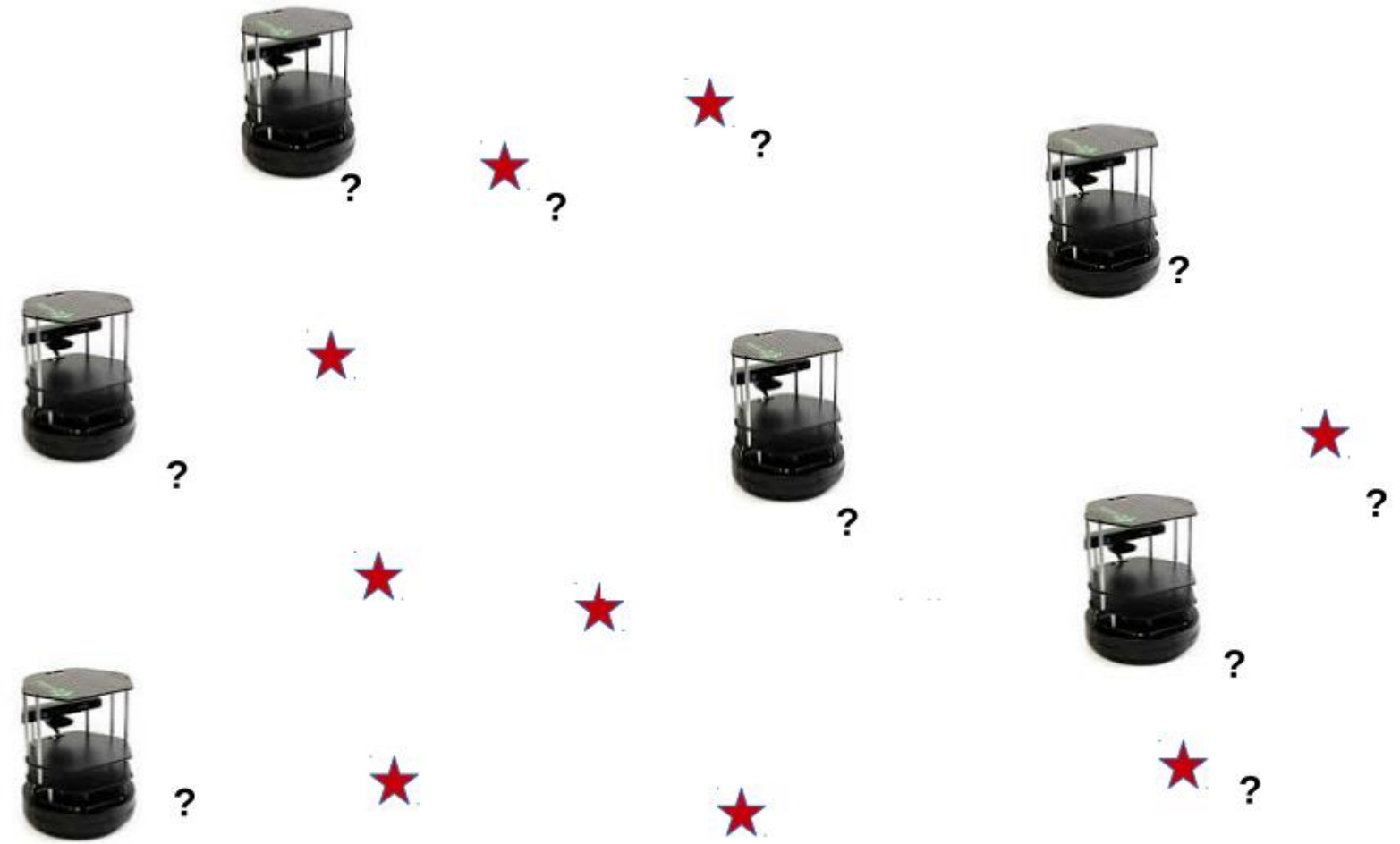


Image from Giorgio Grisetti 2016

SLAM from a Probabilistic Perspective

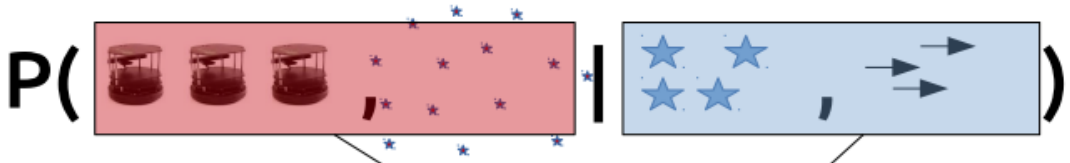
Estimate

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{z})$$

Slide from Giorgio Grisetti 2016

SLAM as Maximum Likelihood Estimation (MLE)

Estimate



$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{z})$$

Measurements

State

x*: state most consistent with observations

Slide from Giorgio Grisetti 2016

SLAM as Maximum Likelihood Estimation (MLE)

Using

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{z})$$

- Bayes' Rule

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})}$$

$$\propto p(\mathbf{z}|\mathbf{x})$$

← Likelihood function

- Independence,

$$= \prod_i p(\mathbf{z}_i|\mathbf{x})$$

We can further simplify the task

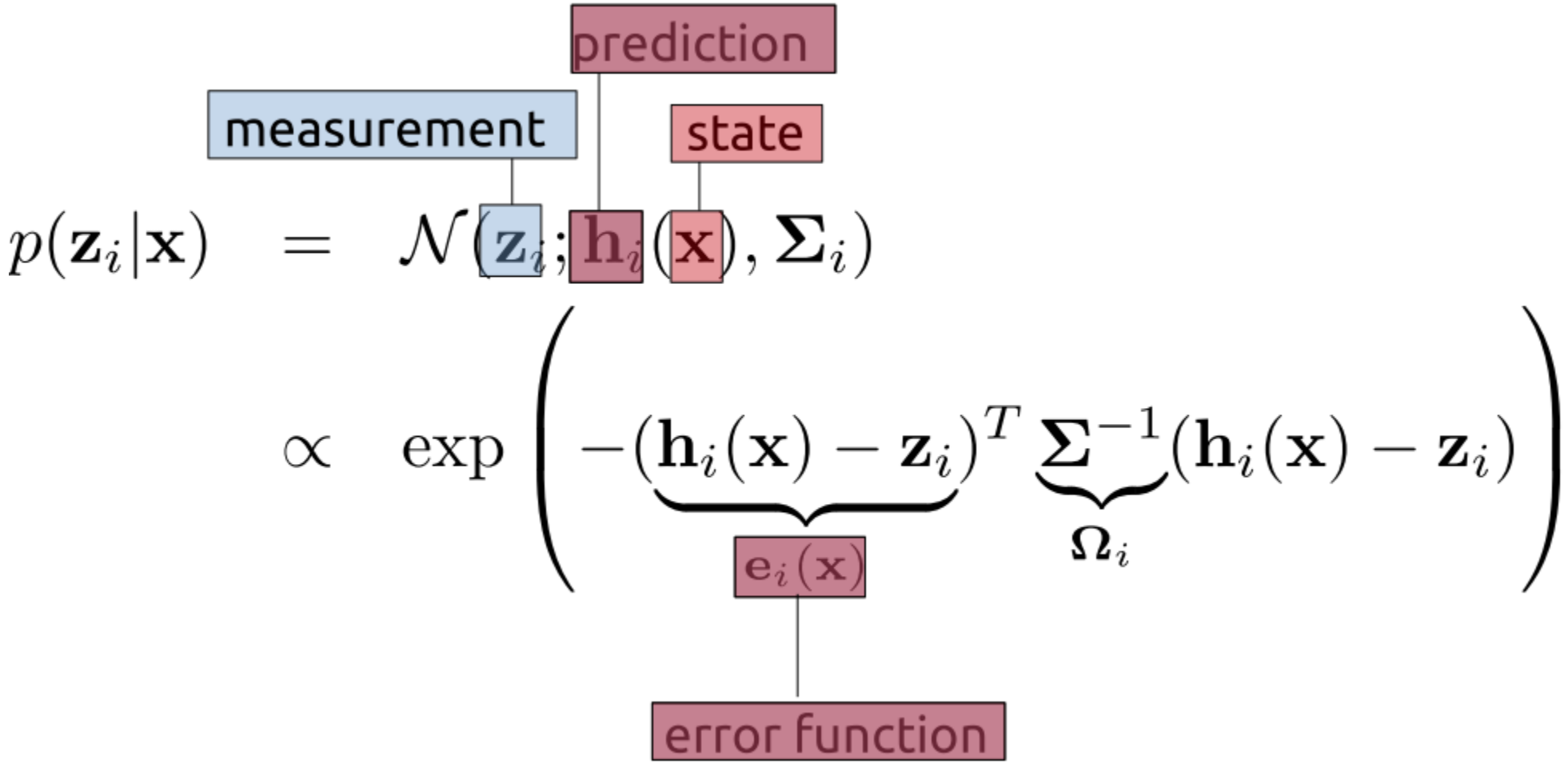
Slide from Giorgio Grisetti 2016

Gaussian Noise Assumption

$$
\begin{aligned}
p(\mathbf{z}_i|\mathbf{x}) &= \mathcal{N}(\mathbf{z}_i; \mathbf{h}_i(\mathbf{x}), \mathbf{\Sigma}_i) \\
&\propto \exp\left(-(\underbrace{\mathbf{h}_i(\mathbf{x}) - \mathbf{z}_i}_{\mathbf{e}_i(\mathbf{x})})^T \underbrace{\mathbf{\Sigma}^{-1}}_{\mathbf{\Omega}_i} (\mathbf{h}_i(\mathbf{x}) - \mathbf{z}_i) \right)
\end{aligned}
$$

Slide from Giorgio Grisetti 2016

Gaussian Noise Assumption



Slide from Giorgio Grisetti 2016

MLE == Least Squares Estimation

Through Gaussian assumption

- Maximization becomes minimization
- Product turns into sum

$$
\begin{aligned}
\mathbf{x}^* &= \operatorname*{argmax}_x \prod_i p(\mathbf{z}_i|\mathbf{x}) \\
&= \operatorname*{argmax}_x \prod_i \exp(-\mathbf{e}_i(\mathbf{x})^T \mathbf{\Omega}_i \mathbf{e}_i(\mathbf{x})) \\
&= \operatorname*{argmin}_x \sum_i \mathbf{e}_i(\mathbf{x})^T \mathbf{\Omega}_i \mathbf{e}_i(\mathbf{x})
\end{aligned}
$$

Slide from Giorgio Grisetti 2016

SLAM as Least Squares

Iterative minimization of

$$F(\mathbf{x}) = \sum_i \mathbf{e}_i(\mathbf{x})^T \mathbf{\Omega}_i \mathbf{e}_i(\mathbf{x})$$

Each iteration refines the current estimate by applying a perturbation

$$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{\Delta x}$$

Perturbation obtained by minimizing a quadratic approximation of the problem in $\mathbf{\Delta x}$

$$F(\mathbf{x} + \mathbf{\Delta x}) \simeq \mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x} + 2\mathbf{b}^T \mathbf{\Delta x} + c$$

Hessian Jacobian

Slide from Giorgio Grisetti 2016

Linearization

The quadratic approximation is obtained by linearizing the error functions around $\mathbf{x}$

$$\mathbf{e}_i(\mathbf{x}^* + \Delta\mathbf{x}) \simeq \underbrace{\mathbf{e}_i(\mathbf{x}^*)}_{\mathbf{e}} + \underbrace{\frac{\partial \mathbf{e}_i(\mathbf{x})}{\partial(\mathbf{x})}\Big|_{\mathbf{x}=\mathbf{x}^*}}_{\mathbf{J}_i} \Delta\mathbf{x}$$

...expanding the products

$$\mathbf{e}_i(\mathbf{x}^* + \Delta\mathbf{x})^T \Omega_i \mathbf{e}_i(\mathbf{x}^* + \Delta\mathbf{x}) \simeq$$

$$\Delta\mathbf{x}^T \underbrace{\mathbf{J}_i^T \Omega_i \mathbf{J}_i}_{\mathbf{H}_i} \Delta\mathbf{x} + 2\underbrace{\mathbf{J}_i^T \Omega_i \mathbf{e}_i}_{\mathbf{b}_i^T} \Delta_x + \underbrace{\mathbf{e}_i^T \Omega_i \mathbf{e}_i}_{c_i}$$

...and grouping the terms

$$\mathbf{H} = \sum_i \mathbf{H}_i \qquad \mathbf{b} = \sum_i \mathbf{b}_i \qquad c = \sum_i c_i$$

Slide from Giorgio Grisetti 2016

Quadratic Form

Find the $\Delta \mathbf{x}$ that minimizes the quadratic approximation of the objective function

$$\begin{aligned} \boldsymbol{\Delta}\mathbf{x}^* &= \underset{\boldsymbol{\Delta}\mathbf{x}}{\operatorname{argmin}} F(\mathbf{x}^* + \boldsymbol{\Delta}\mathbf{x}) \\ &\simeq \underset{\boldsymbol{\Delta}\mathbf{x}}{\operatorname{argmin}} \boldsymbol{\Delta}\mathbf{x}^T \mathbf{H} \boldsymbol{\Delta}\mathbf{x} + 2\mathbf{b}^T \boldsymbol{\Delta}\mathbf{x} + c \end{aligned}$$

Find $\Delta \mathbf{x}$ that nulls the derivative of quadratic form

$$\begin{aligned} \mathbf{0} &= \frac{\partial \left[\boldsymbol{\Delta}\mathbf{x}^T \mathbf{H} \boldsymbol{\Delta}\mathbf{x} + 2\mathbf{b}^T \boldsymbol{\Delta}\mathbf{x} + c \right]}{\partial \boldsymbol{\Delta}\mathbf{x}} \\ -\mathbf{b} &= \mathbf{H}\boldsymbol{\Delta}\mathbf{x} \end{aligned}$$

Slide from Giorgio Grisetti 2016

One Iteration in the Least Square Optimization

Clear H and b

$$\mathbf{H} \leftarrow 0 \qquad \mathbf{b} \leftarrow 0$$

For each measurement, update h and b

$$\begin{aligned}
\mathbf{e}_i &\leftarrow \mathbf{h}_i(\mathbf{x}^*) - \mathbf{z}_i \\
\mathbf{J}_i &\leftarrow \left.\frac{\partial \mathbf{e}_i(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} \\
\mathbf{H} &\leftarrow \mathbf{H} + \mathbf{J}_i^T \mathbf{\Omega}_i \mathbf{J}_i \\
\mathbf{b} &\leftarrow \mathbf{b} + \mathbf{J}_i^T \mathbf{\Omega}_i \mathbf{e}_i
\end{aligned}$$

Update the estimate with the perturbation

$$\begin{aligned}
\mathbf{\Delta x} &\leftarrow \text{solve}(\mathbf{H}\mathbf{\Delta x} = -\mathbf{b}) \\
\mathbf{x}^* &\leftarrow \mathbf{x}^* + \mathbf{\Delta x}
\end{aligned}$$

Slide from Giorgio Grisetti 2016

Recap the SLAM process

Identify the state space X

- Qualify the domain
- Find a locally Euclidean parameterization

Identify the measurement space(s) Z

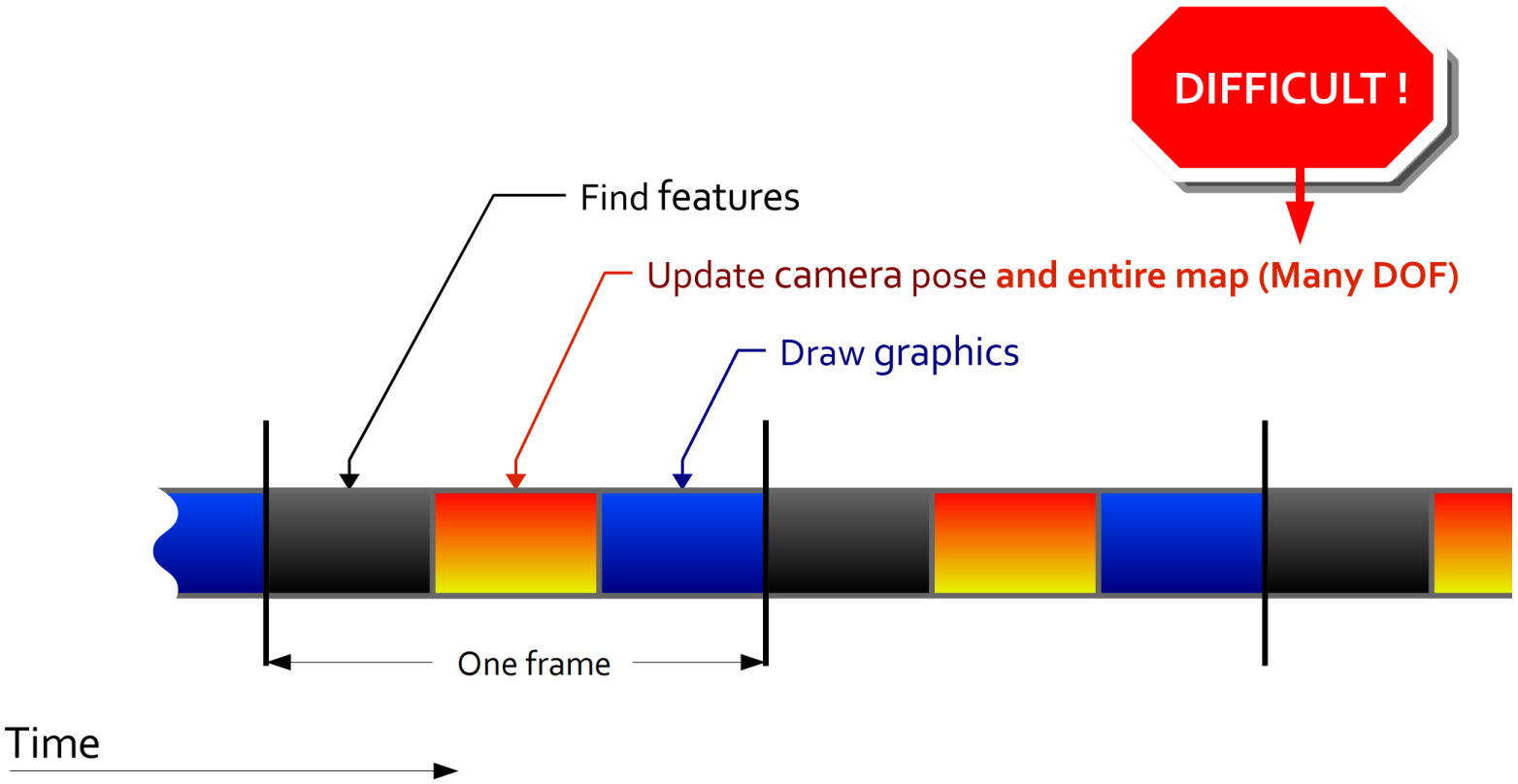
- Qualify the domain
- Find a locally Euclidean parameterization

Identify the prediction functions h(x)

Slide from Giorgio Grisetti 2016

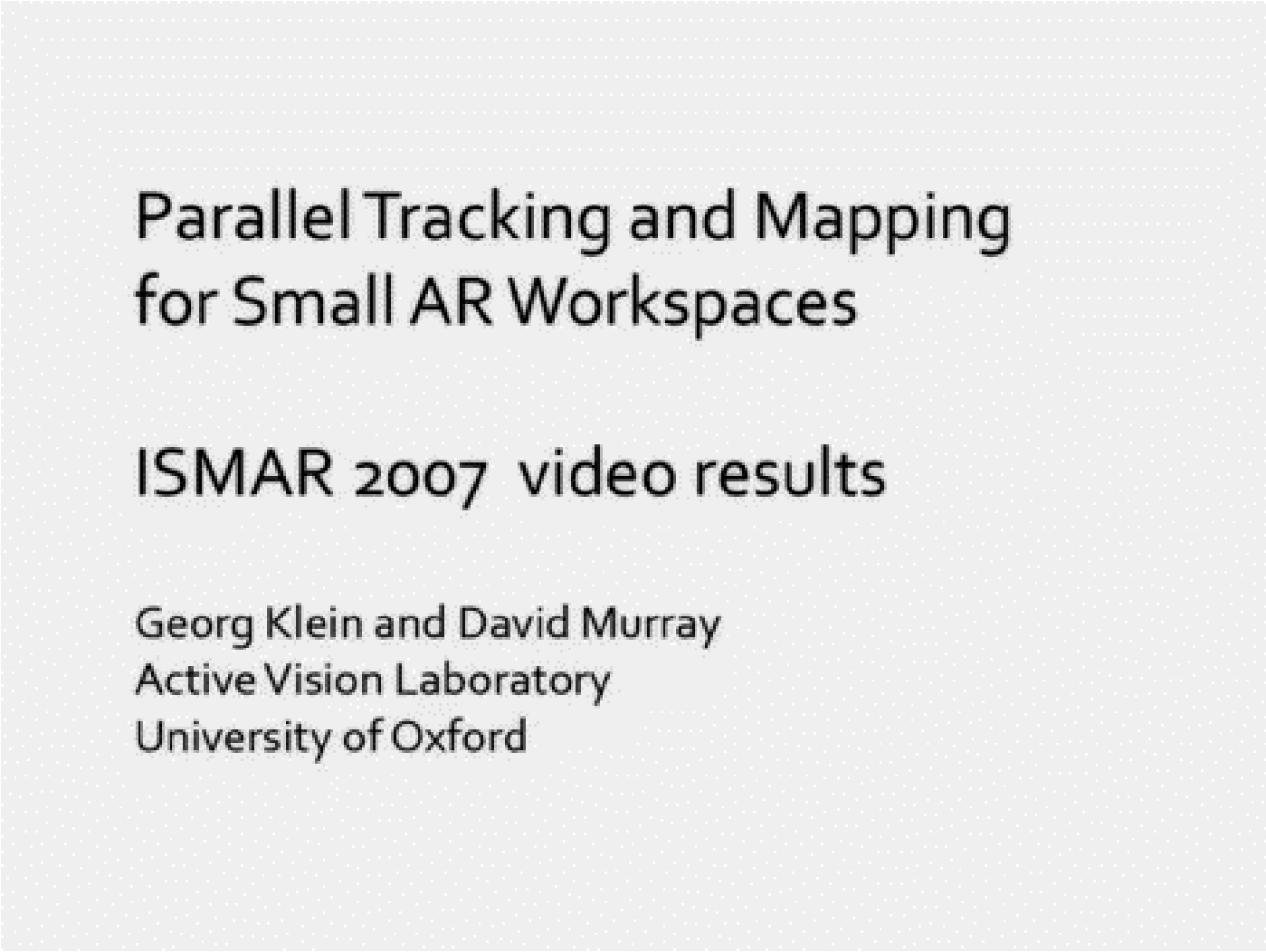
Frame-by-frame SLAM

- Updating entire map every frame is expensive
- Mandates “sparse map of high-quality features”



Slide from Klein & Murray 2007

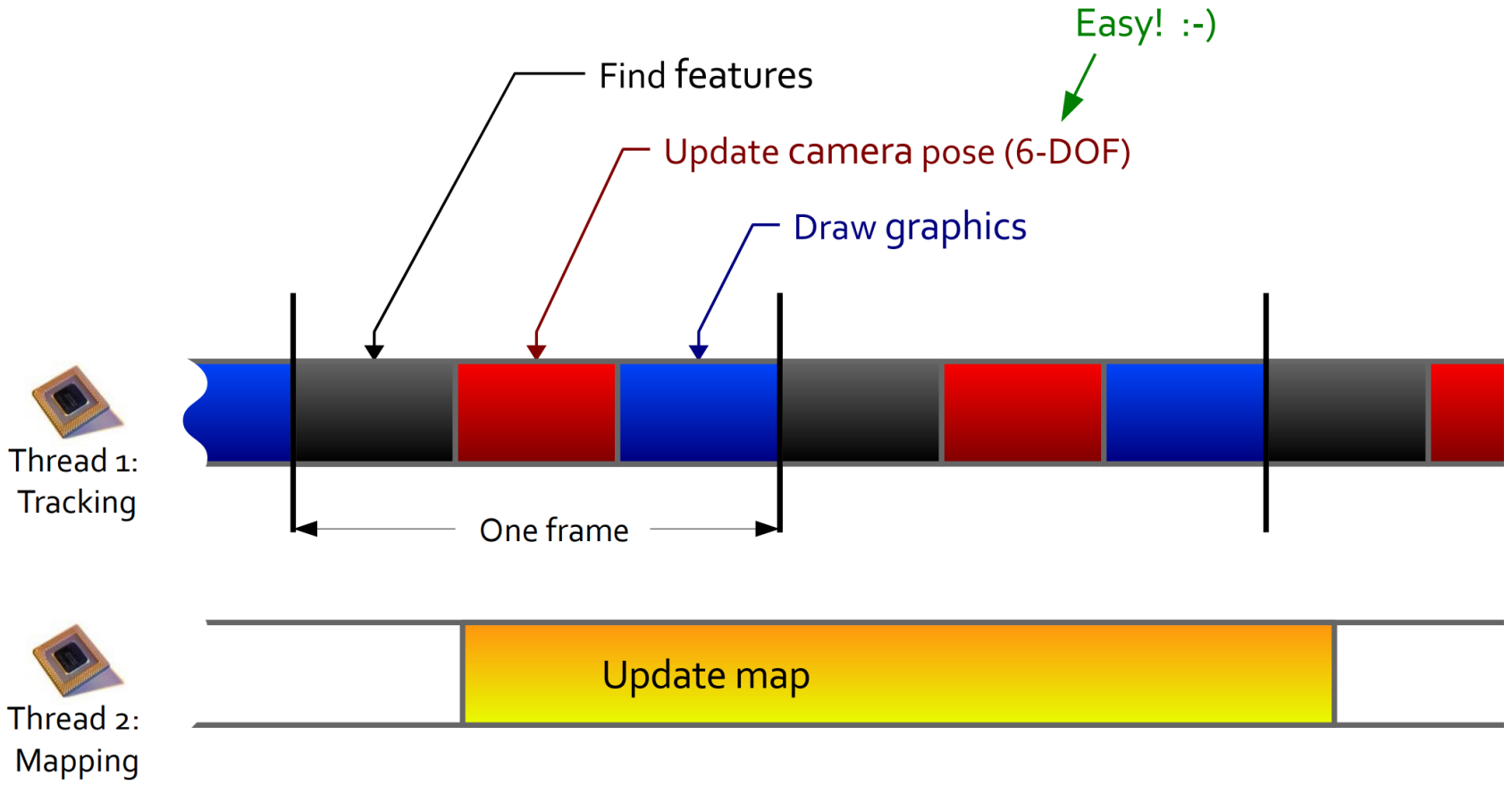
PTAM



Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces." *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007.

Parallel Tracking and Mapping

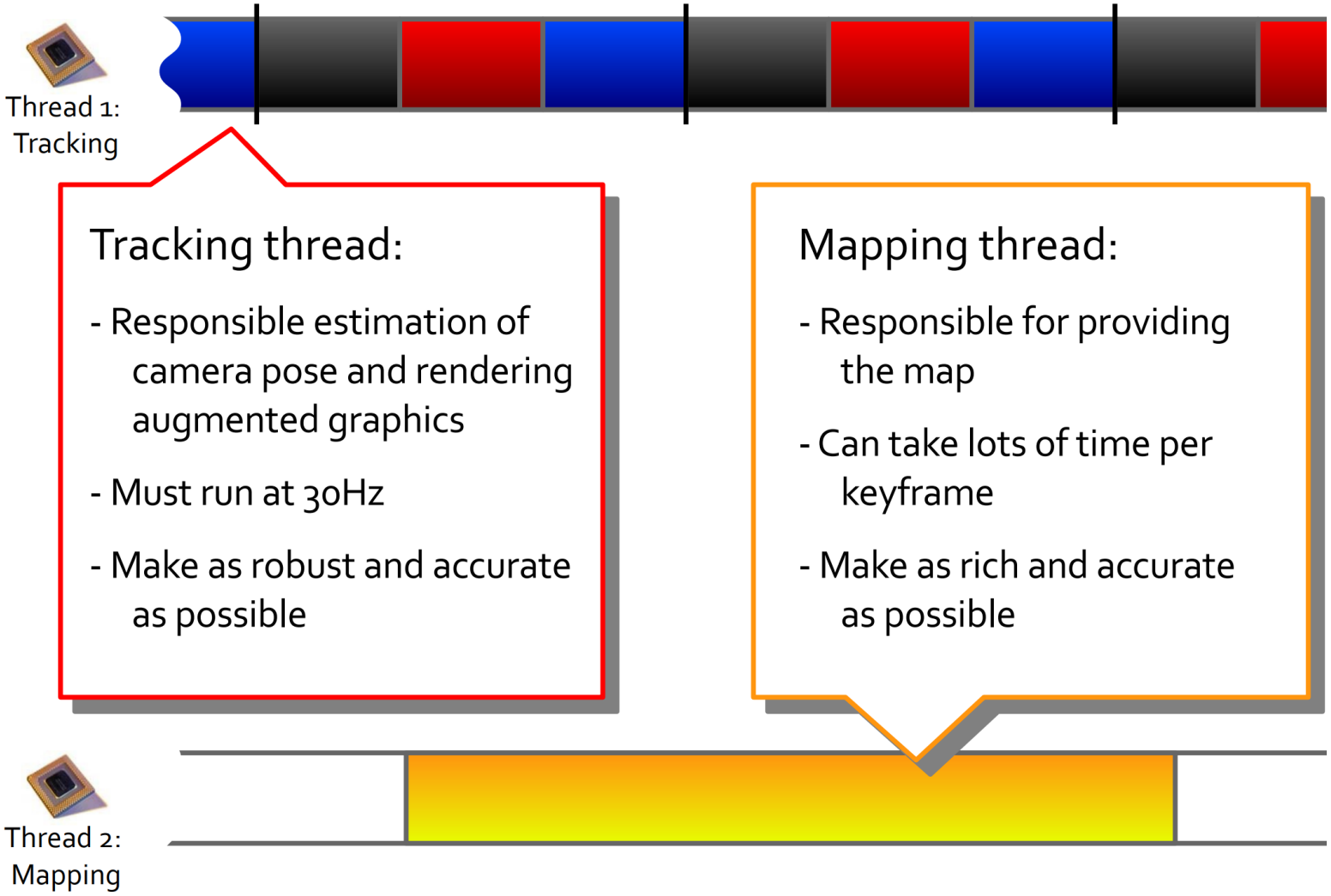
- Use dense map (of low-quality features)
- Don't update the map every frame: **Keyframes**
- Split the tracking and mapping into two threads



Klein, Georg, and David Murray. "Parallel tracking and mapping for small AR workspaces." *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007.

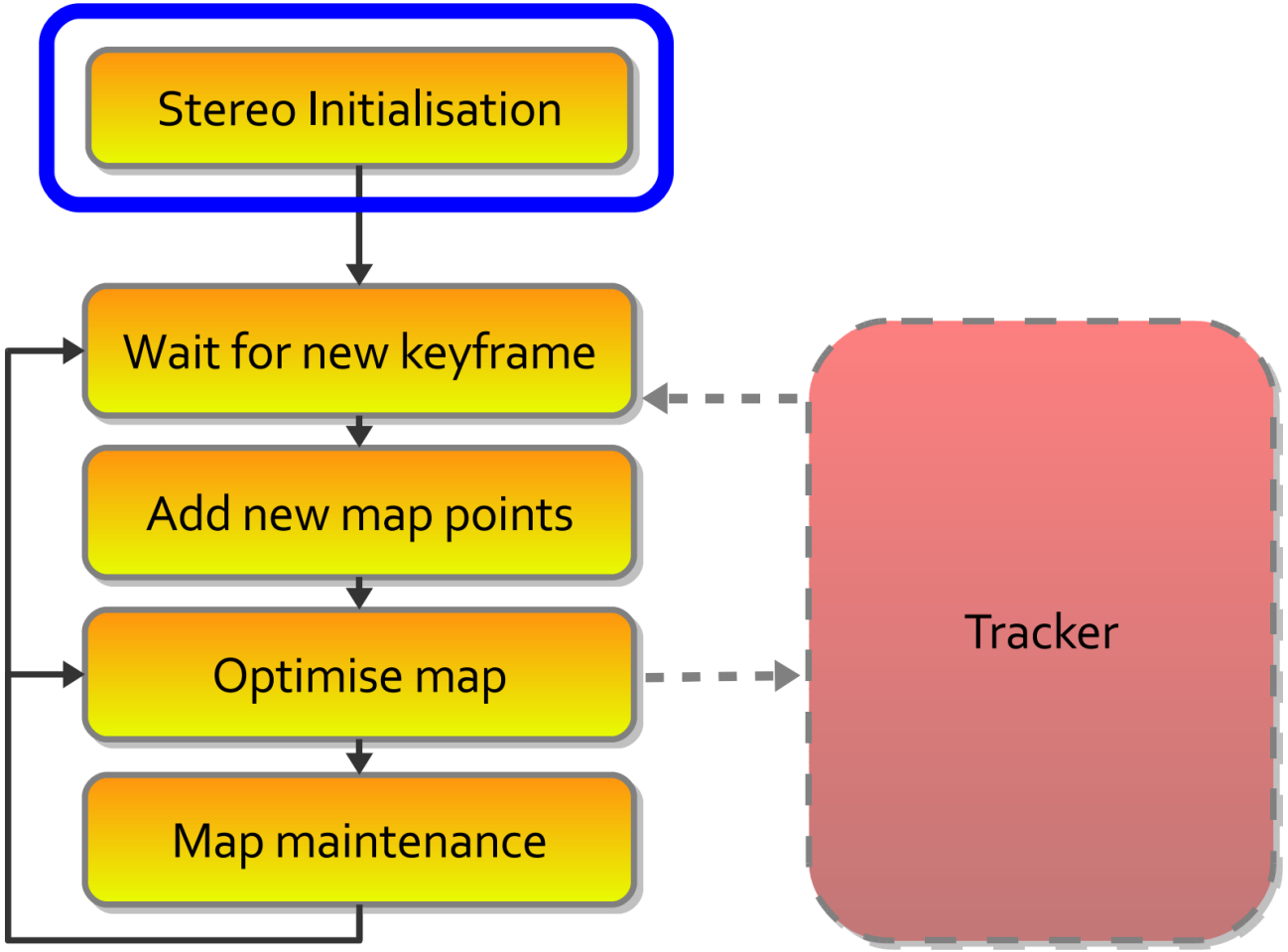
Slide from Klein & Murray 2007

Multi-threads is Common in Modern vSLAM



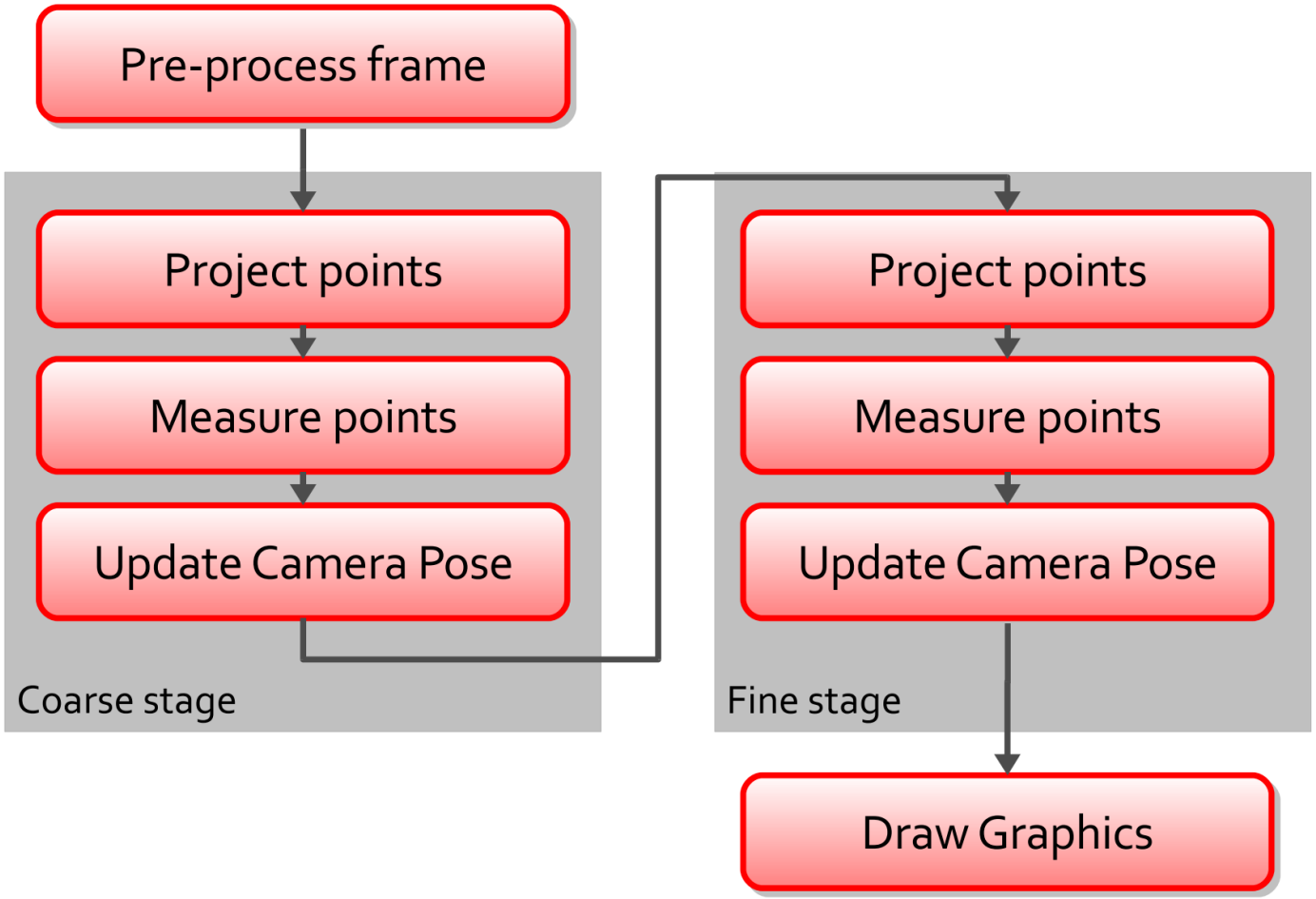
Slide from Klein & Murray 2007

Mapping Thread



Slide from Klein & Murray 2007

Tracking Thread

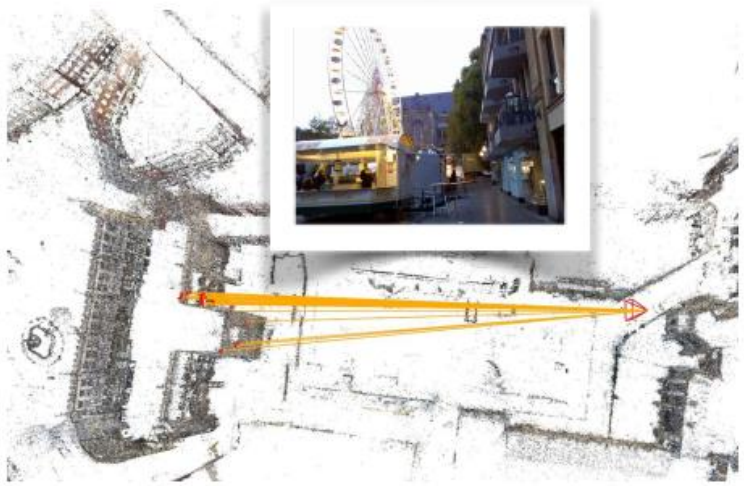


Slide from Klein & Murray 2007

Overview of Visual Place Recognition

- **We predict where the images are captured using database of**
  - Geo-tagged images
    - **Visual place recognition**

  - 3D points and descriptors
    - **Image based localization**



Slides from Sattler et. al., 2015

Keywords Related to Visual Place Recognition

Structure from Motion

Visual SLAM

Bundle adjustment

Tracking

Geometric verification (RANSAC)

Image-based localization

Loop detection & closure

Visual place recognition

ANN

Feature detection & description

Image description & indexing

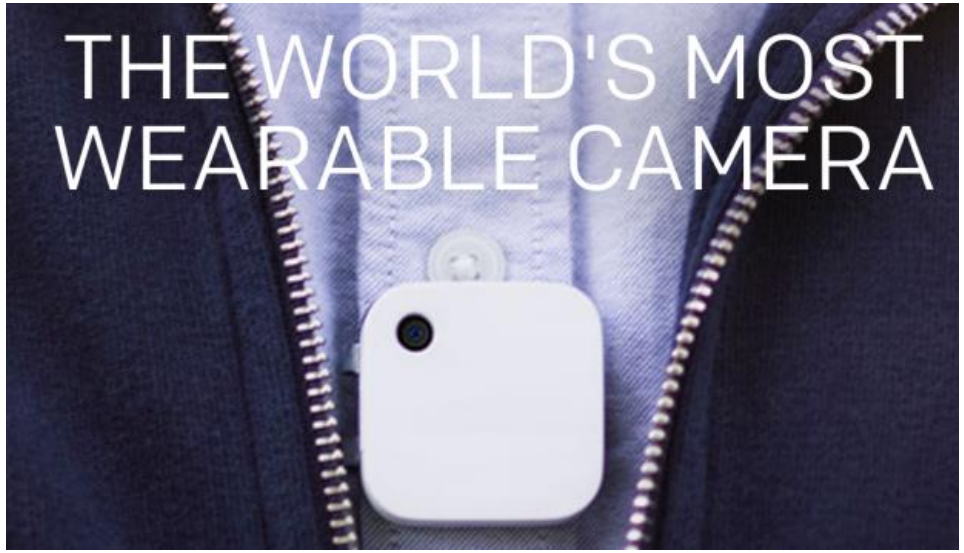
Image retrieval

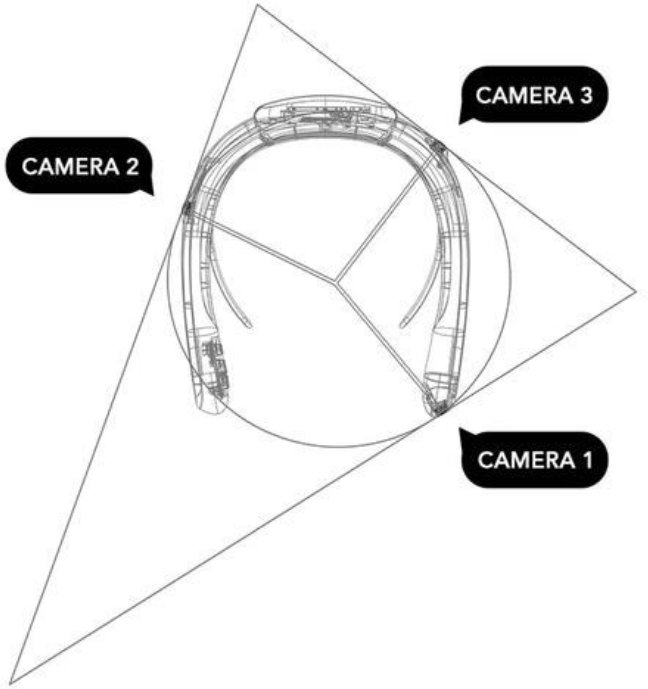
Image classification

Slides from Sattler et. al., 2015

Potential Application of Visual Place Recognition

- Accelerate incremental SfM and SLAM
  - Localization and navigation of robots and cars
- Integrate images to the real-world coordinates
  - Visual time capsule of the planet
- Acquiring images is becoming easier
  - e.g. Narrative Clip, FITT360





Slides from Sattler et. al., 2015

Street-level Visual Place Recognition

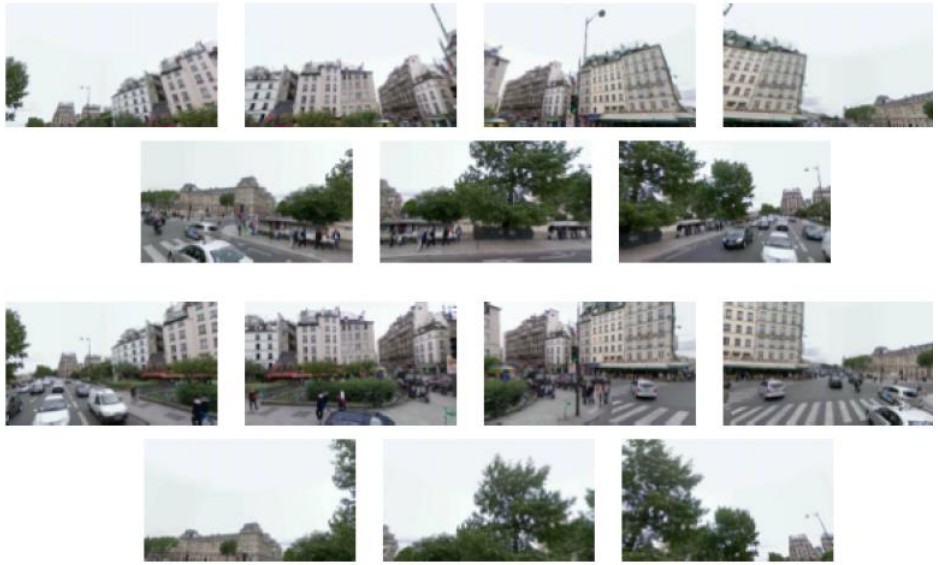
- Given a query image of a particular street or a building, we need to find one, or at most a few, images in the geotagged database depicting the same place!

Slides from Sattler et. al., 2015

Database of Geo-Tagged Images

- Typical source of geo-tagged images:
  - Google Streetview (panorama)
  - Driving/walking with a ladybug camera

- We can use original panoramas or generate perspective cutouts



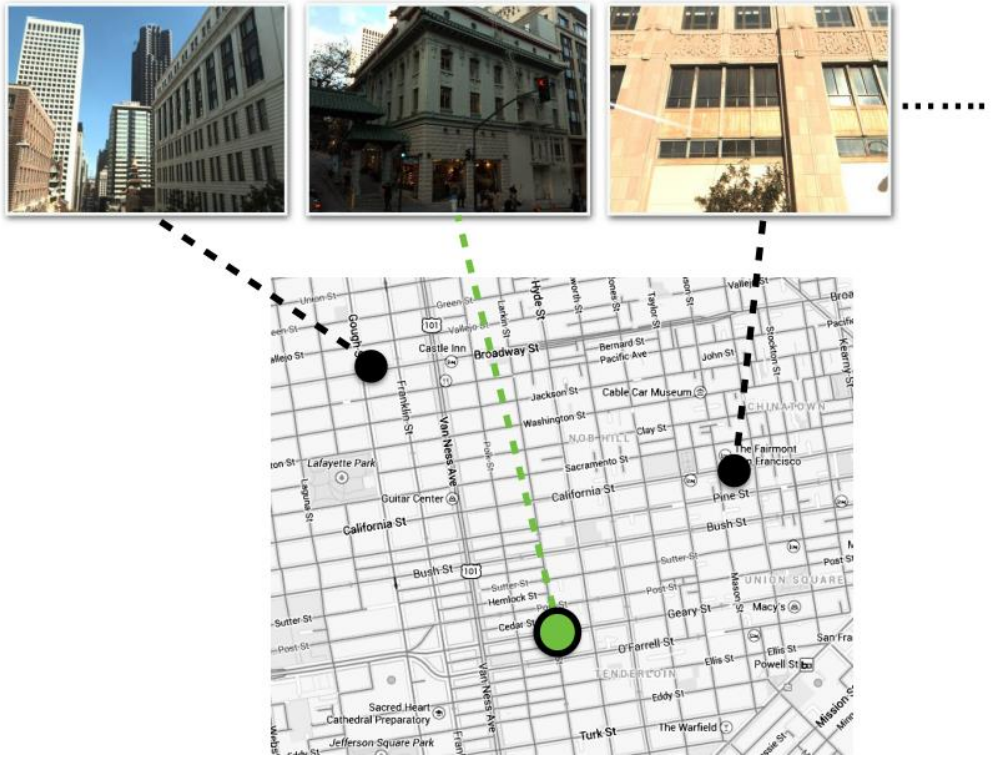
Slides from Sattler et. al., 2015

Visual place recognition by image retrieval

We are interested in "a single query testing image" and "a large image database".

Query image

Database of geo-tagged images



Slides from Sattler et. al., 2015

A standard image retrieval [Philbin-CVPR07]

Query image

Database images (millions, billions)









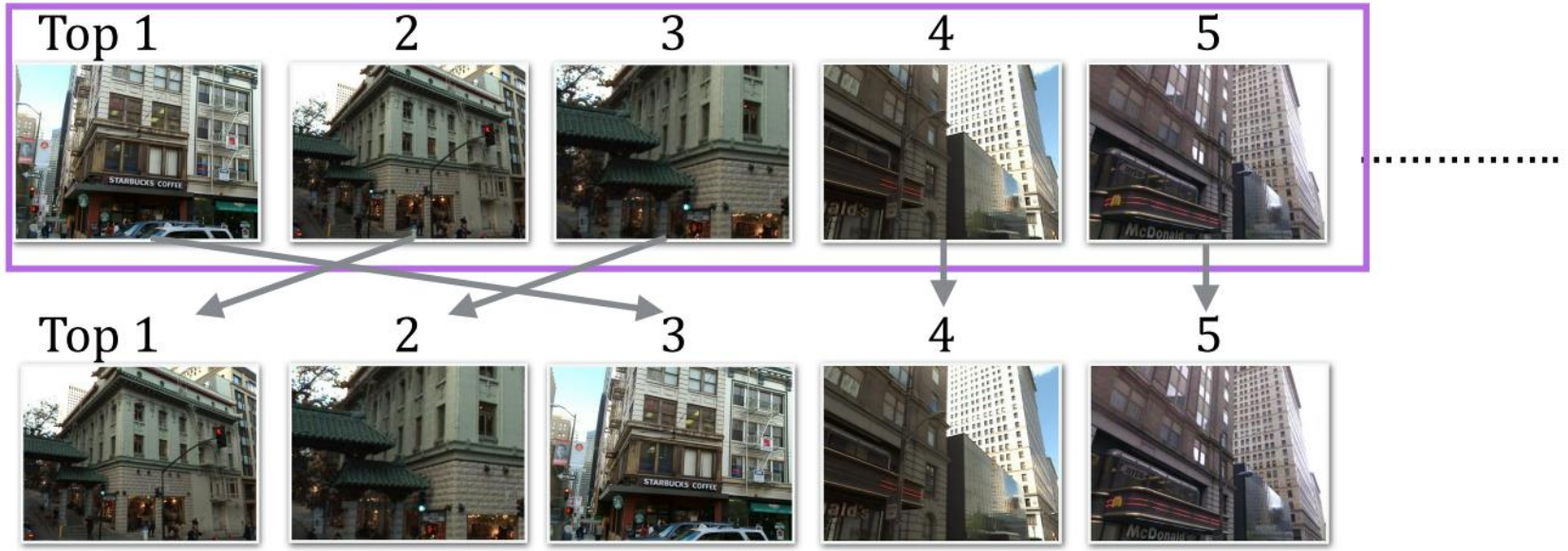
Slides from Sattler et. al., 2015

A standard image retrieval [Philbin-CVPR07]

Query image

Step1: Initial ranking/shortlisting

Step2: Re-ranking to improve the list



Step1 should be fast -> BoVW, VLAD, Fisher vectors

Step2 can be a bit more costly -> geometric verification

Slides from Sattler et. al., 2015

Visual Place Recognition Pipeline

Query image

Database images



.....

Offline

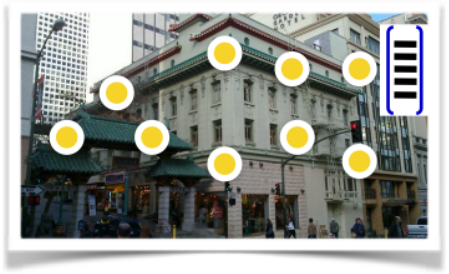
1. Feature detection & description
2. Training visual vocabulary
3. Image description

4. Feature detection & description
5. Image description

6. Initial ranking
7. Re-ranking with geometric verification

Slides from Sattler et. al., 2015

Image description



Quantization (aggregation)

Bag of visual words (VLAD, FV)

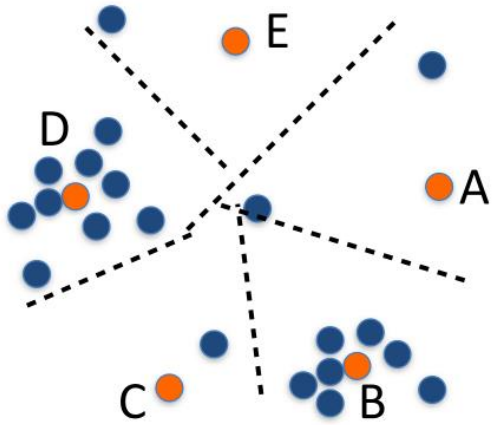
1. Feature detection & description, e.g. SIFT
2. Represent the set of features
   - as a sparse histogram (BoVW)
   - as an aggregated vectors (VLAD, FV)

Slides from Sattler et. al., 2015

How to obtain a visual vocabulary

Training data

Detect & describe features (e.g. SIFT)



Find representative features
(e.g. approximate k-means)

Visual words (centroids)

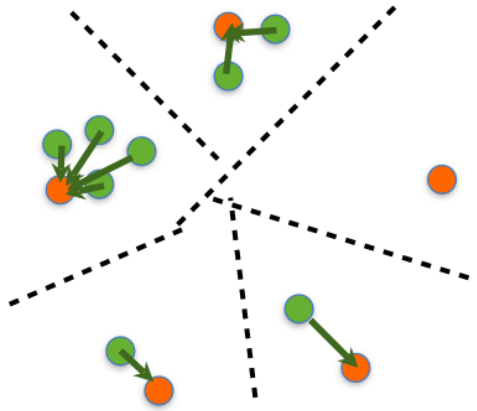
- Feature (128D)
- Centroid (128D)

Slides from Sattler et. al., 2015

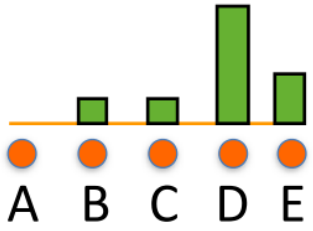
How to make BoVW

For each image,

1. we assign features to the visual word closest in the feature space.
2. we build a histogram by voting.



Feature (128D)
Centroid (128D)



Histogram

Visual words (centroids)

tf-idf weighted and L2 normalized

Design of the weights directly impacts the retrieval results!

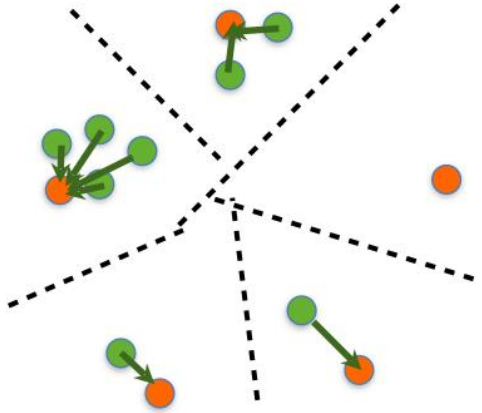
See also [Jegou-CVPR09, Zheng-CVPR13]

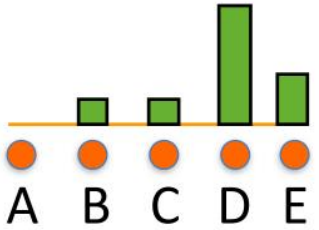
Slides from Sattler et. al., 2015

How to make BoVW

For each image,

1. we assign features to the visual word closest in the feature space.
2. we build a histogram by voting.





Histogram

- Feature (128D)
- Centroid (128D)

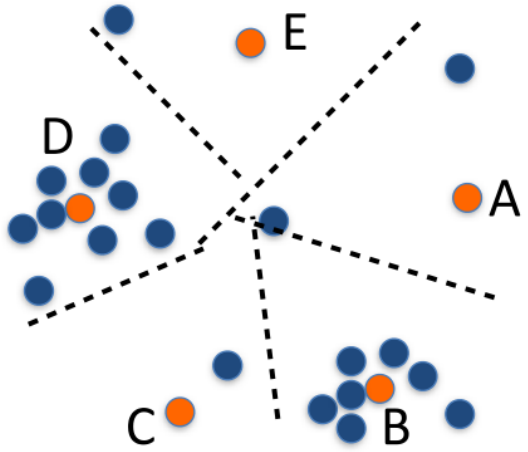
We compute BoVW vectors for both query & database images. Then, seek the best match.

Slides from Sattler et. al., 2015

How to obtain a visual vocabulary (centroids) for VLAD

Training data

Detect & describe features (e.g. SIFT)



Find representative features
(e.g. approximate k-means)

A B C D E

Visual words (centroids)

Feature (128D)

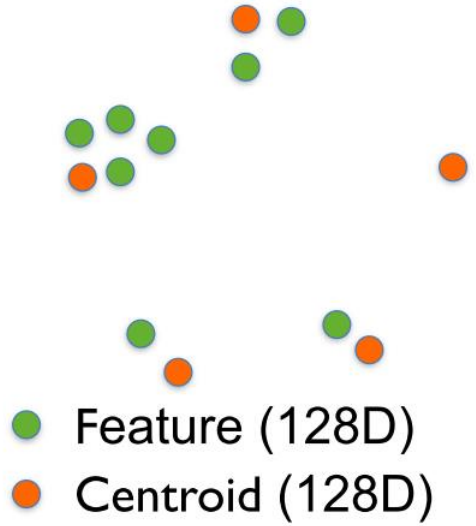
Centroid (128D)

The paradigm is the same as BoVW but VLAD uses a fewer centroids (64 to 512)

Slides from Sattler et. al., 2015

How to make VLAD

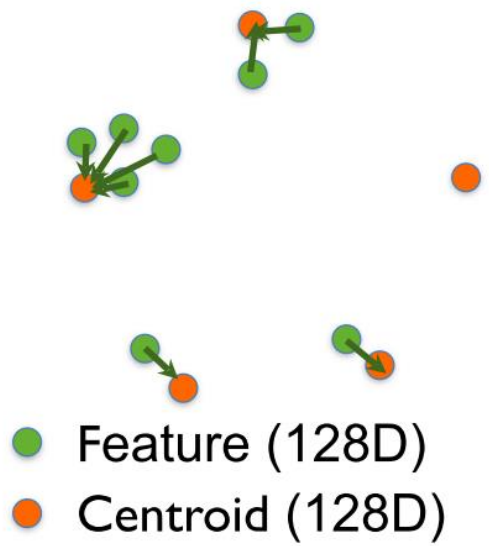
For each image,
assign features to centroids



Slides from Sattler et. al., 2015

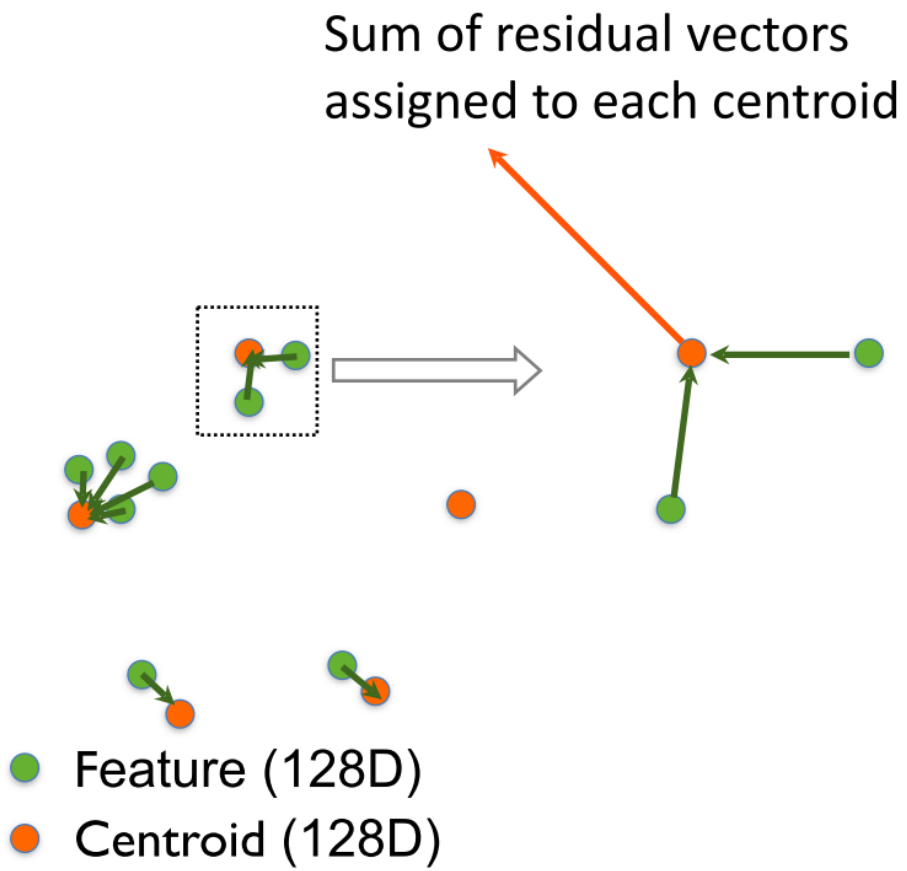
How to make VLAD

For each image,
assign features to centroids



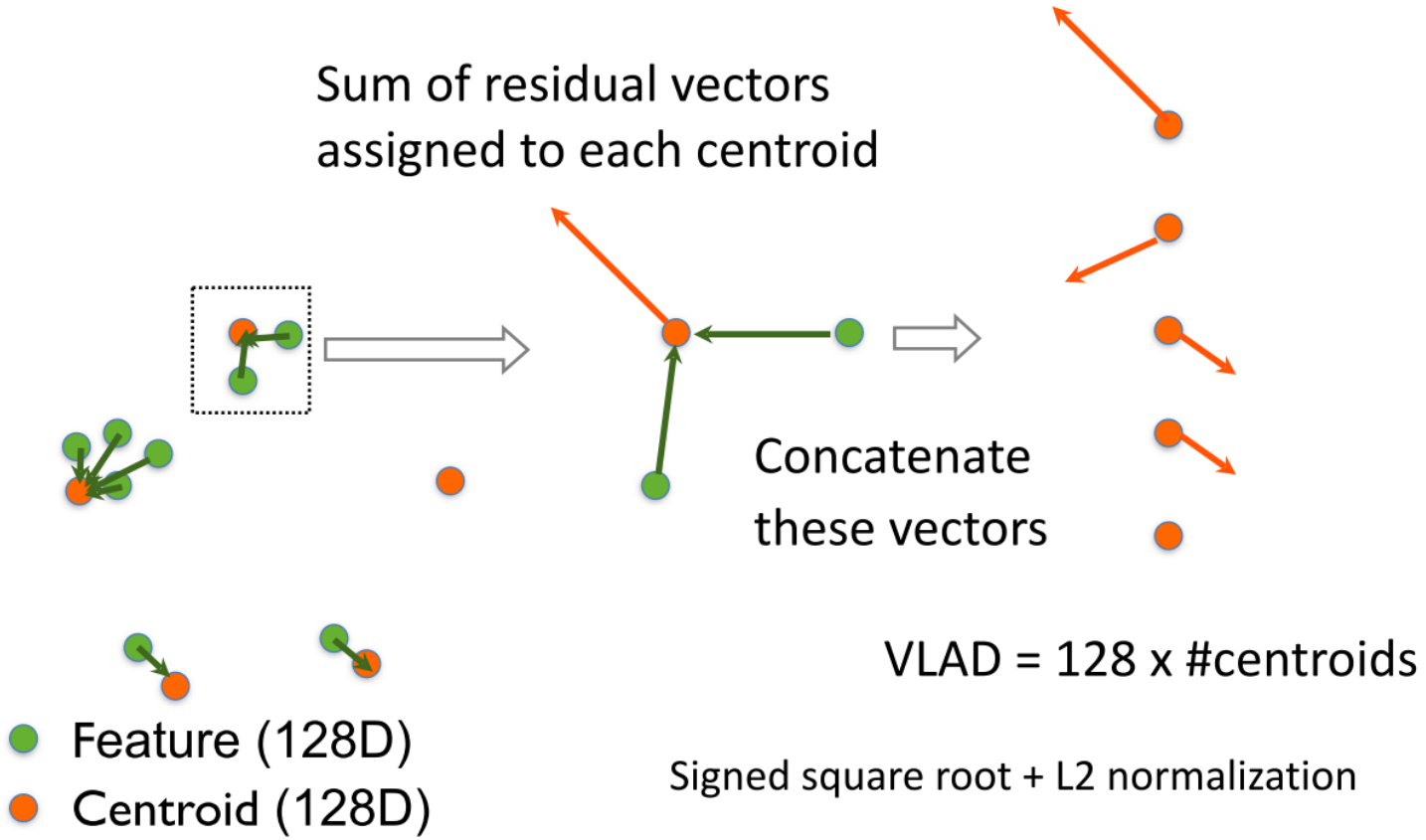
Slides from Sattler et. al., 2015

How to make VLAD



Slides from Sattler et. al., 2015

How to make VLAD



Slides from Sattler et. al., 2015

BoVW

Sparse (with a large vocab.)

- Inverted file indexing
- Size per image ~=
  #features x 8 byte
  (4 bytes for the index and 4 bytes for the weight)
  E.g. 4000 x 8 = 32K byte

VLAD (FV)

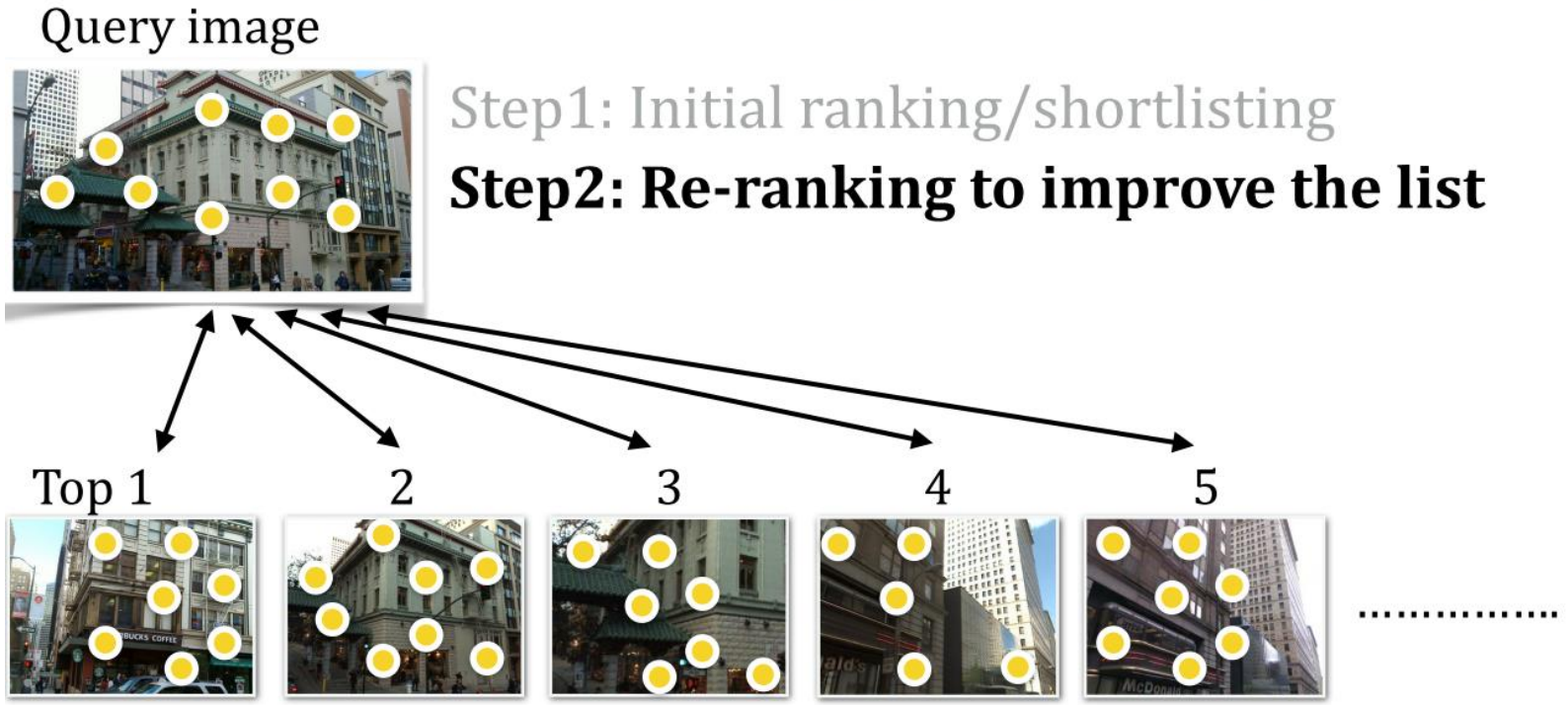
Dense

- ANN, product quantization
- Size per image ~=
  #centroids x desc dim. x 4 byte
  (4 bytes (single precision) per dimension}
  E.g. 256 x 128 x 4 = 65K byte

+ No extra memory requirement to use densely detected features

Slides from Sattler et. al., 2015

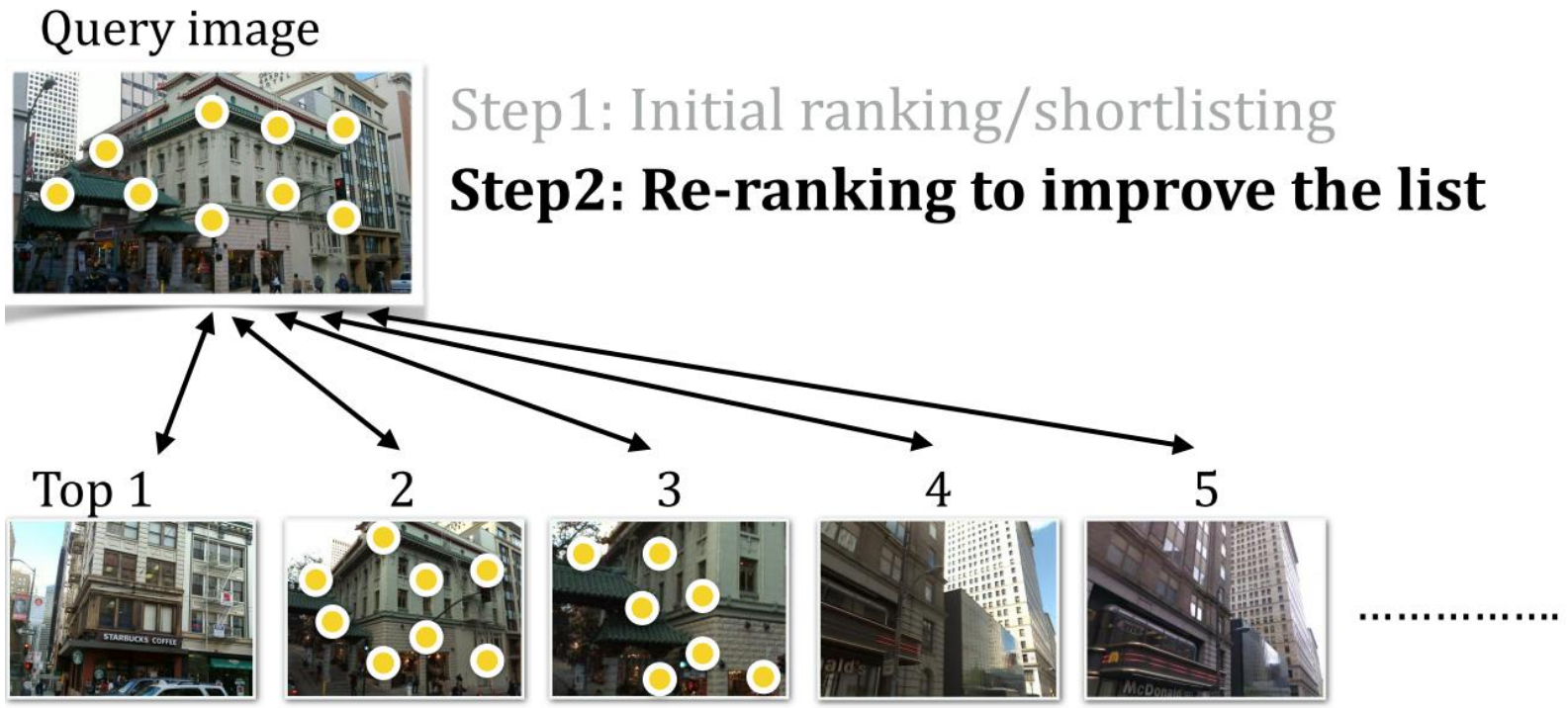
Review: the baseline image retrieval (BoVW)



- Generate tentative matches of features
- Verify the tentative matches by fitting geometric transformations (affine, homography, ...), i.e. RANSAC
- Re-rank the shortlisted images by the number of verified matches

Slides from Sattler et. al., 2015

Review: the baseline image retrieval (BoVW)



- Generate tentative matches of features
- Verify the tentative matches by fitting geometric transformations (affine, homography, ...), i.e. RANSAC
- Re-rank the shortlisted images by the number of verified matches

Slides from Sattler et. al., 2015

NetVLAD

CNN architecture for weakly supervised place recognition

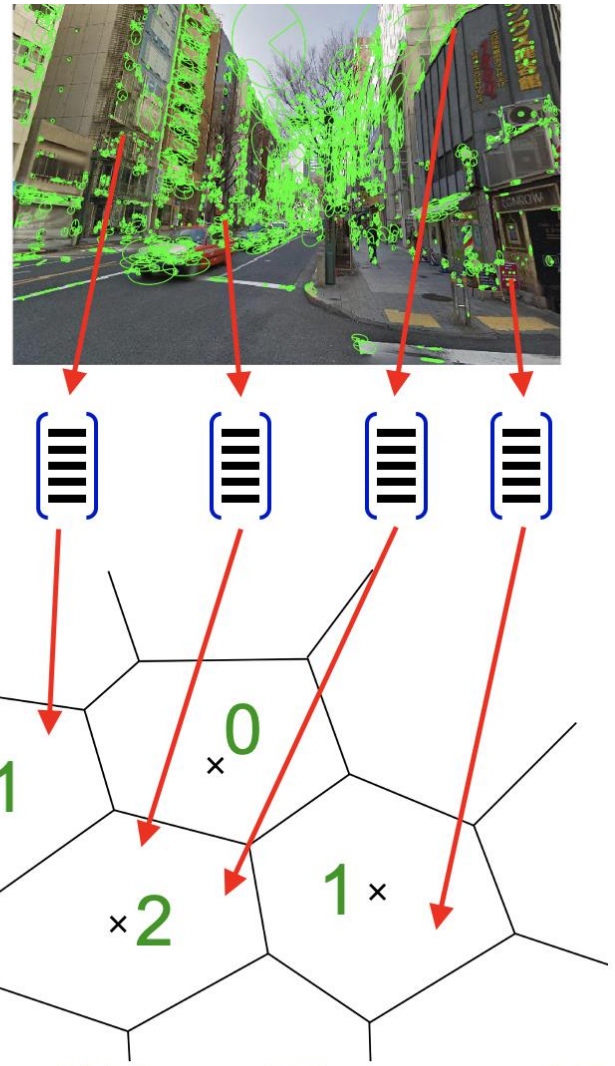
https://arxiv.org/pdf/1511.07247.pdf

Review: Pooling local descriptors - Bag-of-Words (BoW)

0/1 assignment of desc. *i* to cluster *k*

$$B(k) = \sum_{i=1}^{N} a_k(x_i)$$

Sum over all N descriptors in the image



B = [1, 0, 2, 1, …]

[Sivic and Zisserman 03]

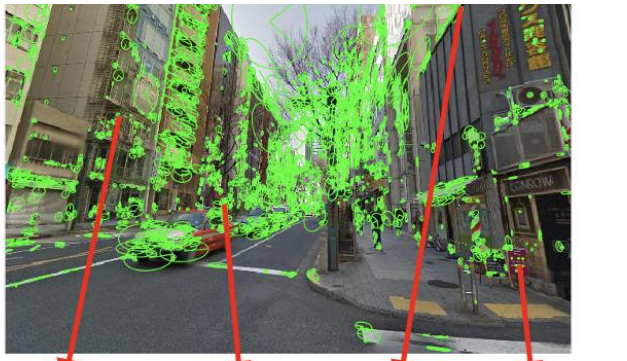
Review: Vector of Locally Aggregated Descriptors (VLAD)

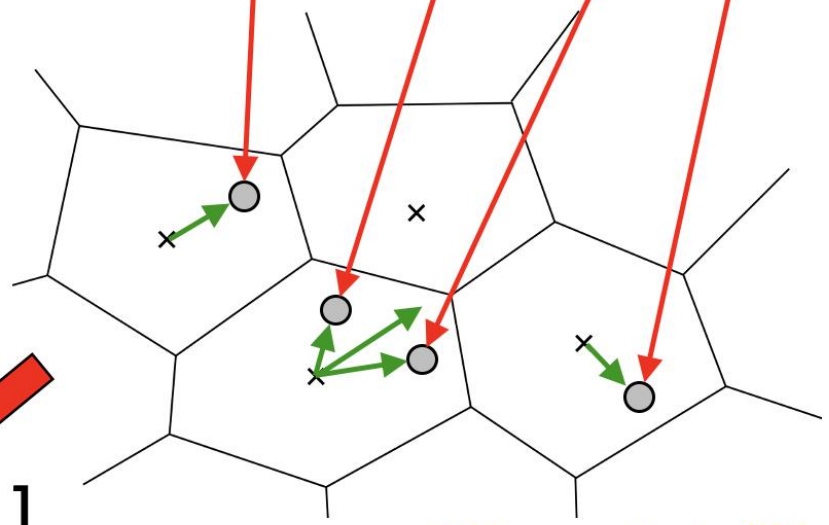
0/1 assignment of desc. *i* to cluster *k*

$$V(:,k) = \sum_{i=1}^{N} a_k(x_i)(x_i - c_k)$$

Residual vector

Sum over all N descriptors in the image

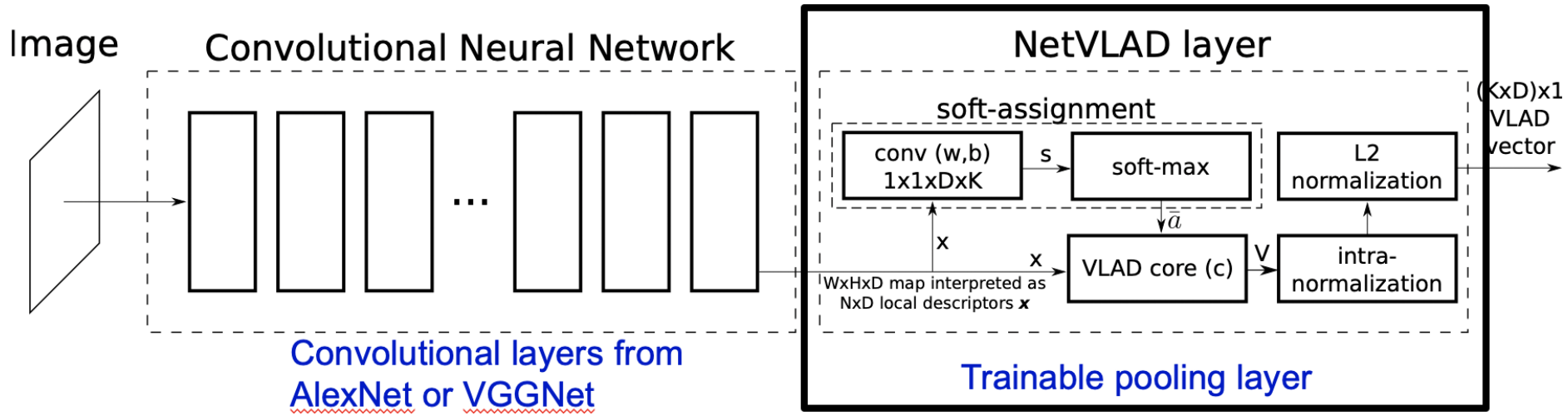




V = [↗, . . , ↗, ↘, …]

[Jégou et al. 10]

NetVLAD: Trainable pooling layer



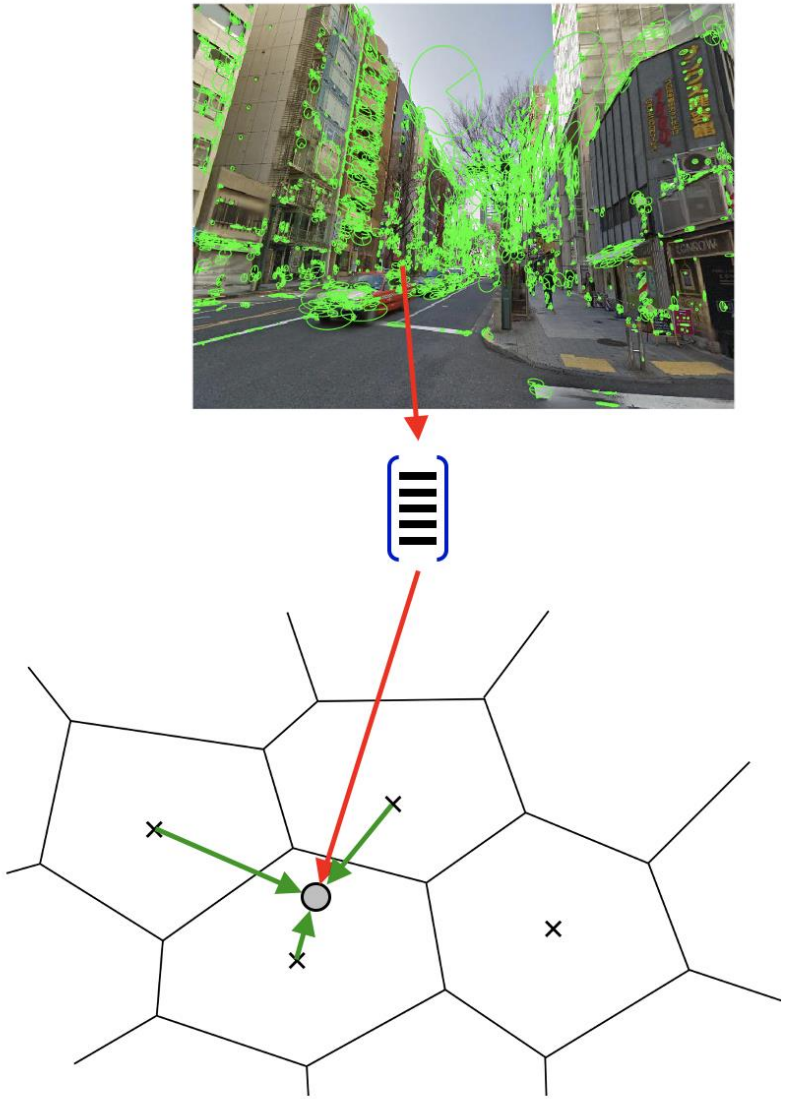
1. Part of the CNN architecture
2. Trainable end-to-end

NetVLAD: Trainable pooling layer

0/1 assignment of desc. *i* to cluster *k*

$$V(:,k) = \sum_{i=1}^{N} a_k(x_i)(x_i - c_k)$$

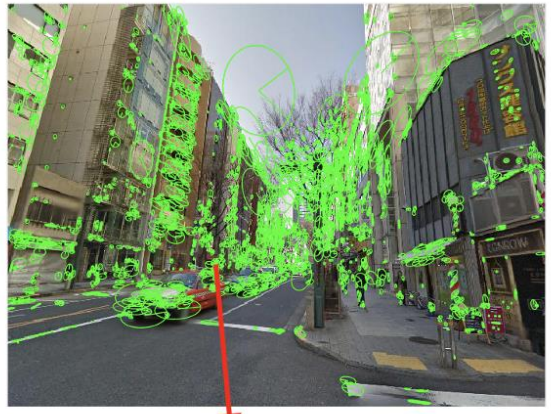
Replace hard-assignment of descriptors to clusters with soft-assignment

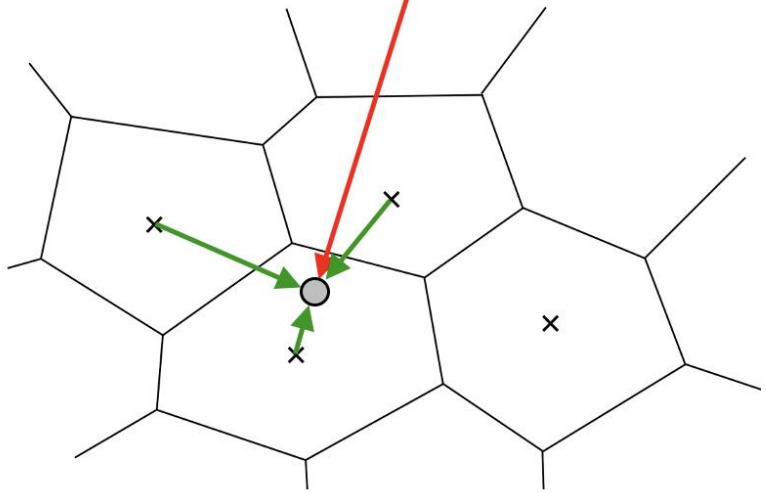


NetVLAD: Trainable pooling layer

soft assignment of desc. *i* to cluster *k*

$$V(:,k) = \sum_{i=1}^{N} \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b_{k'}}} (x_i - c_k)$$

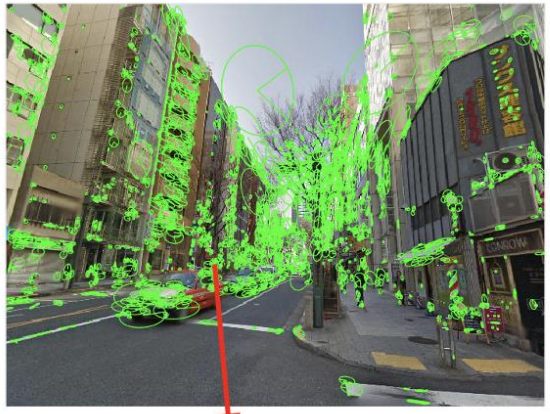


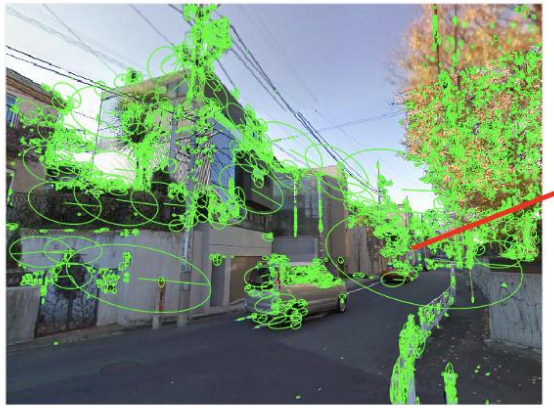


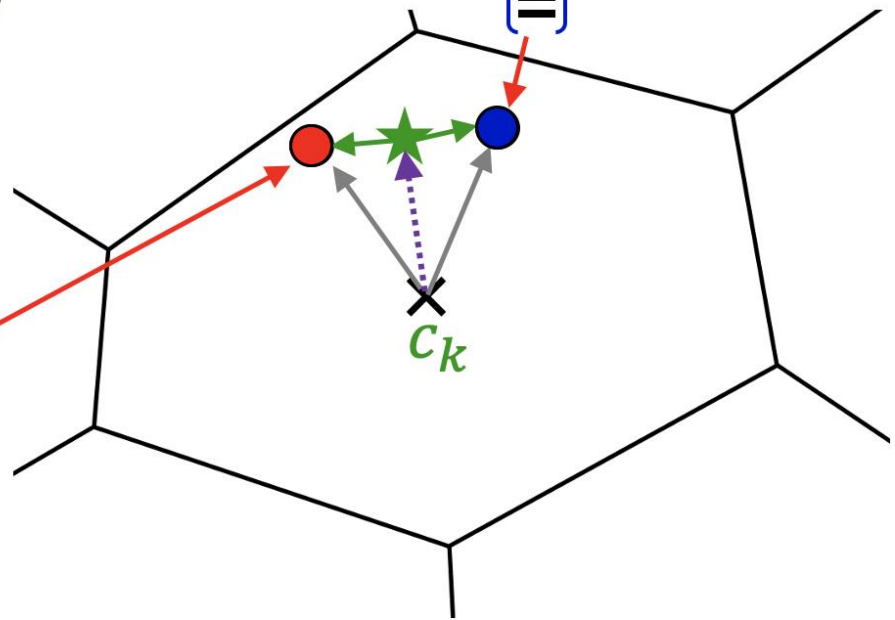
NetVLAD: Trainable pooling layer

Decouple assignment (w_k b_k) from anchor point c_k

$$V(:,k) = \sum_{i=1}^{N} \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b_{k'}}} (x_i - c_k)$$





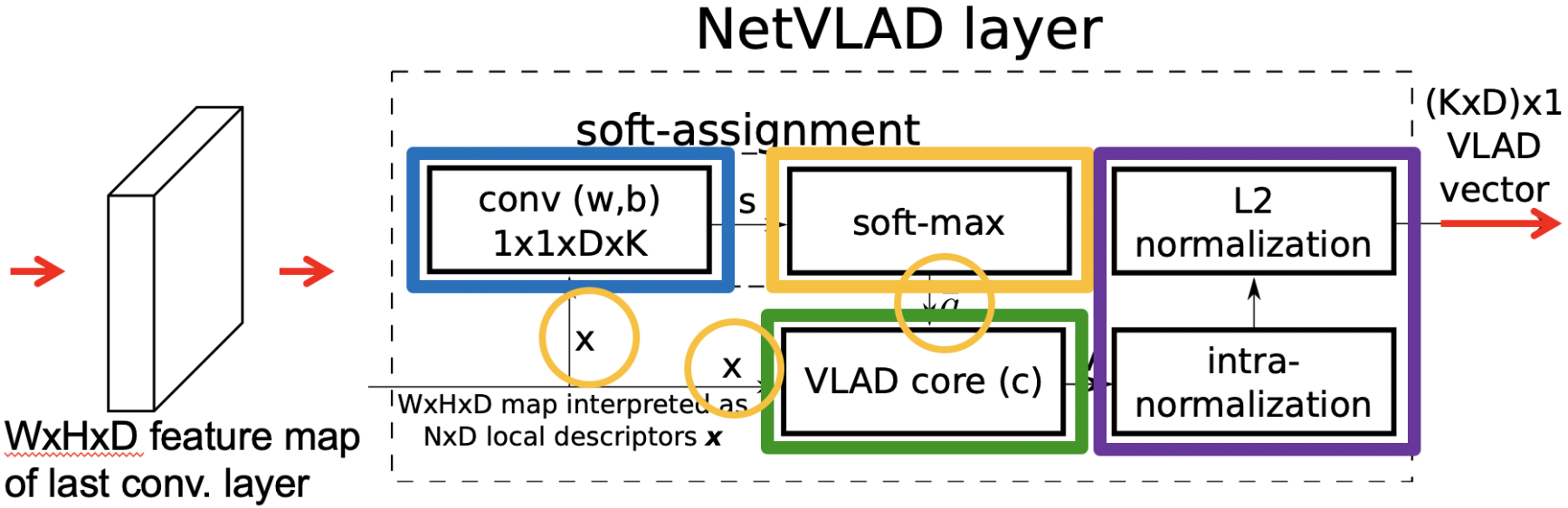


NetVLAD as a trainable layer

Convolution with filters $\{w_k\}$ and biases $\{b_k\}$

$$V(:,k) = \sum_{i=1}^{N} \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b_{k'}}} (x_i - c_k)$$

Soft-max $\rightarrow \bar{a}$ Residual vector



How to Annotate Training Data?

- The same locations at different times and seasons

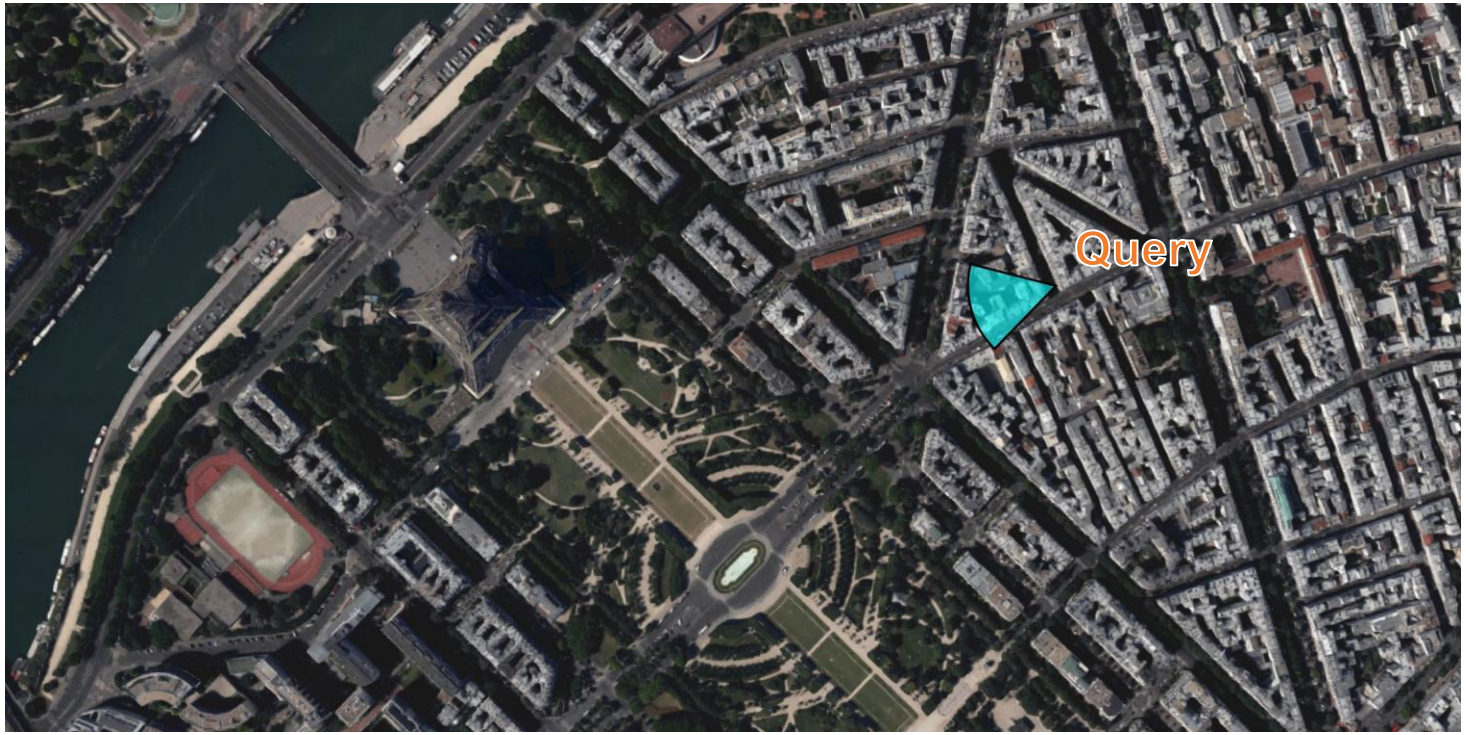
How to Annotate Training Data?

- The same locations at different times and seasons

How to Annotate Training Data?

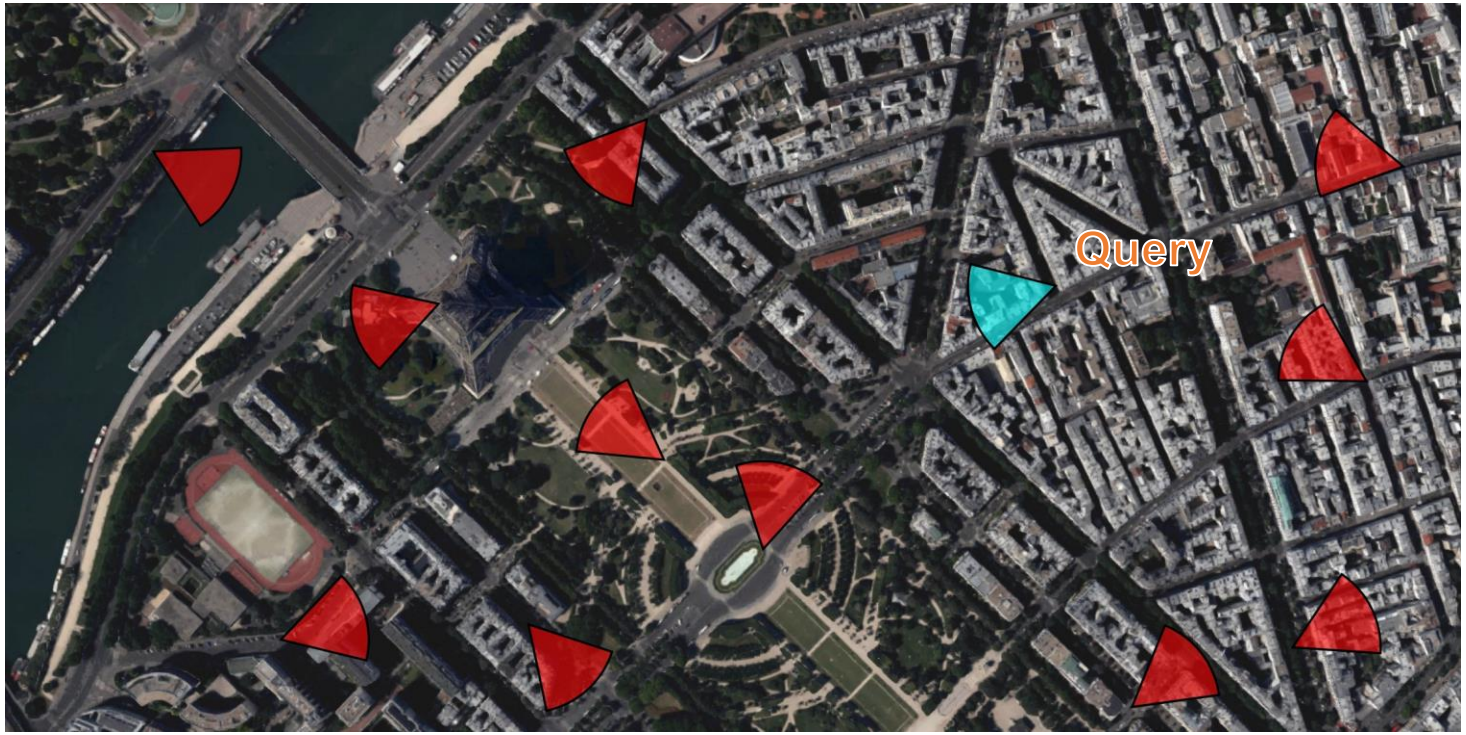
- The same locations at different times and seasons

GPS!



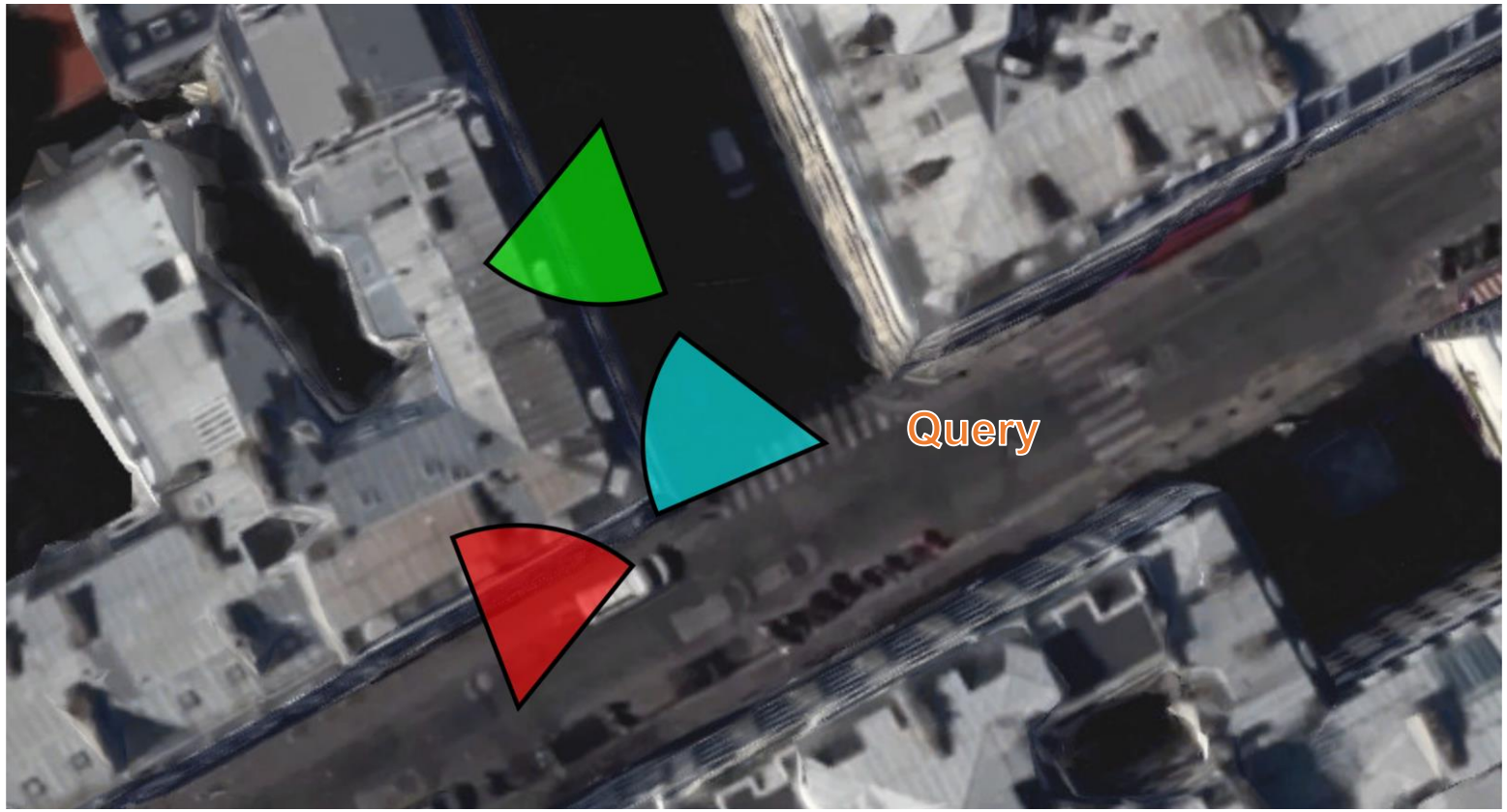
But GPS provides only weak supervision

- Given a query, GPS gives us:
  - *Definite* negatives :
    - geographically far from the query



But provides only weak supervision

- Given a query, GPS gives us:
  - *Potential* positives:
    - geographically close to the query



Which loss function to ues? Weakly supervised ranking loss

- Inspired by the triplet loss
  [Schultz and Joachims 04, Weinberger et al. 06, Wang et al. 14, Schroff et al. 15]

For a tuple (q, $\{p_i^q\}$, $\{n_j^q\}$) :

$$L_\theta = \sum_j l\left(\min_i d_\theta^2(q, p_i^q) + m - d_\theta^2(q, n_j^q)\right)$$

hinge loss $l(x) = \max(x, 0)$

margin

Sum over negatives

Distance to the best potential positive

Distance to the negative

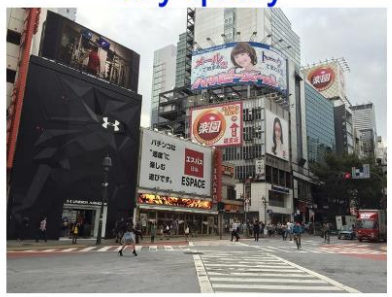
- Can be optimized with Stochastic Gradient Descent

Experiments: Datasets and evaluation protocol

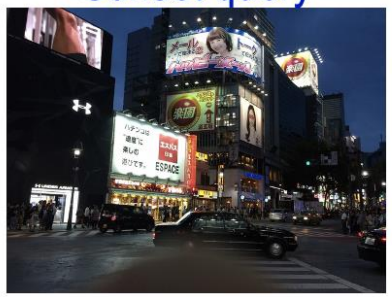
- Pittsburgh [Torii et al. 13]
  - > Database: 250k images from Street View
  - > Queries: 24k images from Street View at other times

- Tokyo 24/7 [Torii et al. 15]
  - > Database: 76k images from Street View
  - > Queries: 315 images from mobile phone cameras

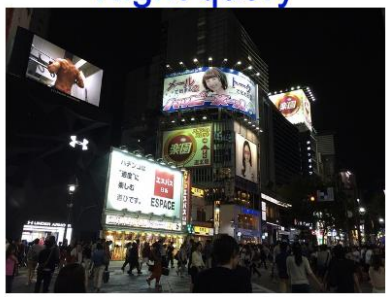
Day query



Sunset query

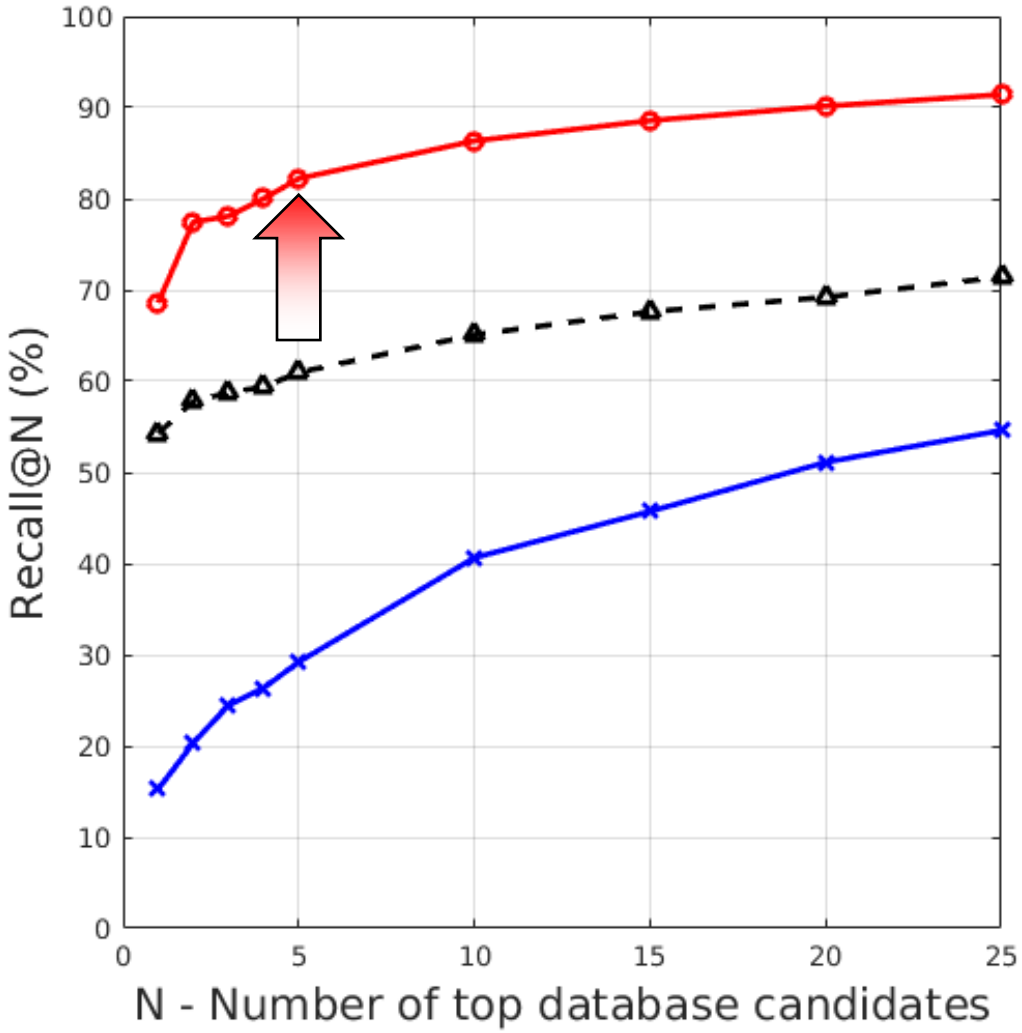


Night query



Database image

New state-of-the-art result on all datasets

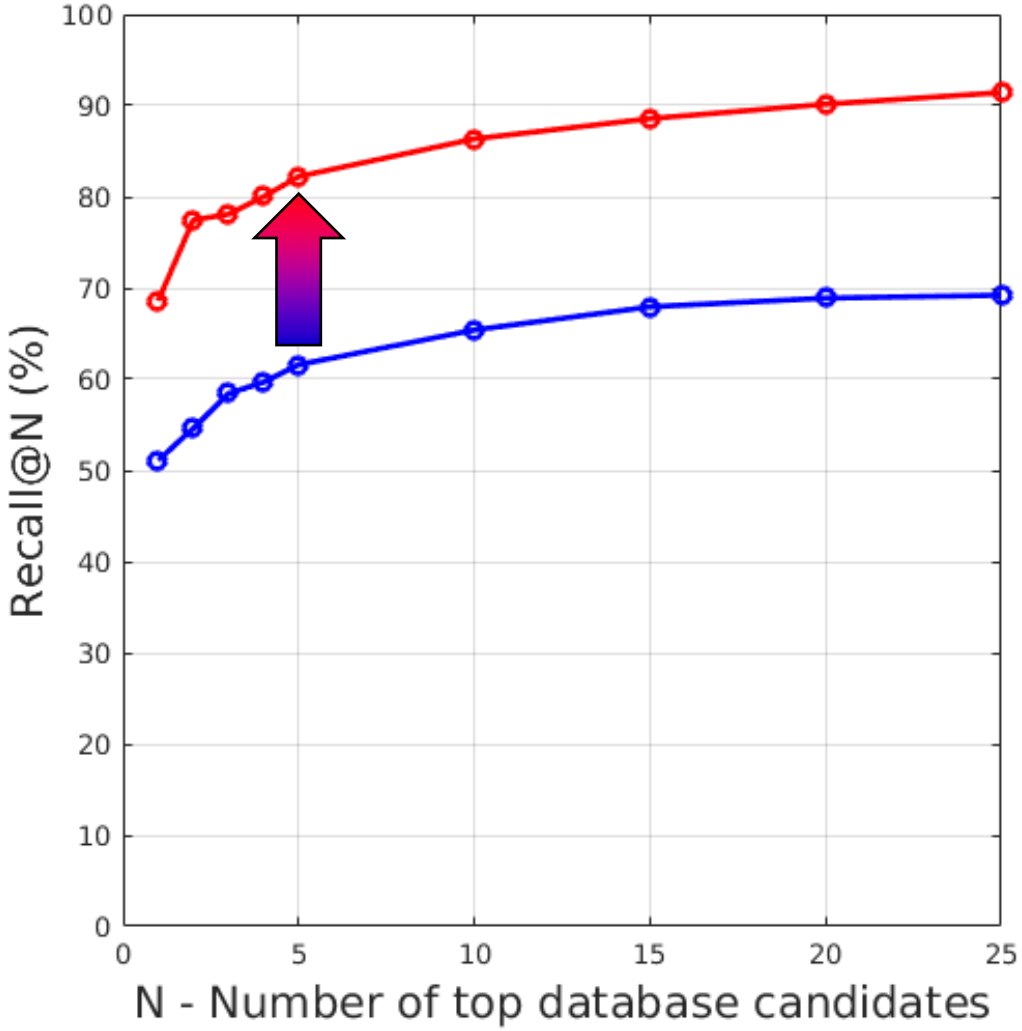


Ours: Trained NetVLAD

Off-the-shelf Max pooling
[*Razavian et al.* ICLR'15]

recall@5
60.9%
82.2%
35.0%

End-to-end training vs off-the-shelf

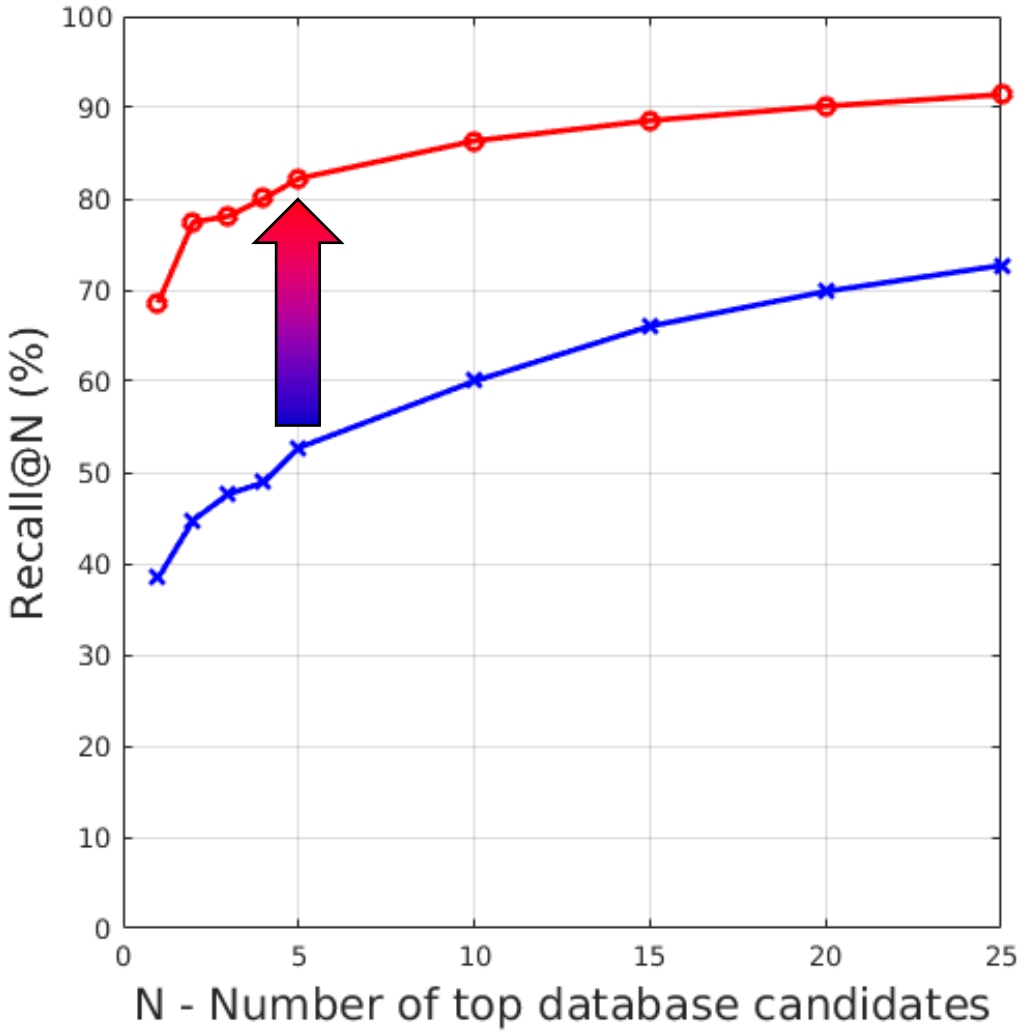


Ours: Trained NetVLAD

Off-the-shelf VLAD

recall@5
61.6%
82.2%
33.4%

NetVLAD vs Max

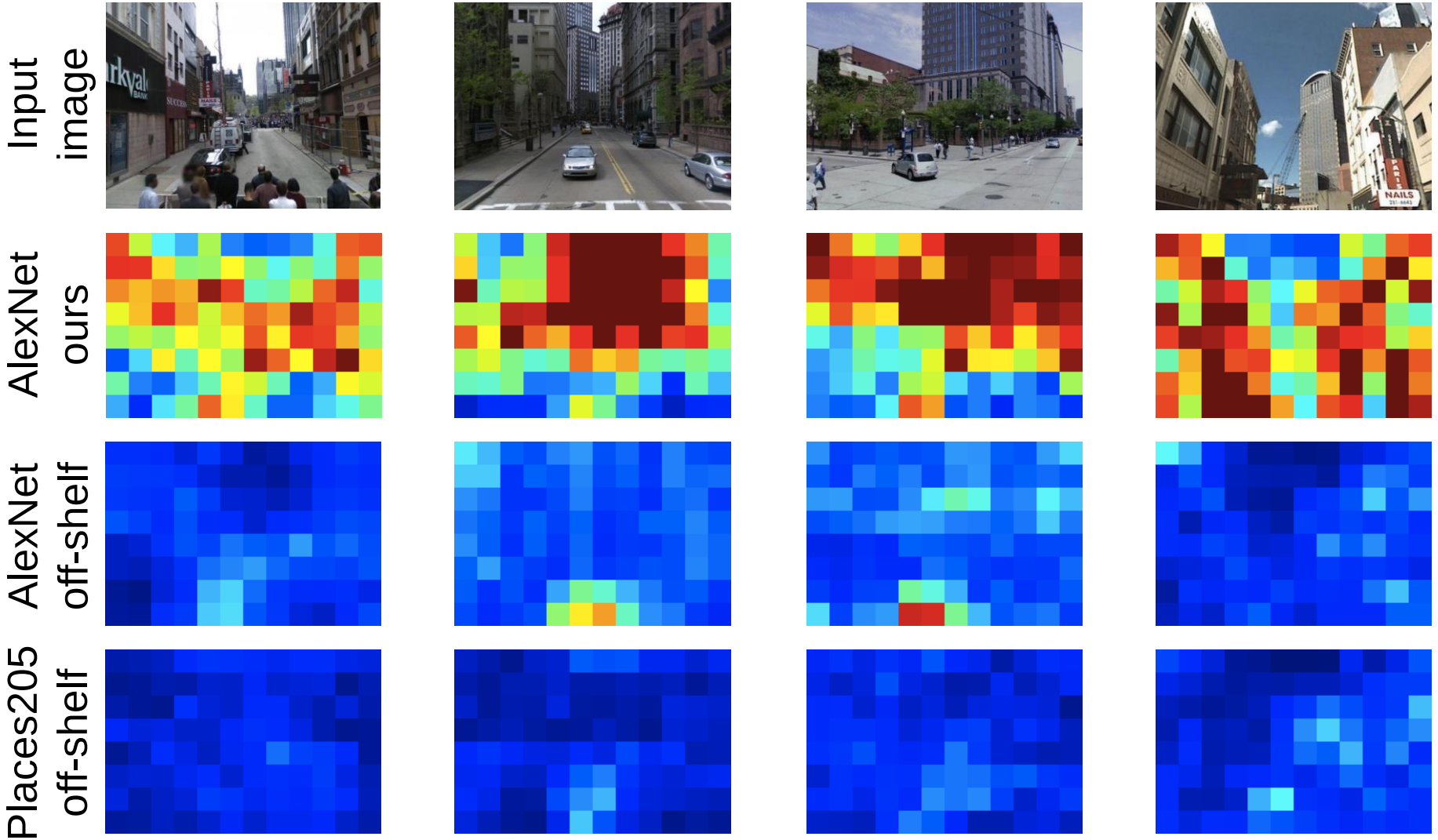


Ours: Trained NetVLAD

Ours: Trained Max pooling

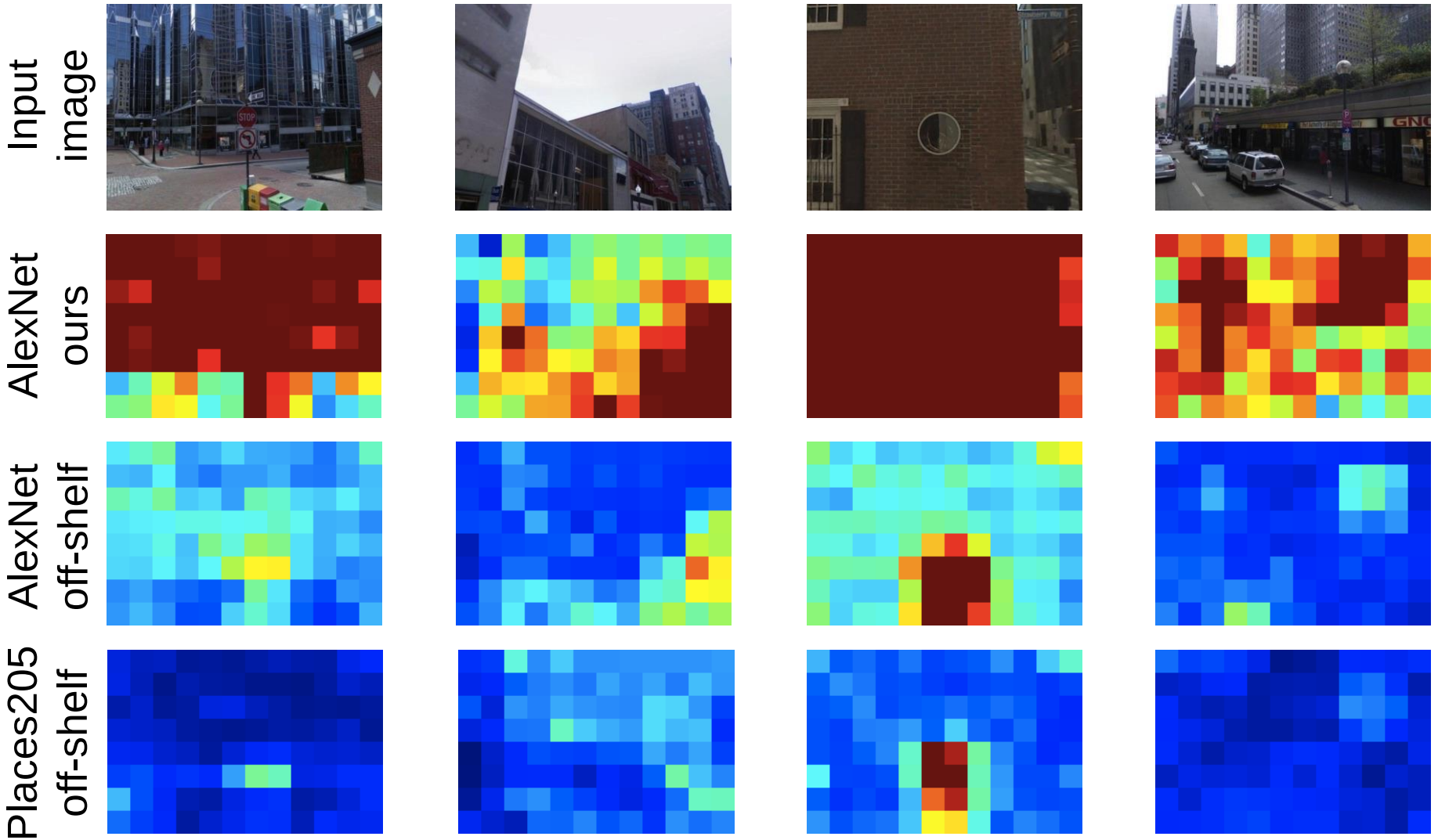
recall@5
52.7%
82.2%
56.0%

What has been learnt?



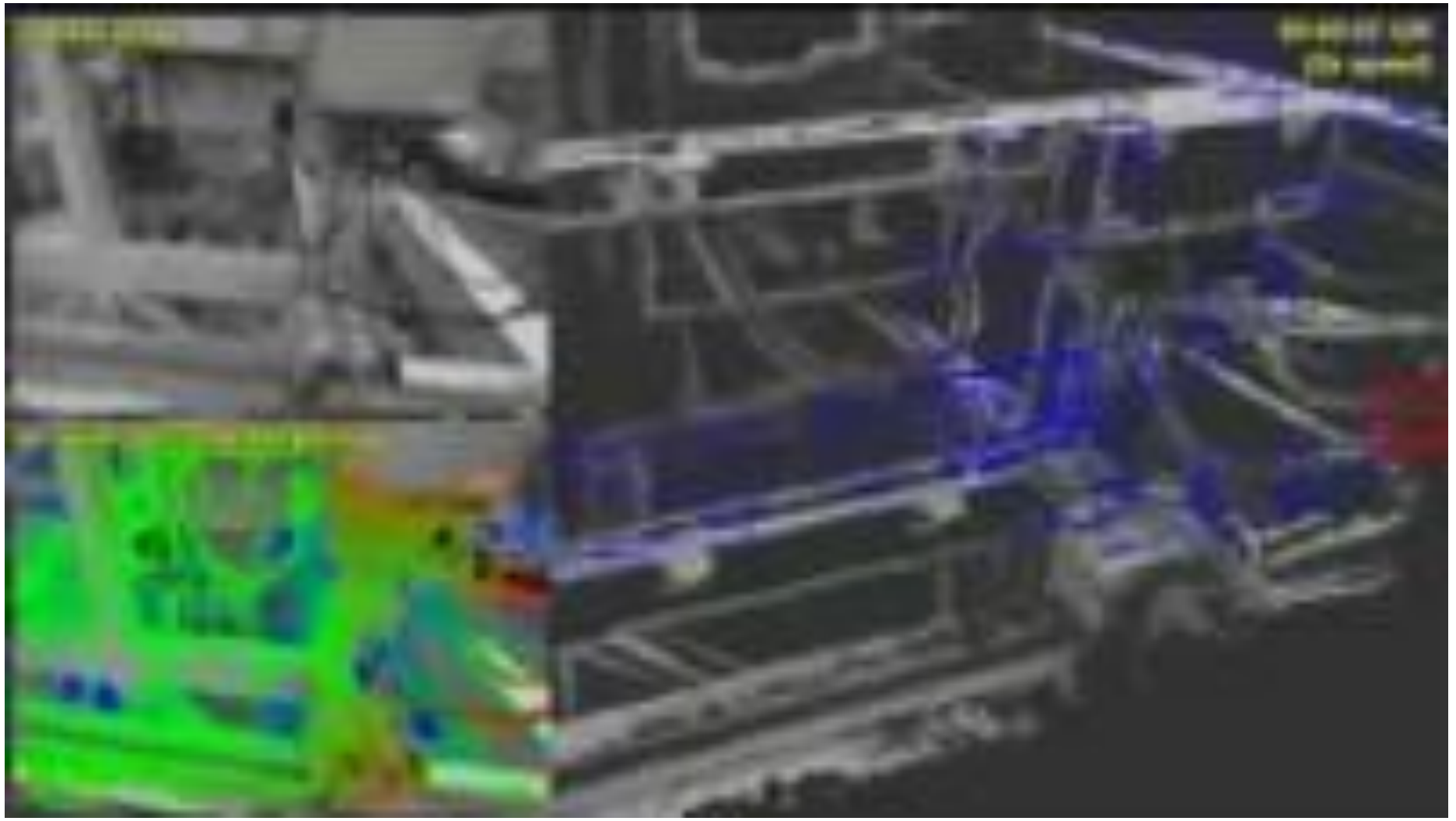
[Zeiler and Fergus 14]

What has been learnt?



[Zeiler and Fergus 14]

LSD-SLAM: Another vSLAM Pipeline



https://vision.in.tum.de/research/vslam/lsdslam

LSD-SLAM: Another vSLAM Pipeline

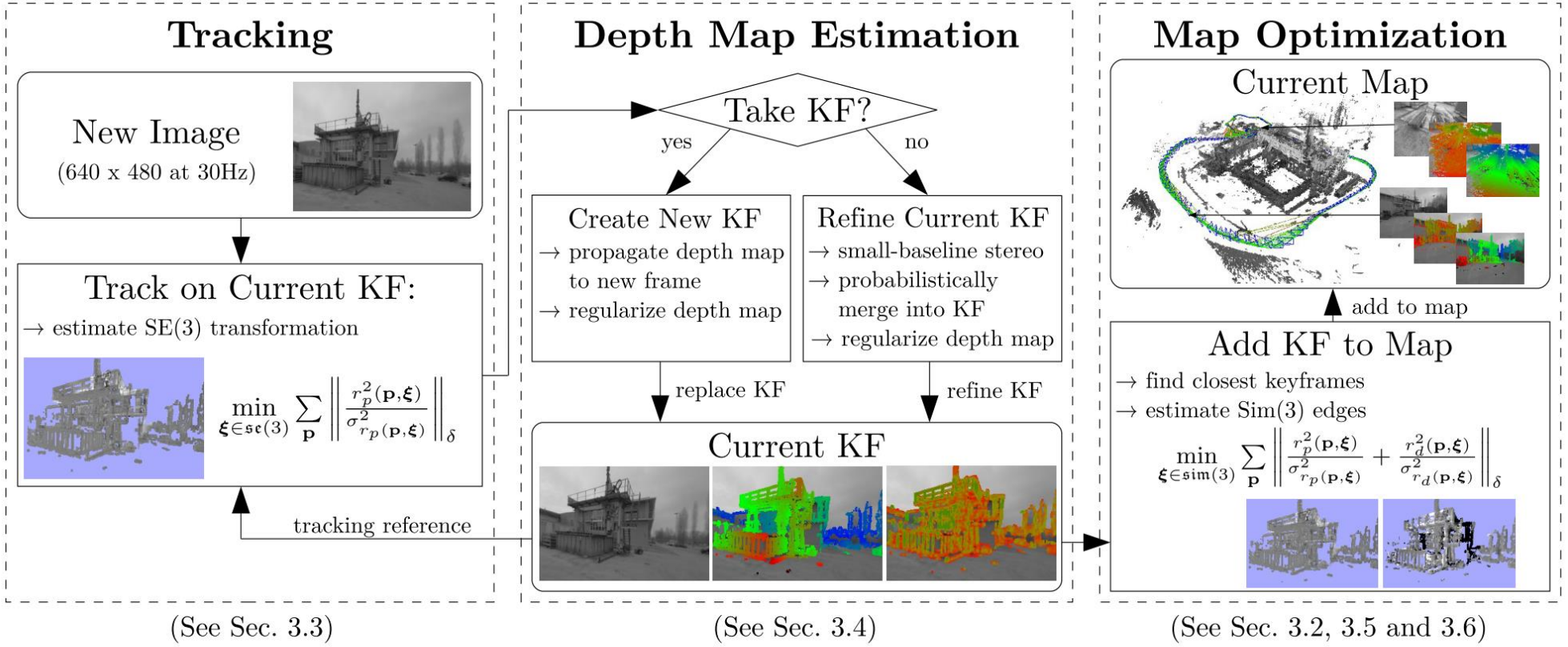


Fig. 3: Overview over the complete LSD-SLAM algorithm.

Engel, Jakob, Thomas Schöps, and Daniel Cremers. "LSD-SLAM: Large-scale direct monocular SLAM." In European Conference on Computer Vision, pp. 834-849. Springer, Cham, 2014.

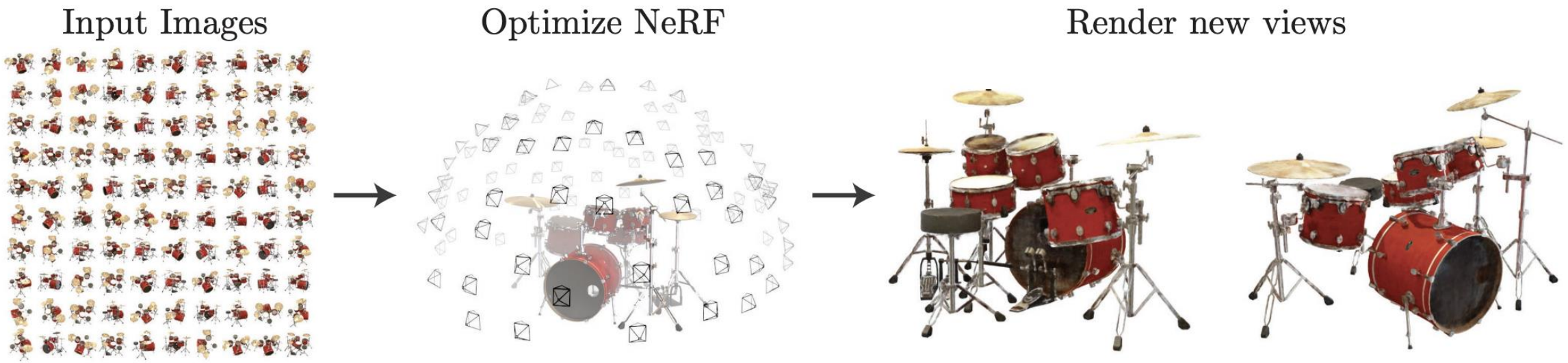
Image from Engel et al. 2014

How to Learn More about SLAM/vSLAM Deeply

- Read papers
  - ICRA/IROS/CVPR/ICCV/3DV
  - Tutorial slides
- Read codes
  - https://github.com/tzutalin/awesome-visual-slam
  - ORB-SLAM3
- Run other's codes
  - On their datasets
  - On your own dataset/cameras
- Can you improve them?

Neural Radiance Fields (NeRF)

What is a Neural Radiance Field for?



Collection of 2D Monocular Images + Camera Extrinsics & Intrinsics → Virtual 3D Object (Novel View Synthesis)

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

Neural Radiance Fields (NeRF)

What is a Neural Radiance Field for?

NeRF-SLAM
Real-Time Dense Monocular SLAM with Neural Radiance Fields

Google's Project Starline
For virtual 3D video conferencing

https://www.youtube.com/watch?v=Q13CishCKXY
https://github.com/ToniRV/NeRF-SLAM

Neural Radiance Fields (NeRF)

What is a Neural Radiance Field?

Neural *Radiance Field*

Neural to denote the use of Deep Neural Networks, particularly to learn the scene-specific plenoptic function.

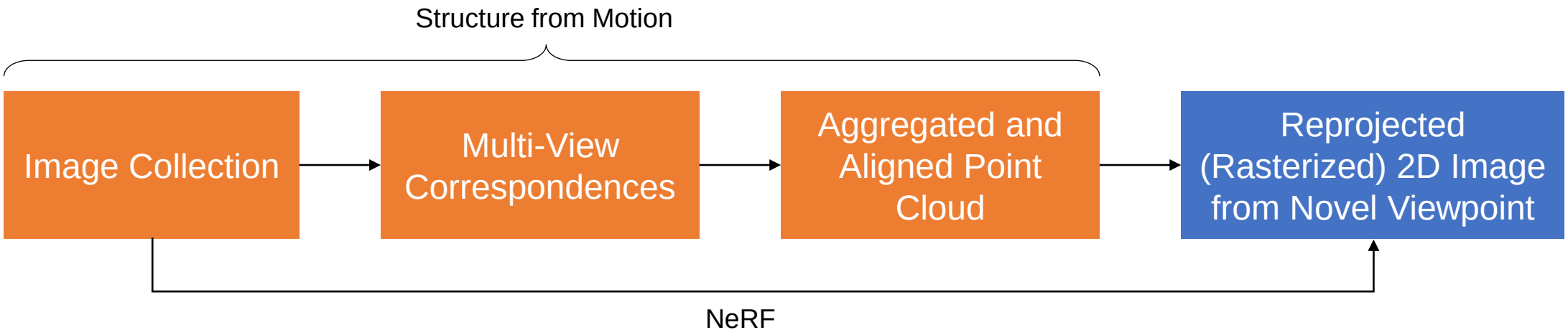
This could also be much more compact and efficient compared to storing explicit light-field information.

Corresponds to **Light Fields**: a function that describes how light traces in *all spatial points* from all *viewing direction* – also known the **plenoptic** function.

First described by Andrey Gershun in 1936, the plenoptic function is parameterized by 5 variables – **[x, y, z]** for spatial coordinates and **[θ, ϕ]** for the viewing angles.

Neural Radiance Fields (NeRF)

What is the difference with SfM?



Structure from Motion

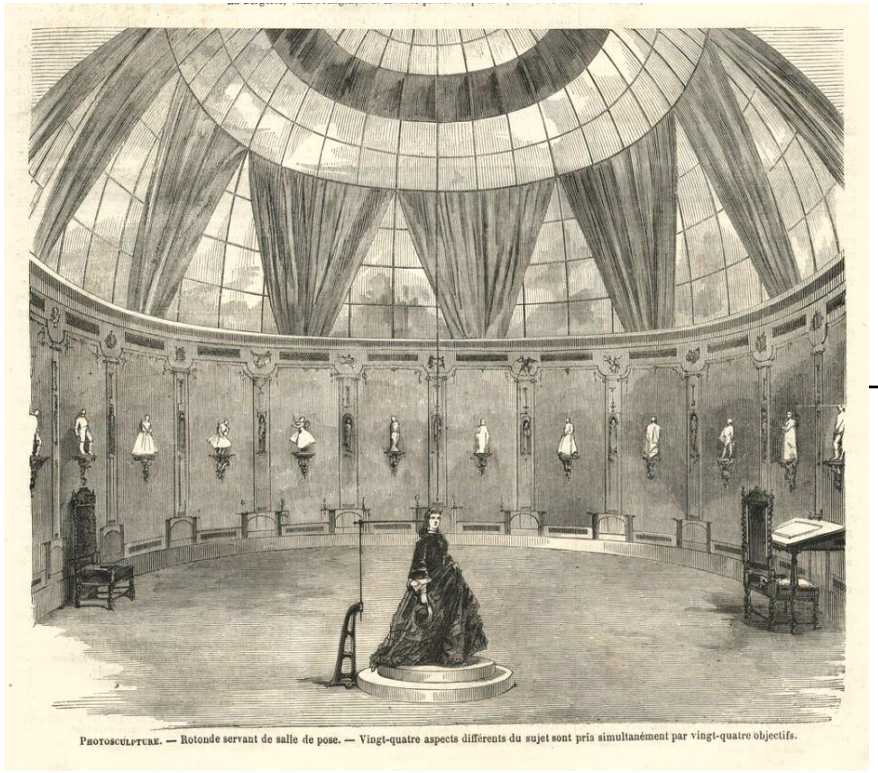
- Estimates camera parameters and multi-view correspondences.
- Reconstruct point cloud from keypoints extracted from respective images.
- Requires rasterization in order to create 2D image from novel viewpoint.
- Able to create 3D mesh via Surface Reconstruction techniques.

Neural Radiance Fields

- Does not estimate camera parameters – requires them as input. We could derive these inputs from SfM methods (COLMAP!)
- Does not reconstruct point cloud but learns an appropriate plenotic function instead.
- Directly predicts pixel value of 2D image from a novel viewpoint.
- Able to create 3D mesh via Surface Reconstruction techniques.

A Brief Historical Review

Methods with similar idea dates back to the late 1800s in France. For example, photosculpting pioneered by François Willème.



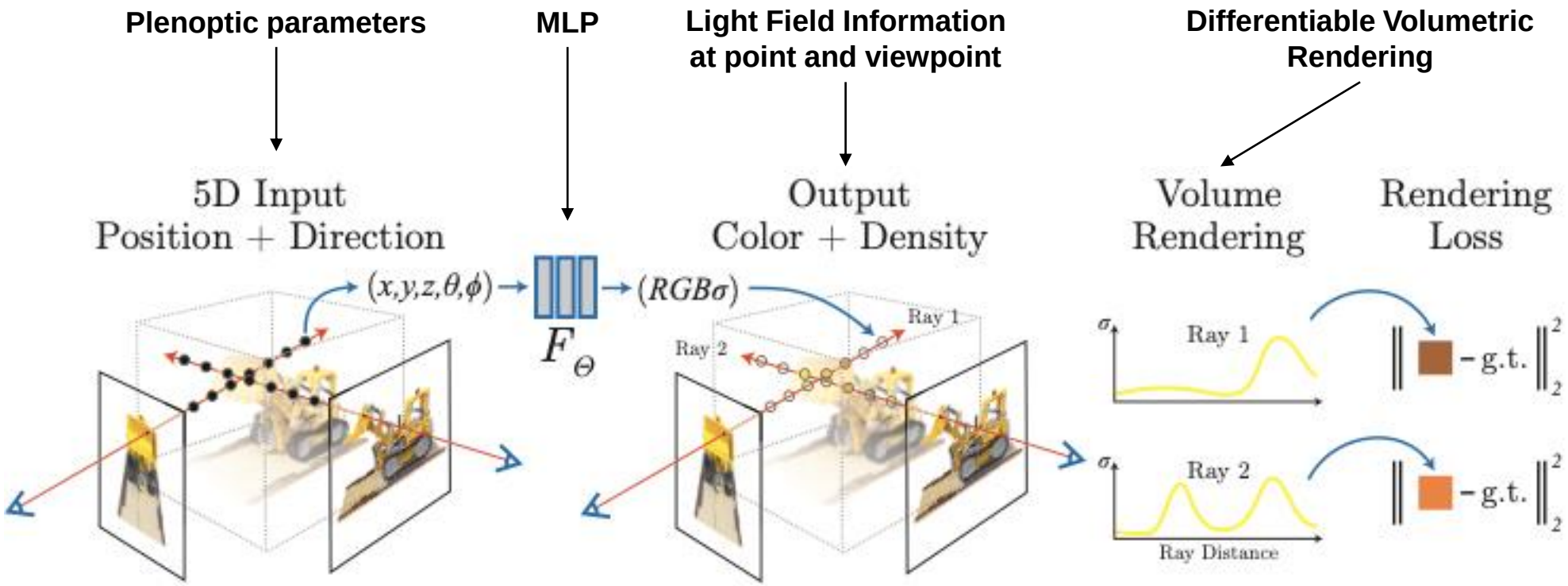
Willème's studio with 24 cameras at 15° interval surrounding a subject for capture.

An assembled wooden figure made by tracing across the images with a pantograph.

Final sculpture made by casting.

Images from The Patrick Montgomery Collection

Neural Radiance Fields (NeRF)



Key Idea: Use an MLP to learn the plenoptic function to completely characterize the subject, so we can perform novel view synthesis.

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

Neural Radiance Fields (NeRF)

Challenge: We can approximate the plenotic function via an MLP, but how can we make it learnable with gradient methods?

A differentiable volumetric function!

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \quad \text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right) \quad , \quad \mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

- $C(\mathbf{r})$: **Color Value of ray *r***
- t_n, t_f: **The *near* and *far* bound of ray tracing**
- $\mathbf{c}(\mathbf{r}(t), \mathbf{d})$: **Color Value along ray *r* at point *t***
- $T(t)$: **Accumulated Transmittance along ray from *near* to *far* bounds**
- $\sigma(\mathbf{r}(s))$: **Volume Density at *r***
  Function of ray's 3D spatial coordinate only, no view angles
- $\mathbf{r}(t)$: **Camera Ray *r***
- t: **Sampling Interval**
- $\mathbf{o}$: **Ray Origin**
- $\mathbf{d}$: **Ray Direction**

But the above is expensive to compute! Can we do better?

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

Neural Radiance Fields (NeRF)

Yes! Discretize the ray and compute the integral numerically via Monte Carlo Quadrature (Stratified Sampling)

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

$$\downarrow$$

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i\delta_i))\mathbf{c}_i\,, \text{ where } T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j\delta_j\right) \quad \text{and}$$

$$t_i \sim \mathcal{U}\left[t_n + \frac{i-1}{N}(t_f - t_n),\ t_n + \frac{i}{N}(t_f - t_n)\right] \quad \text{, discretized random sampling of points along ray}$$

$$\delta_i = t_{i+1} - t_i$$

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

Neural Radiance Fields (NeRF)

With some additional tricks

1. Positional encoding of inputs for better high-frequency details.
2. Hierarchical volume sampling for better efficiency.
3. Related to (2), a two-stage, course and fine model refinement schema.

The final loss function is simply the sum of squared L_2 norm of the color value for both the course and fine model – dubbed the photometric loss for brevity.

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

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

Tricks in Working with NeRF

Here are some tricks and know-hows to help you get started with NeRF

1. If the dataset you're working with does not come with ground truth camera parameters (intrinsic and extrinsic), you could leverage SfM tools such as COLMAP to generate them.

2. Understand the coordinate system convention of your dataset first before you begin! There are various conventions out there, and these conventions could vary across datasets and NeRF variants. For e.g.
    1. NeRF – Uses the Blender format where +Z is towards the camera, and more.
    2. Instant-NGP – Uses the OpenCV format where +Z is away from the camera, and more.
    3. The above are the two most popular conventions you'll encounter - be good to know them.

3. On capturing images for NeRF training
    1. You should capture images of the subject from as many angles, as densely as possible as a rule of them.
    2. There shouldn't be any moving elements from frame to frame, otherwise your model will not converge – Instant-NGP provides some built-in tools to blot out moving humans in your training images for this reason.
        1. If you would like to capture scenes with moving elements, consider Dynamic Radiance Fields instead.
    3. Also make sure to eliminate any blurring or glares as this would adversely affect the training.

4. For prototyping, it's usually good to begin with variants that are quick to train so that you can iterate fast! (e.g. Instant-NGP, TensoRF) Though note that some methods, though often discussed together with NeRF loosely, are not exactly NeRF due to their methods (e.g. Plenoxels, though performs the same task, is not Neural nor implicit).

Tricks in Working with NeRF

Here are some tricks and know-hows to help you get started with NeRF

5. If you're using methods such as Instant-NGP, the model should start converging in only a few epochs (takes seconds). Therefore, you can use this for sanity check – if your model does not start converging after a few epochs, you probably have some issues upstream (e.g. with your dataset; check your coordinate convention).

6. Another way to sanity check your dataset is to either
    1. Utilize the GUI tool to check the camera poses (if available for the variant you're using), OR
    2. Visualize the camera poses manually by plotting them with Python or Blender (convenient if you're working with the Blender convention).

7. NeRF typically utilizes Marching Cube for surface reconstruction to generate a mesh. However, remember that NeRF sits at a level below surface reconstruction, therefore you could technically write an integrator to integrate NeRF with any 3rd party surface reconstruction algorithms you want.

Advances in Neural Radiance Fields

There are many many new work being published on NeRF in recent years. Here are some key ones.

1. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding (Instant-NGP)
    1. Much faster train (minutes) and inference time (real-time) via multiresolution grid and hash encoding.
    2. https://github.com/NVlabs/instant-ngp

2. Tensorial Radiance Fields (TensoRF)
    1. Similarly, enables much faster train and inference time than NeRF via matrix decompositions.
    2. https://apchenstu.github.io/TensoRF/

3. Zip-NeRF: Anti-Aliased Grid-Based Neural Radiance Fields
    1. Combines ideas from mip-NeRF 360 (fast to train and lowers aliasing artifacts) + Instant-NGP to provide real-time high-quality rendering of complex scenes (useful for indoor SLAM)
    2. https://arxiv.org/pdf/2210.00379.pdf

4. Tri-MipRF: Tri-Mip Representation for Efficient Anti-Aliasing Neural Radiance Fields
    1. Combines ideas from TensoRF + mip-NeRF 360, enabling even faster train and render time with better reconstruction quality.
    2. https://wbhu.github.io/projects/Tri-MipRF/

Advances in Neural Radiance Fields

There are many many new work being published on NeRF in recent years. Here are some key ones.

5. RoDynRF: Robust Dynamic Radiance Fields
    1. Fundamental evolution of NeRF – now it tracks and reconstruct across time as well, hence the name, Dynamic Radiance Fields; recreate dynamic 3D scenes with moving subjects.
    2. https://robust-dynrf.github.io

6. **NeRF: Neural Radiance Field in 3D Vision, Introduction and Review**
    1. A very comprehensive review paper on NeRF from this year.
    2. https://arxiv.org/pdf/2210.00379.pdf

Additional Applications with NeRF

1. Real-Time Dense Monocular SLAM with Neural Radiance Fields (NeRF-SLAM)
    1. Combines dense SLAM + NeRF to perform real-time SLAM with reconstruction on monocular images only.
    2. https://github.com/ToniRV/NeRF-SLAM

Next Week

++ Decision boundary and metrics

* Dimensionality reduction with PCA

++ Building classifier

- Supervised
  - + LDA
  - ++ KNN
  - + SVM
- Unsupervised
  - ++ K-means

+ Deep learning based image classification

- + Advanced CNNs
- + Vision Transformers (ViT)

References for Next Week

- SZ: Ch 5.1, 5.2, 5.3, 5.4, 7.1.3,
- Abdi, Hervé, and Lynne J. Williams. “Principal component analysis.” Wiley interdisciplinary reviews: computational statistics 2.4 (2010): 433-459.
- Turk, Matthew, and Alex Pentland. “Eigenfaces for recognition.” Journal of cognitive neuroscience 3.1 (1991): 71-86.
- Belhumeur, Peter N., Joao P. Hespanha, and David J. Kriegman. “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection.” IEEE Transactions on pattern analysis and machine intelligence 19.7 (1997): 711-720.
- Howard, Andrew G., et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications.” arXiv preprint arXiv:1704.04861 (2017).
- Dosovitskiy, Alexey, et al. “An image is worth 16x16 words: Transformers for image recognition at scale.” arXiv preprint arXiv:2010.11929 (2020).
- Han, Kai, et al. “A survey on vision transformer.” IEEE transactions on pattern analysis and machine intelligence (2022).