SLAM I: The problem of SLAM

Autonomous Mobile Robots

Margarita Chli
Martin Rufli, Roland Siegwart

SLAM | today's lecture

Section **5.8** + some extras…

- SLAM: what is it?

- Approaches to SLAM:
  - Bundle Adjustment
  - Filtering (UKF/EKF/Particle Filter SLAM)
  - Keyframes

- EKF SLAM in detail
- EKF SLAM case study: MonoSLAM
- Components for a scalable SLAM system

SLAM | Simultaneous Localization And Mapping

The SLAM problem:

How can a body **navigate** in a previously unknown environment while constantly building and updating a **map** of its workspace using onboard sensors & onboard computation only?

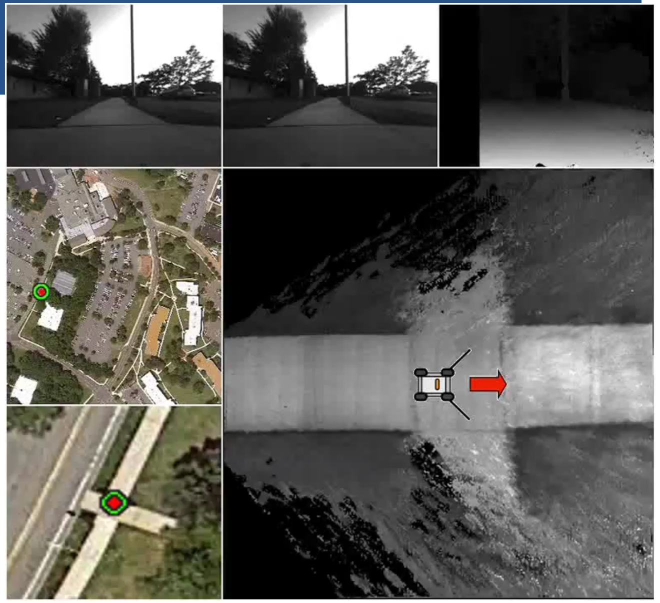
- When is SLAM necessary?
  - When a robot must be truly **autonomous** (no human input)
  - When there is **no prior** knowledge about the environment
  - When we cannot place **beacons** and cannot use **external positioning systems** (e.g. GPS)
  - When the robot needs to know where it is

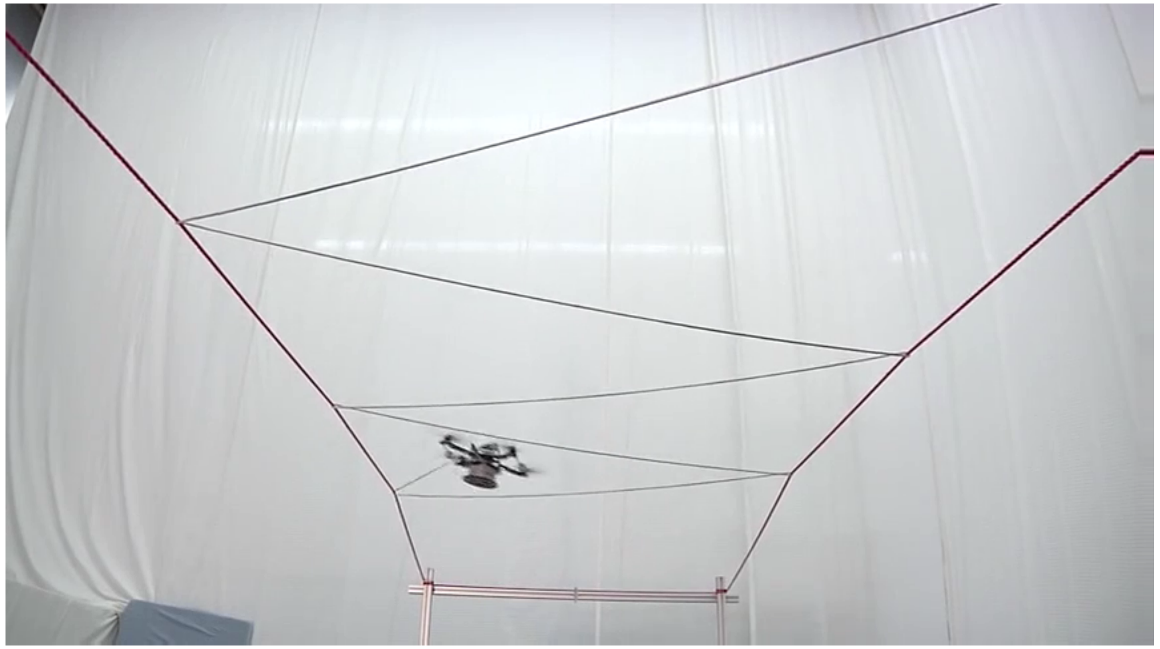
SLAM | **Simultaneous** Localization **And** Mapping

- **The backbone of spatial awareness of a robot**
- One of the most challenging problems in probabilistic robotics
- An unbiased map is necessary for localizing the robot
  Pure localization with a known map.

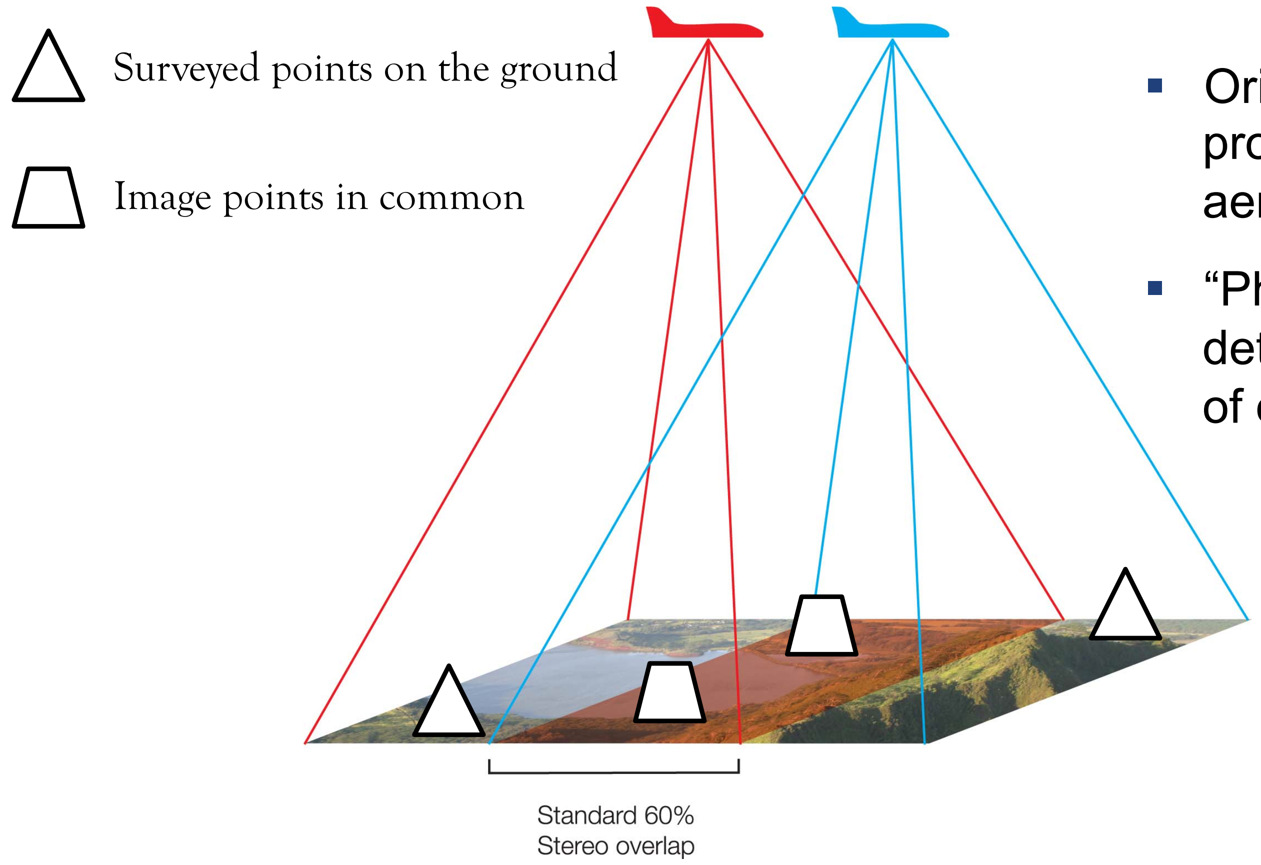
  SLAM: no a priori knowledge of the robot's workspace
- An accurate pose estimate is necessary for building a **map** of the environment
  Mapping with known robot poses.

  SLAM: the robot poses have to be estimated along the way


Robot localization using Satellite images
[Senlet and Elgammal, ICRA 2012]


Helicopter position given by Vicon tracker
ETH Zurich Flying Machine Arena, IDCS & NCCR DF, 2013

SLAM | a short history of photogrammetry



Surveying for Mapping---part 1, http://www.icsm.gov.au/mapping/surveying1.html

- Originated from efforts to formalize production of topographic maps from aerial imagery
- "Photogrammetry" – the practice of determining the geometric properties of objects from images

SLAM | a short history of photogrammetry

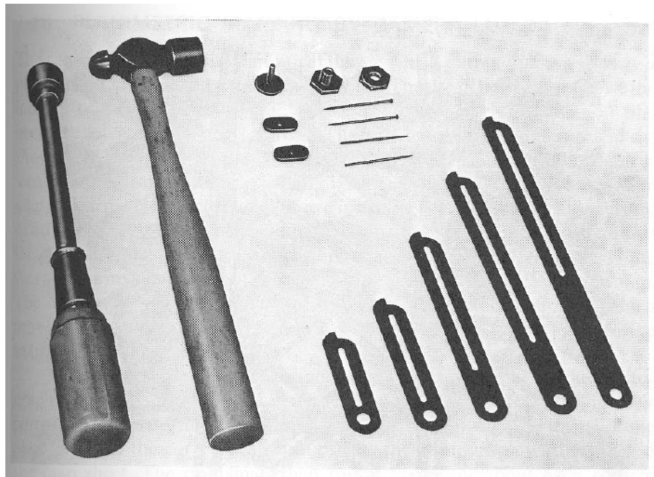
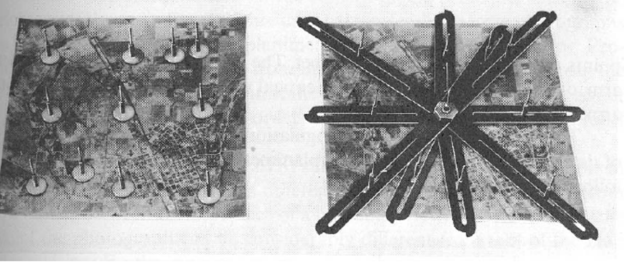
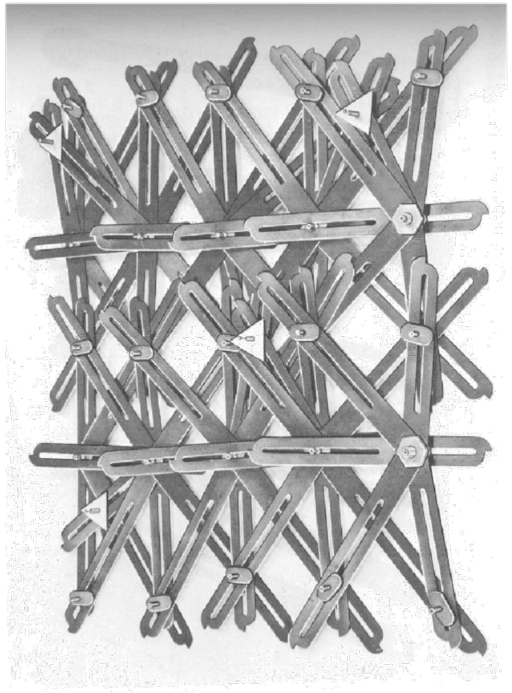


Figure 7-21. Slotted metal arms.





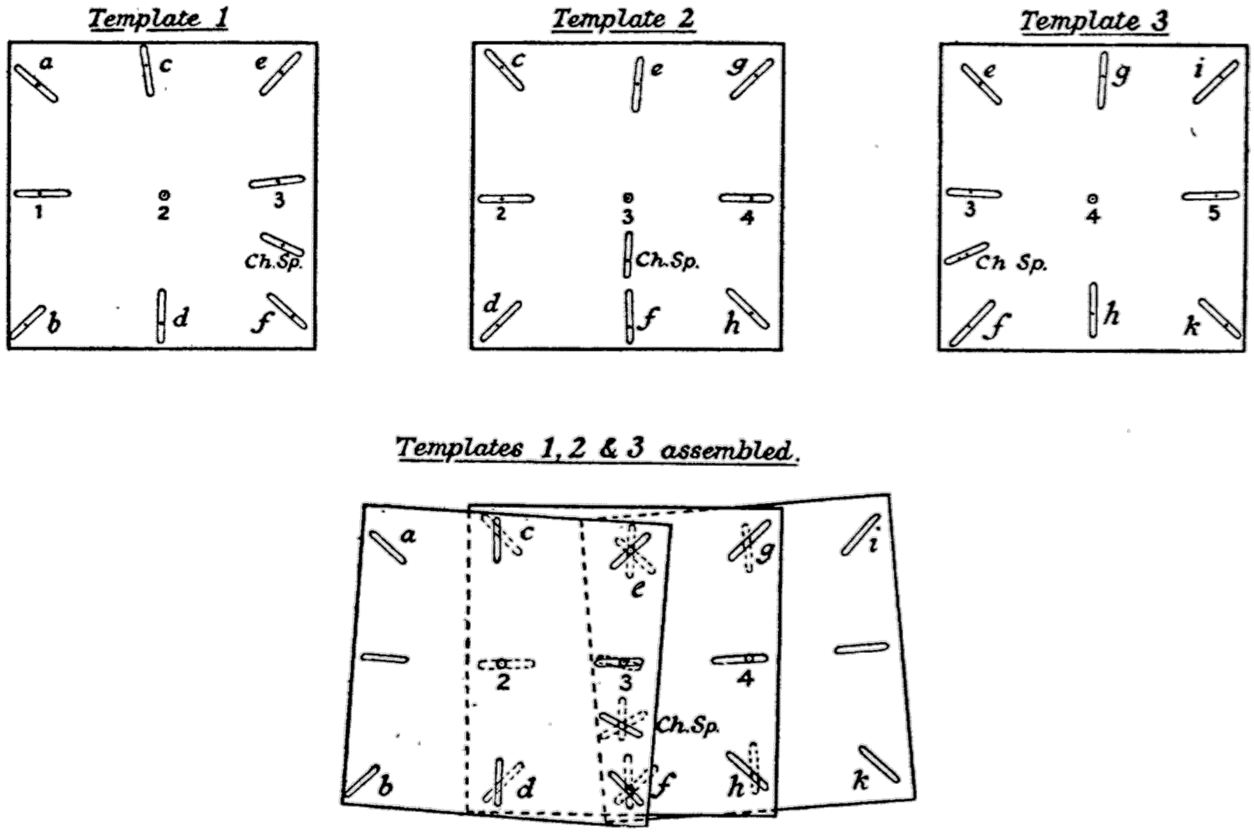
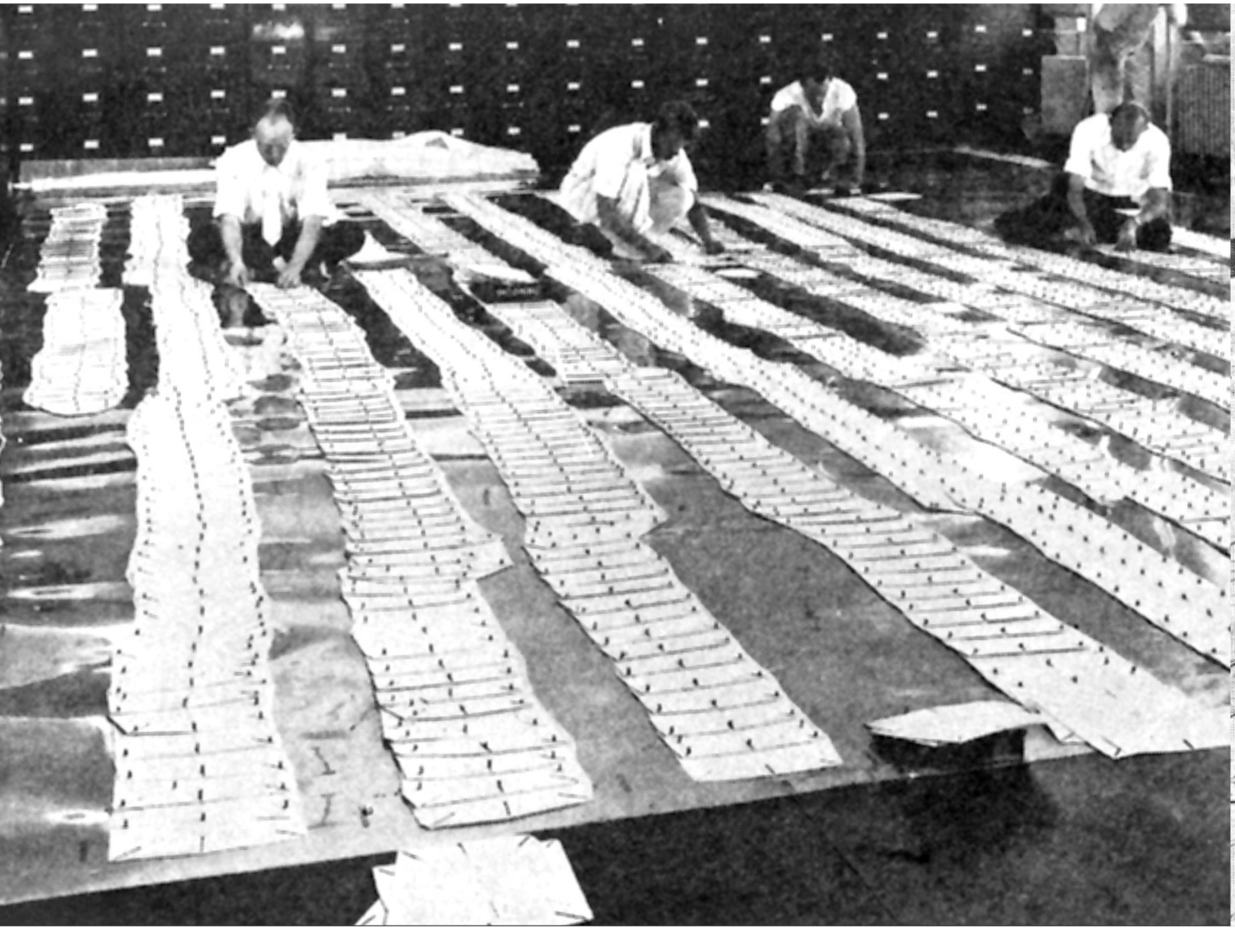
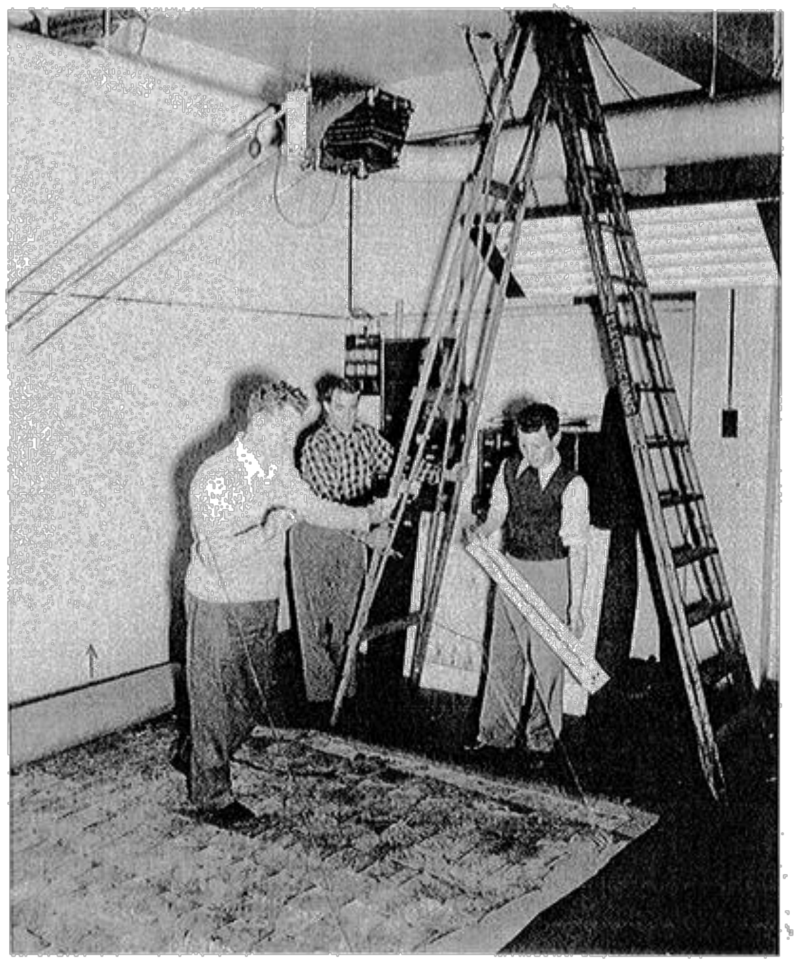


Fig. 9. Slotted template method of assembling air photographs

SLAM | a short history of photogrammetry



SLAM | a short history of photogrammetry



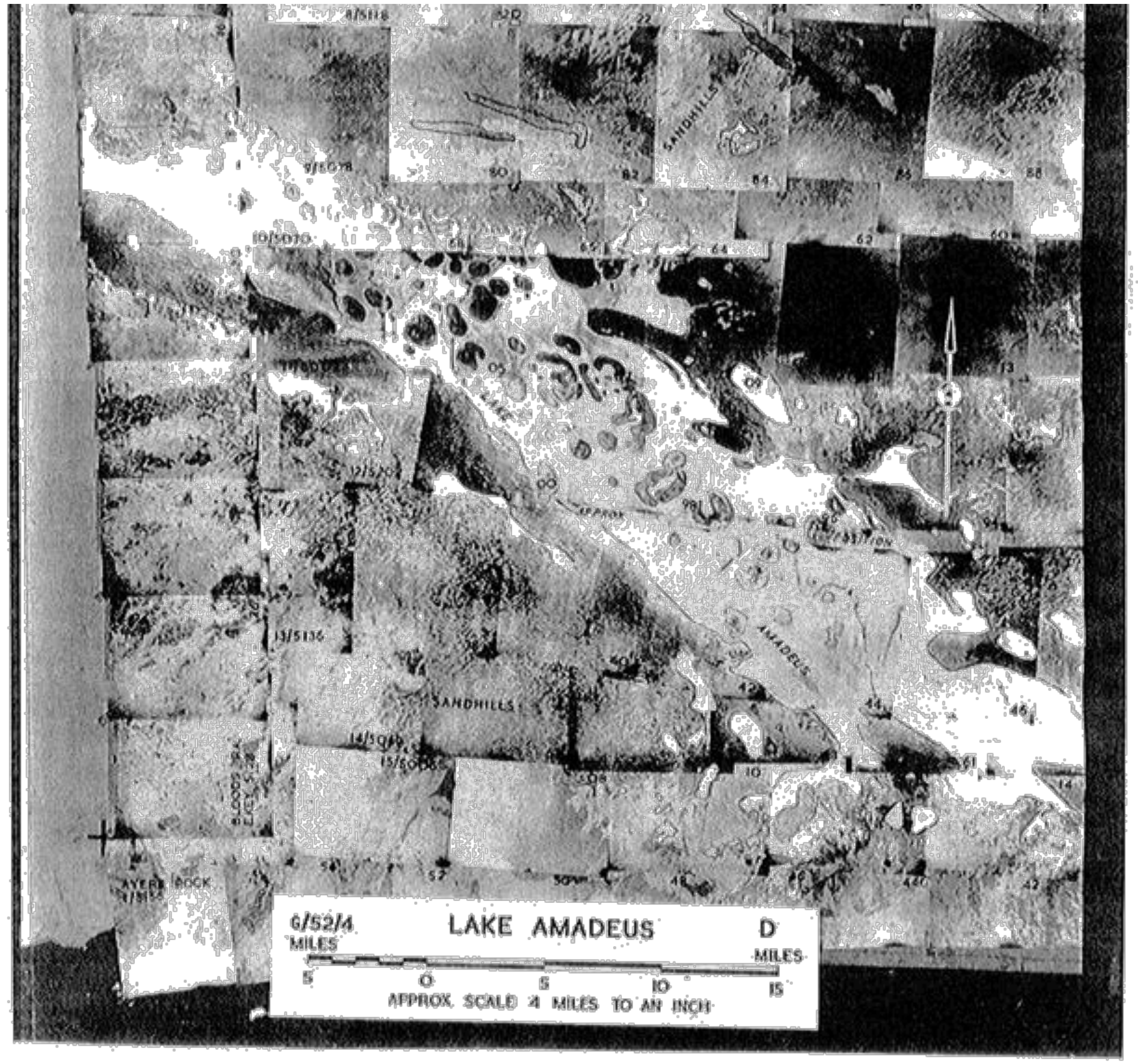
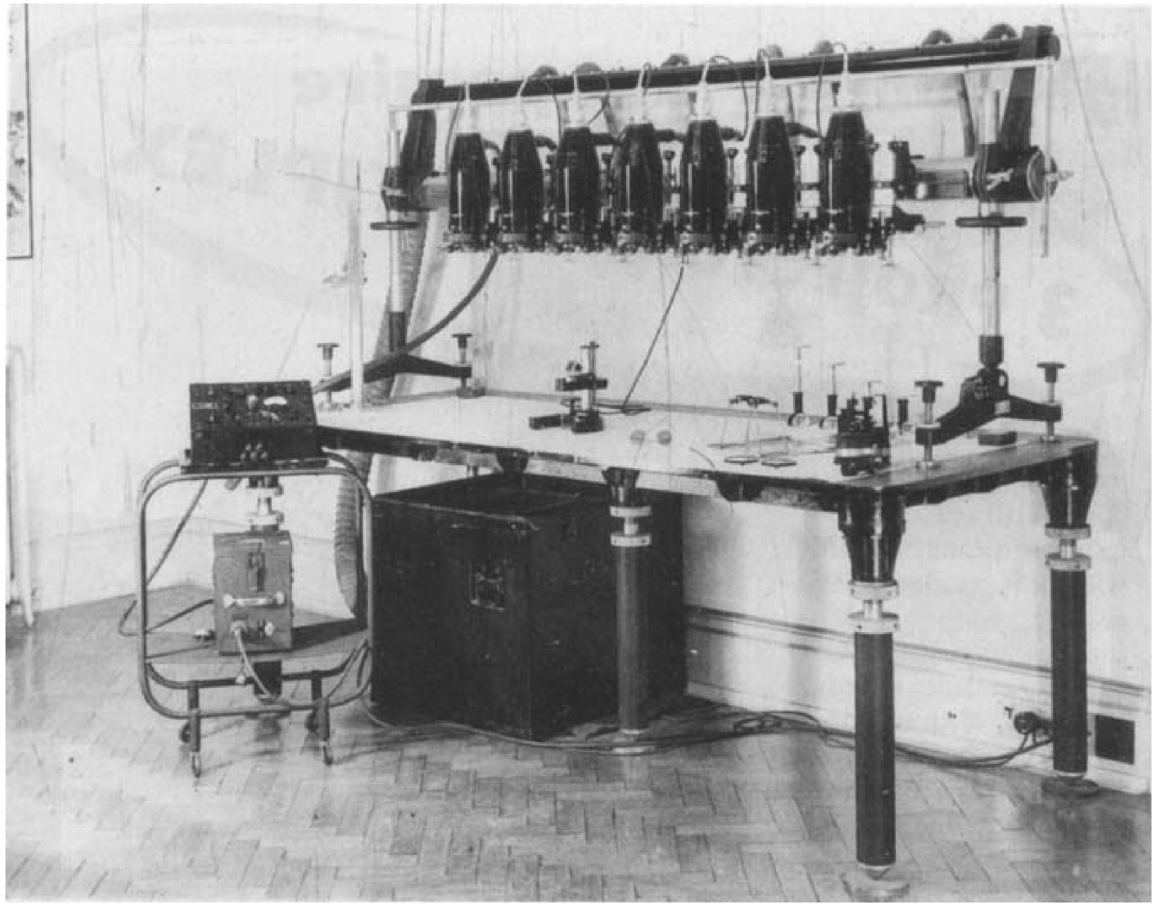


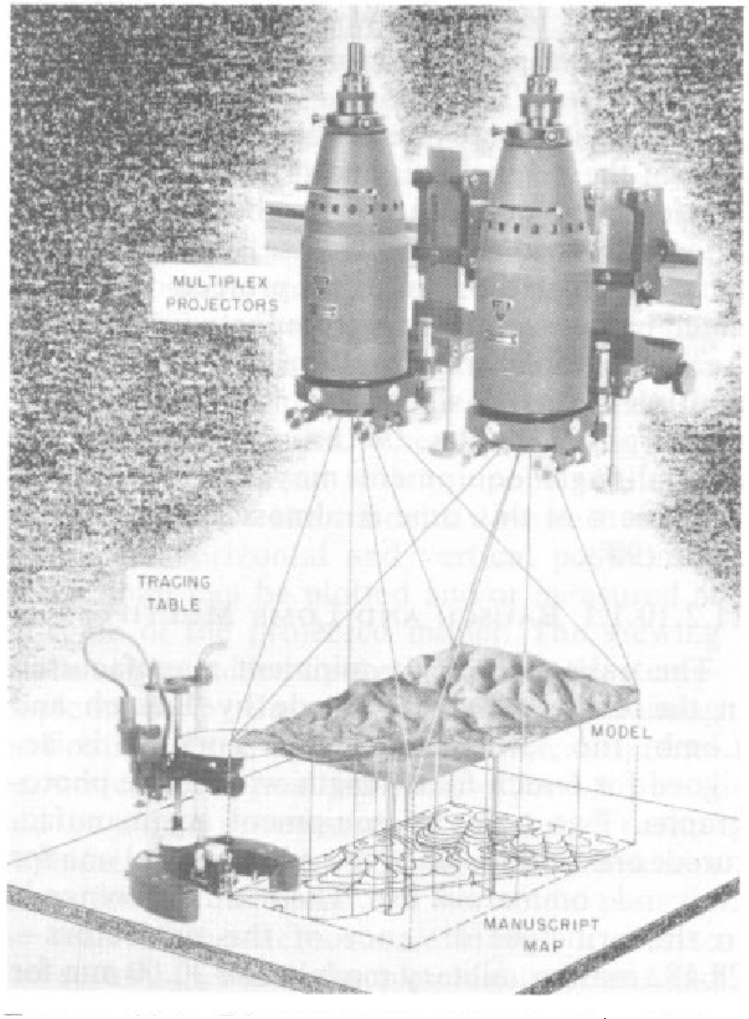
Fig. 6. Overhead camera set up to photograph a section of a 'four mile' mosaic. Fig. 7. One of six sections of SG-52-04 Lake Amadeus (Amadeus 5048).

NATMAP EARLY DAYS, MAP COMPILATION FROM AERIAL PHOTOGRAPHS 1948 -1970S David R. Hocking

SLAM | a short history of photogrammetry

1940s: Opto-mechanical systems: aerial images set on glass plates, arranged in a series of projectors





C. Burnside. The photogrammetric society analogue instrument project: a second extract. The Photogrammetric Reco
14(83):769–782, 1994.

SLAM | a short history of photogrammetry

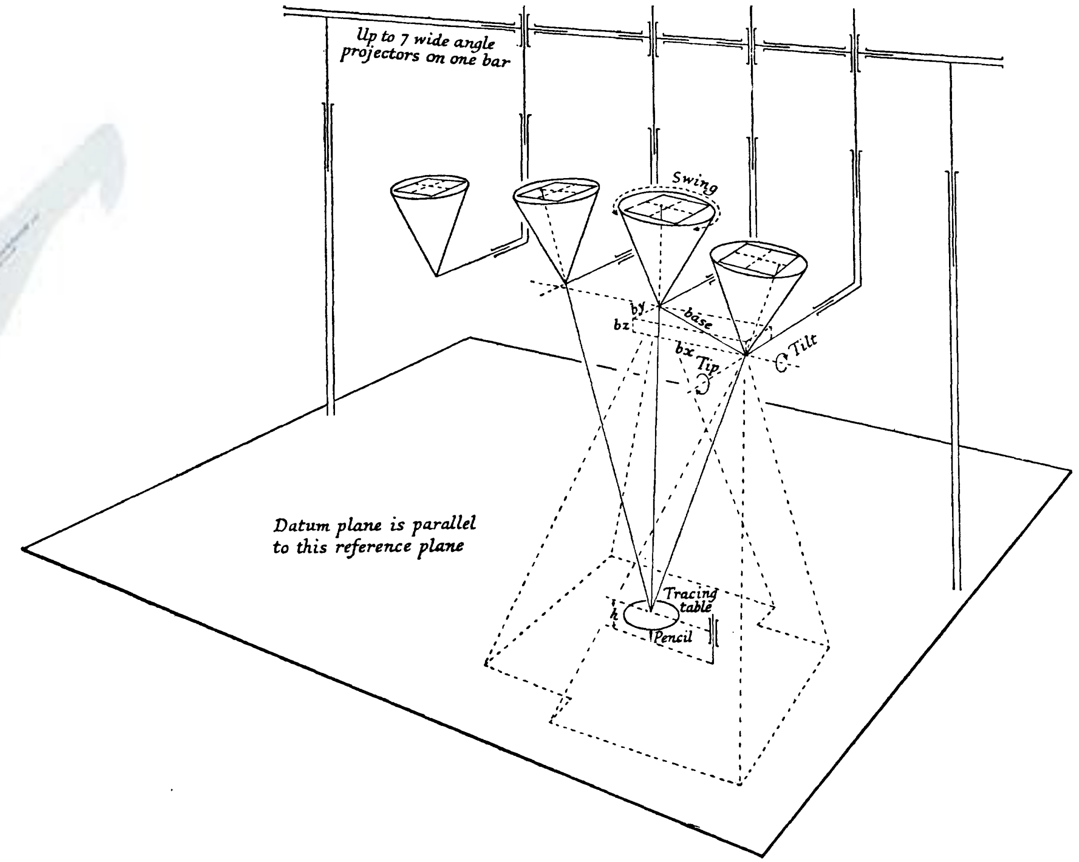
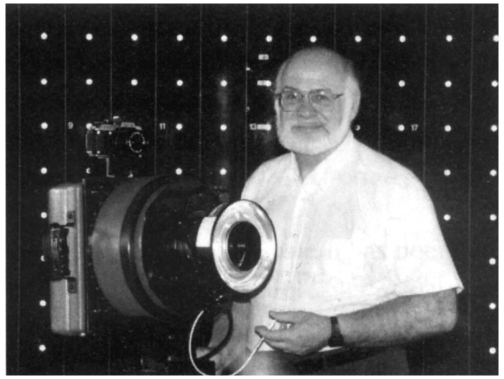

Up to 7 wide angle projectors on one bar
Swing
by
bz
base
bx
Tip
Tilt
Datum plane is parallel to this reference plane
Tracing table
h
Pencil

Fig. 10. Theory of Multiplex

C. Hart. Air survey: The modern aspect. The Geographical Journal, 108(4/6):179–198, 1946

SLAM | a short history of photogrammetry

Bundle Adjustment


D.C. Brown

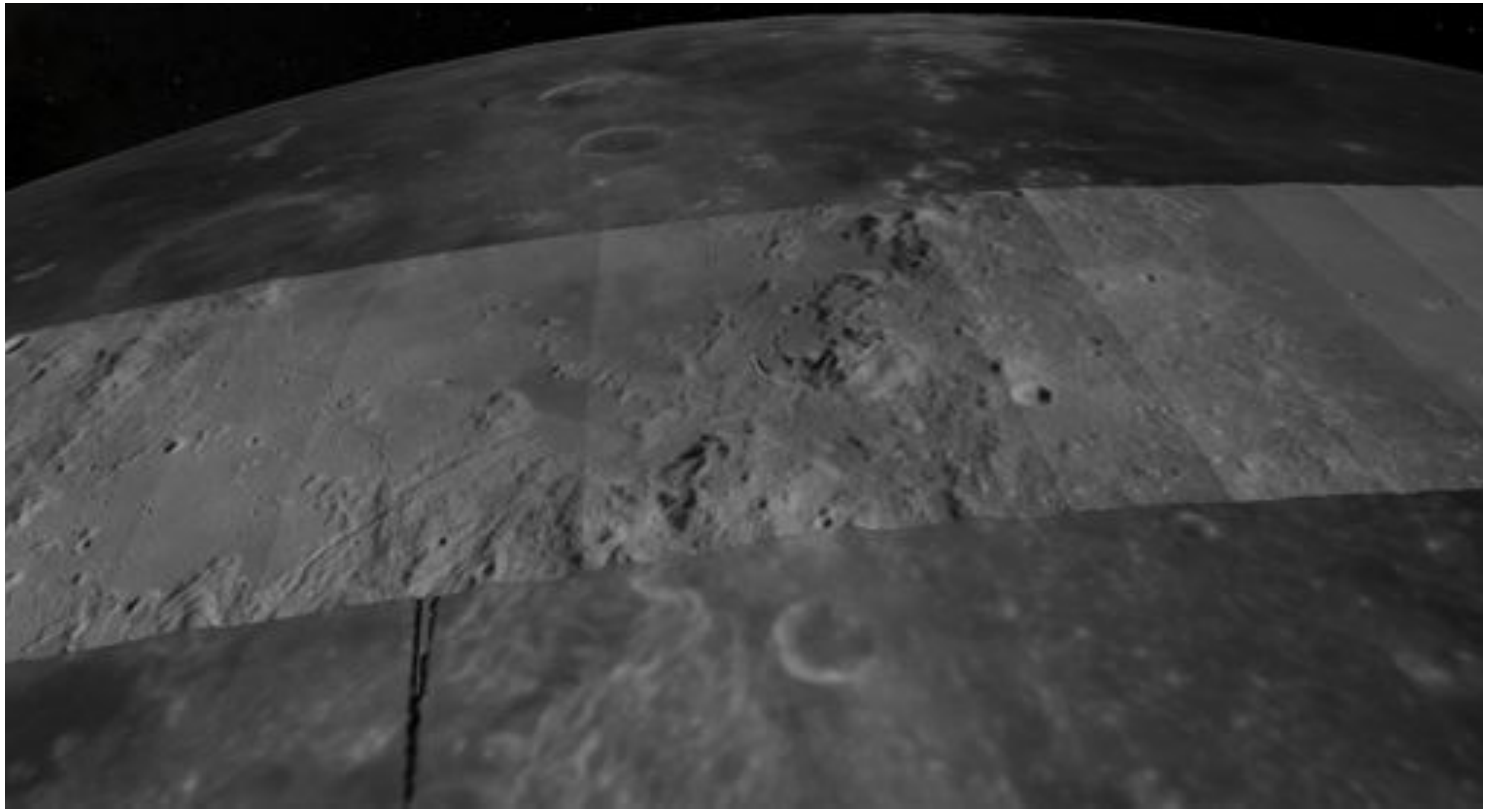
"A rigorous least squares adjustment, believed to be of unprecedented universality, is given for the simultaneous adjustment of the entire set of observations arising from a general m-station photogrammetric net.

...

A computing program for automatic electronic computers is outlined."

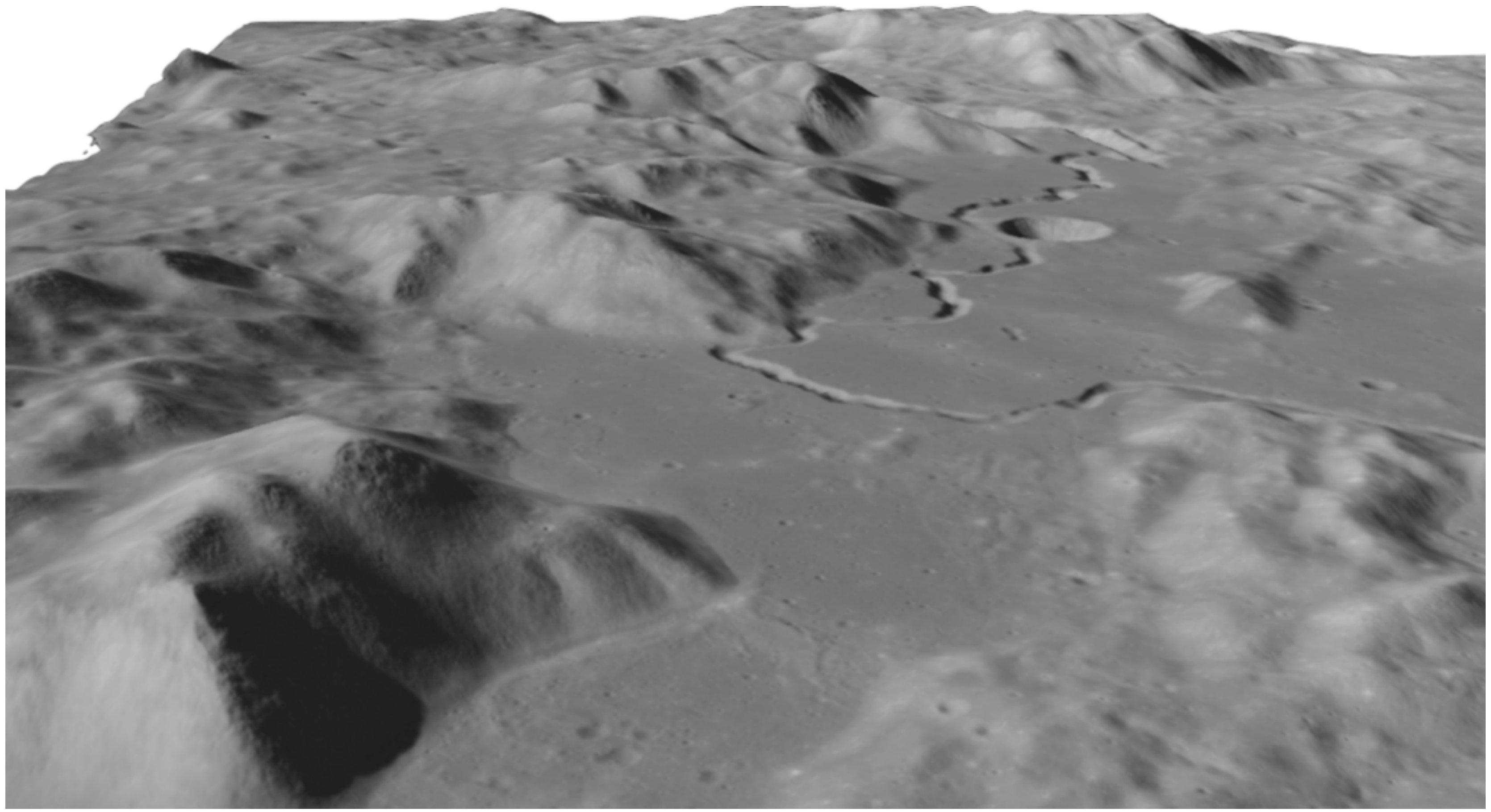
Brown, D.C., A Solution to the General Problem of Multiple Station Analytical Stereo triangulation, RCA Technical Report No. 43, February 1958

SLAM | a short history of photogrammetry



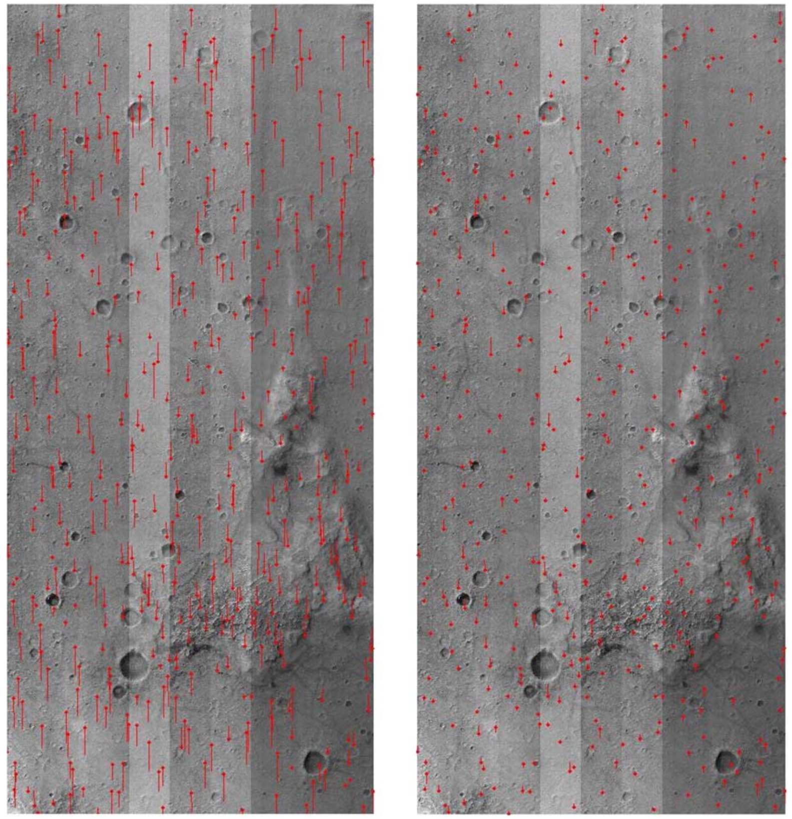
Michael J. Broxton, Ara V. Nefian, Zachary Moratto, Taemin Kim, Michael Lundy, and Aleksandr V. Segal, "3D Lunar Terrain Reconstruction from Apollo Images", *International Symposium on Visual Computing 2009*

SLAM | a short history of photogrammetry



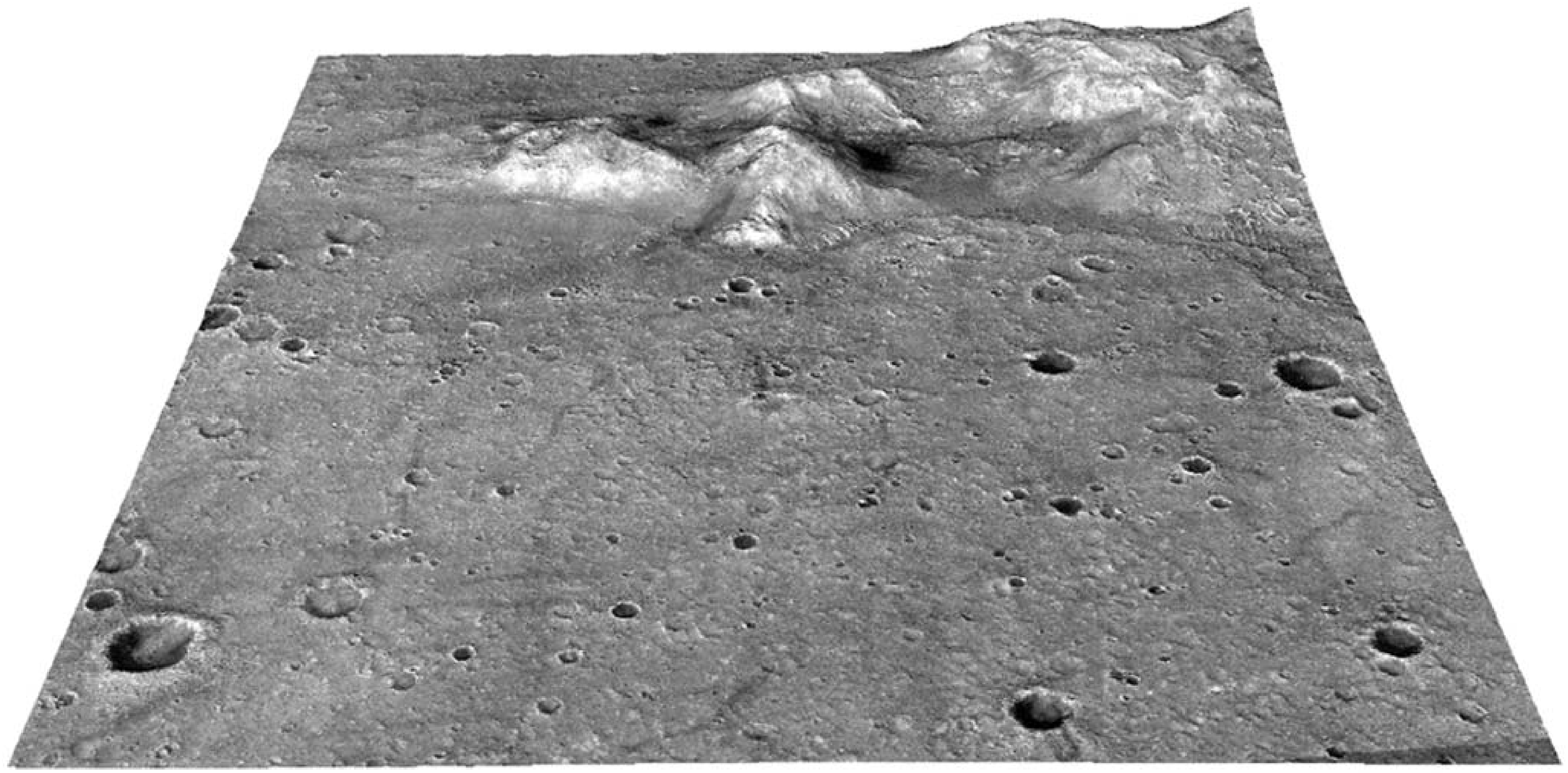
Michael J. Broxton, Ara V. Nefian, Zachary Moratto, Taemin Kim, Michael Lundy, and Aleksandr V. Segal, "3D Lunar Terrain Reconstruction from Apollo Images", *International Symposium on Visual Computing 2009*

SLAM | a short history of photogrammetry



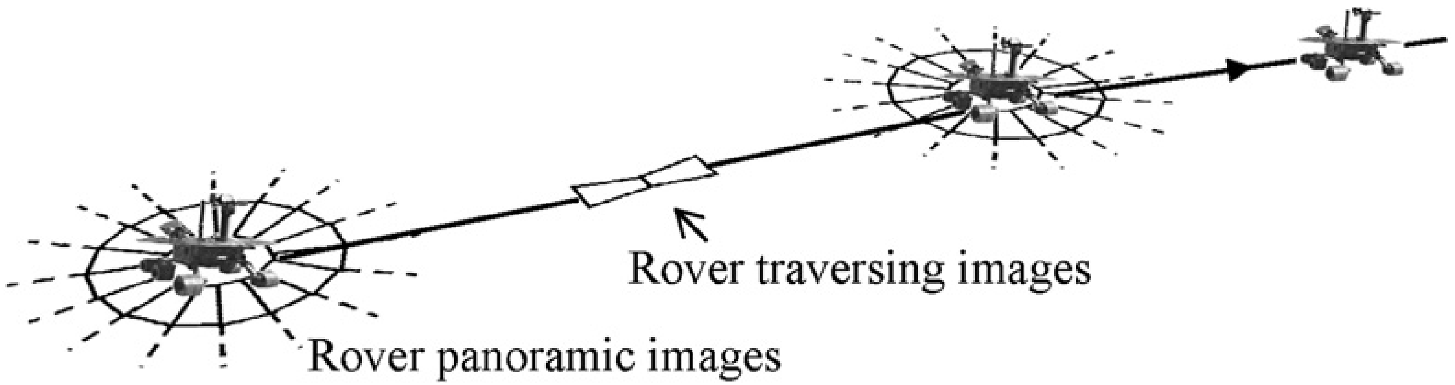
R. Li, J. Hwangbo, Y. Chen, and K. Di. Rigorous photogrammetric processing of hirise stereo imagery for mars topographic mapping. Geoscience and Remote Sensing, IEEE Transactions on, (99):1–15, 2008.

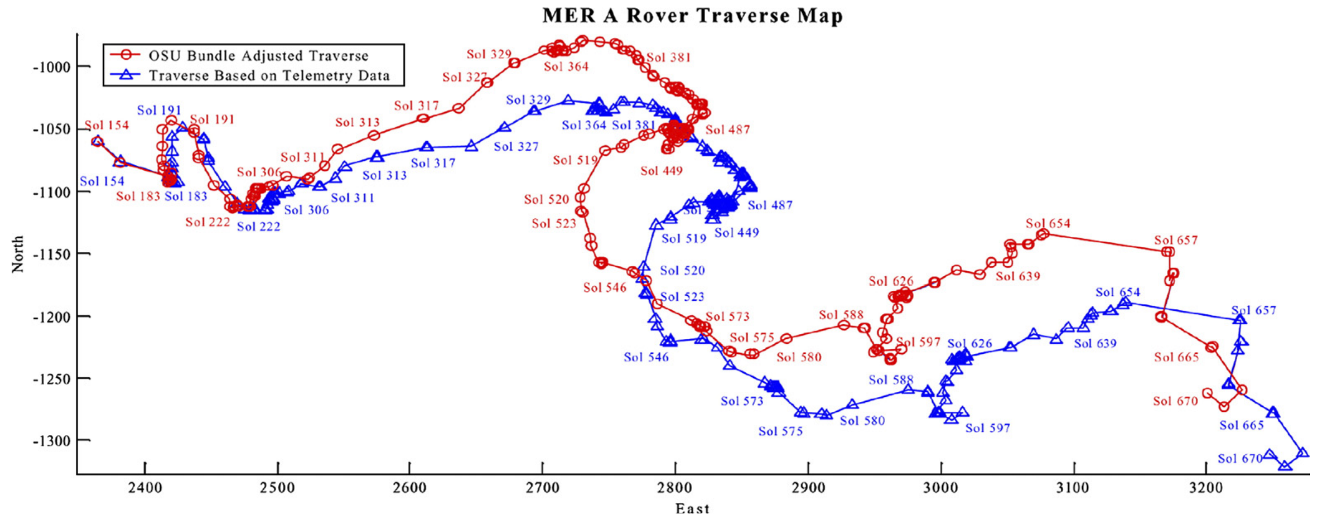
SLAM | a short history of photogrammetry



R. Li, J. Hwangbo, Y. Chen, and K. Di. Rigorous photogrammetric processing of hirise stereo imagery for mars topographic mapping. Geoscience and Remote Sensing, IEEE Transactions on, (99):1–15, 2008.

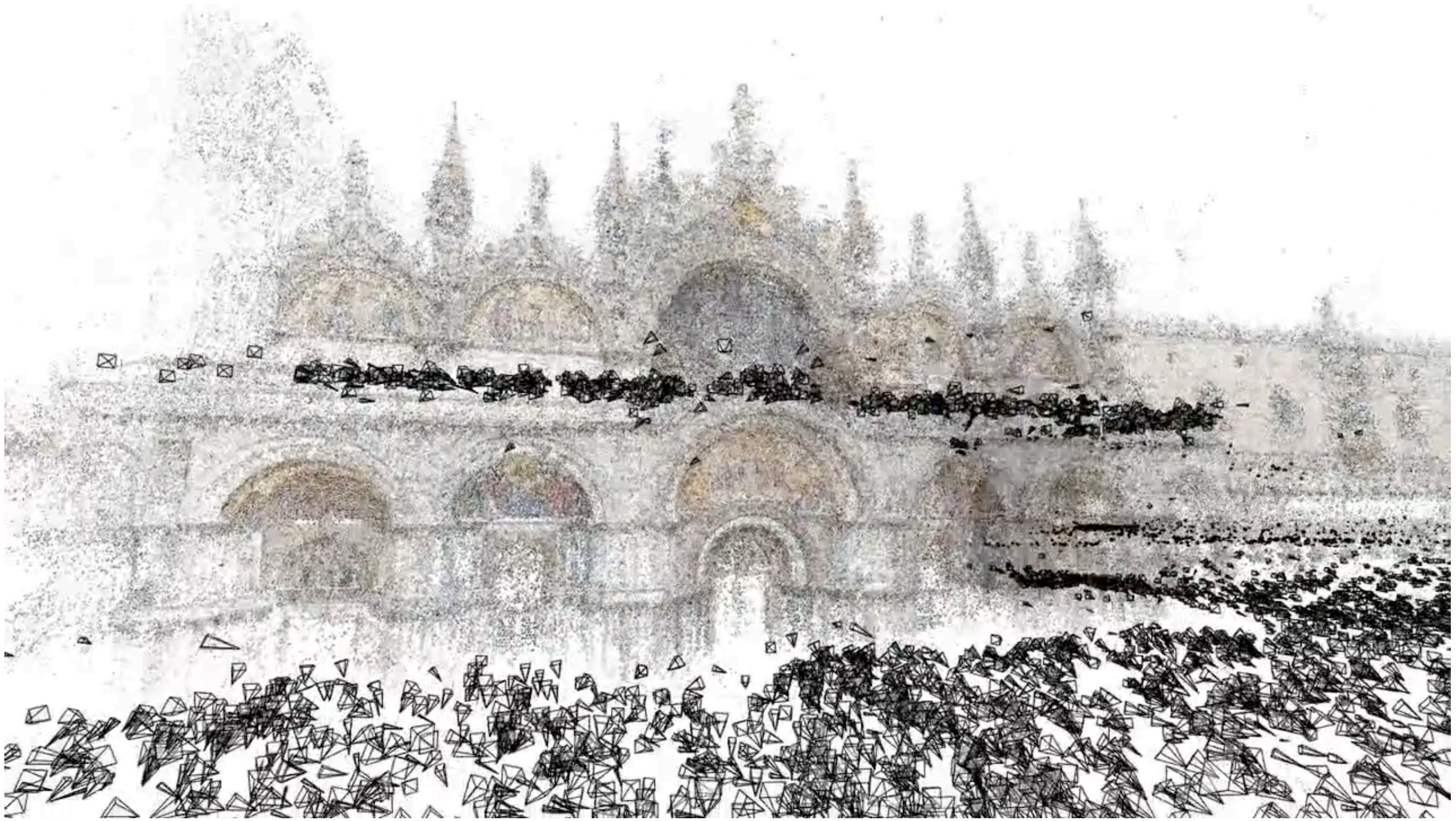
SLAM | a short history of photogrammetry





K. Di, F. Xu, J. Wang, S. Agarwal, E. Brodyagina, R. Li, and L. Matthies. Photogrammetric processing of rover imagery of the 2003 mars exploration rover mission. ISPRS Journal of Photogrammetry and Remote Sensing, 63(2):181–201, 2008.

SLAM | from photogrammetry to SFM

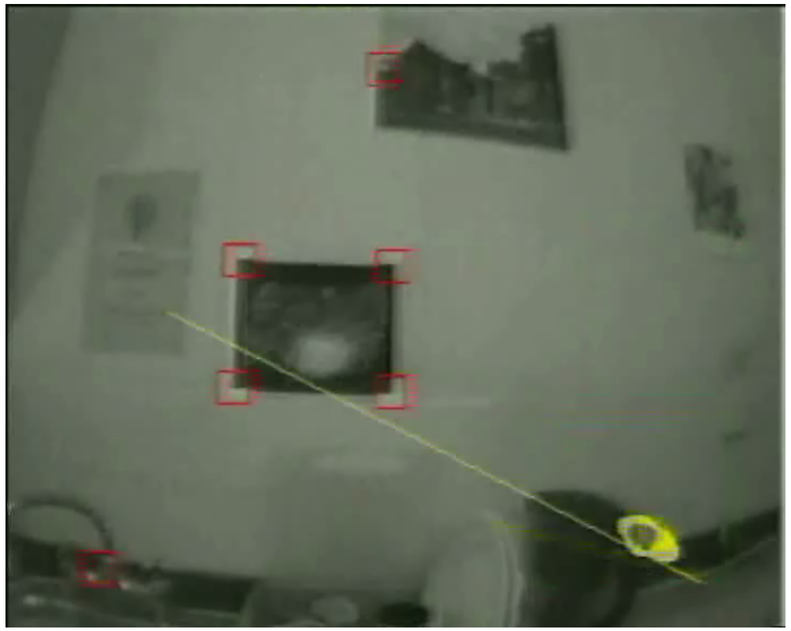


S. Agarwal, Y. Furukawa, N. Snavely, B. Curless, S. M. Seitz and R. Szeliski, “Reconstructing Rome”, IEEE Computer, 2010

SLAM | from photogrammetry to SFM to SLAM

SLAM | perceiving motion w.r.t. scene

- Can we track the motion of a camera/robot while it is moving?



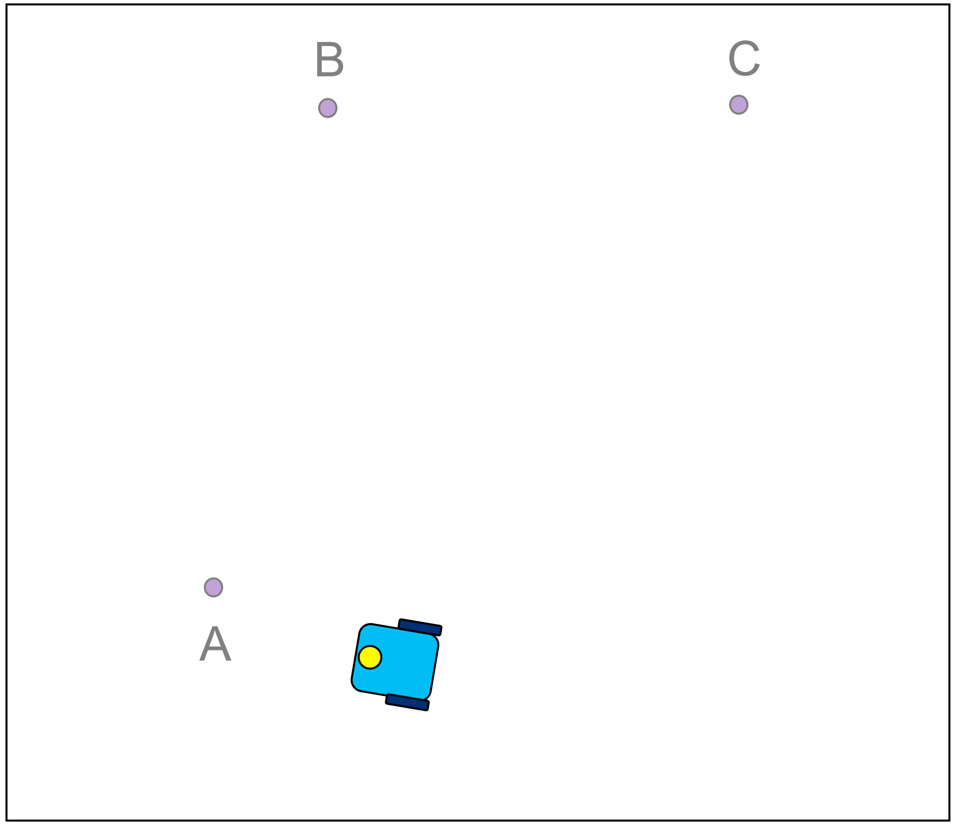
The videos are courtesy of Andrew J. Davison

- Pick natural scene features to serve as landmarks (in most modern SLAM systems)
- Range sensing (laser/sonar): line segments, 3D planes, corners
- Vision: point features, lines, textured surfaces.
- **Key**: features must be distinctive & recognizable from different viewpoints

how to do SLAM | with a Gaussian Filter

V4RL VISION FOR ROBOTICS LAB

- Use internal representations for
  - the positions of landmarks (: map)
  - the camera parameters
- Assumption:
  Robot's uncertainty at starting position is zero

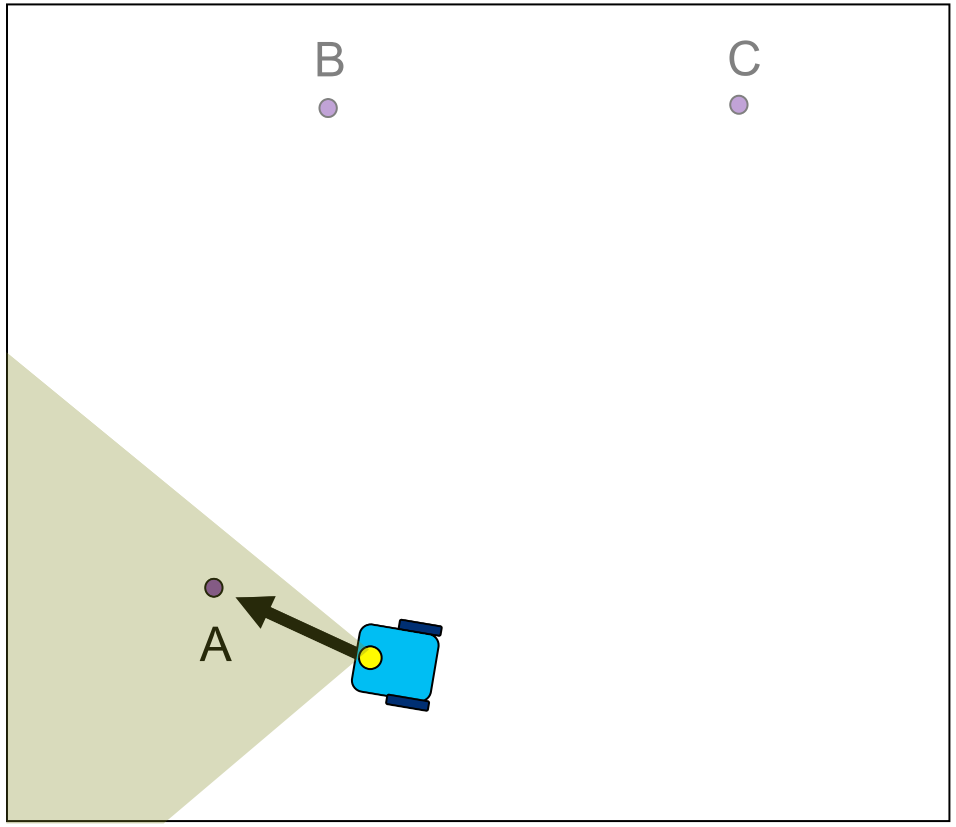


Start: robot has zero uncertainty

how to do SLAM | with a Gaussian Filter

On every frame:

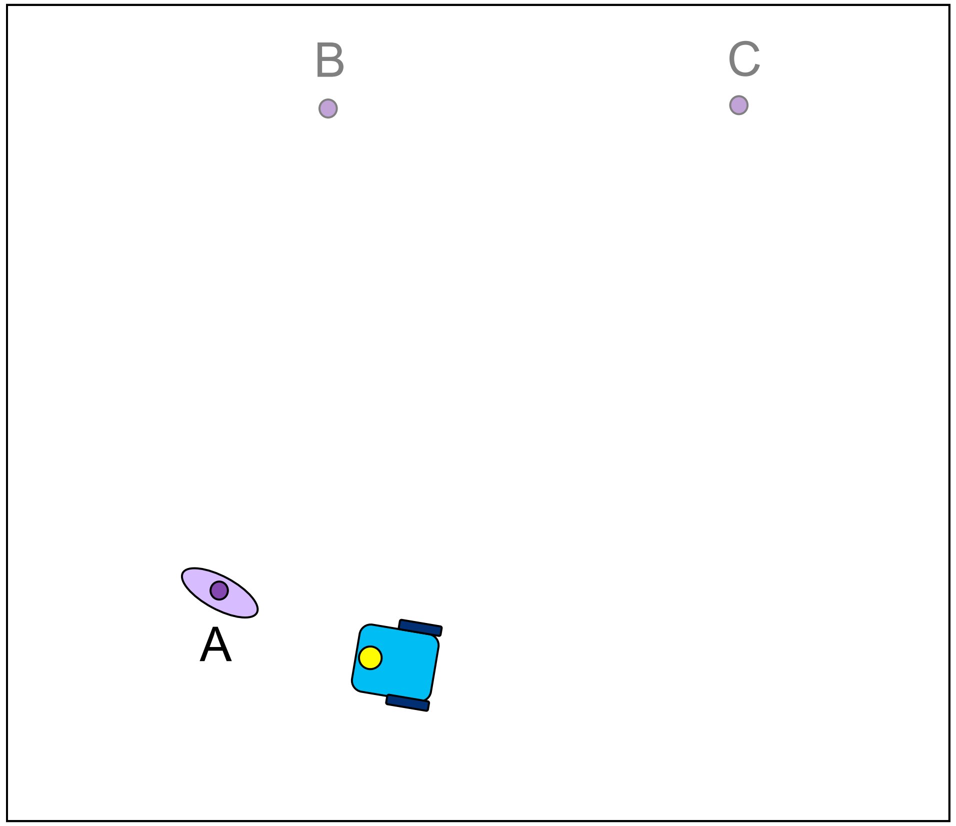
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



First measurement of feature A

how to do SLAM | with a Gaussian Filter

- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**

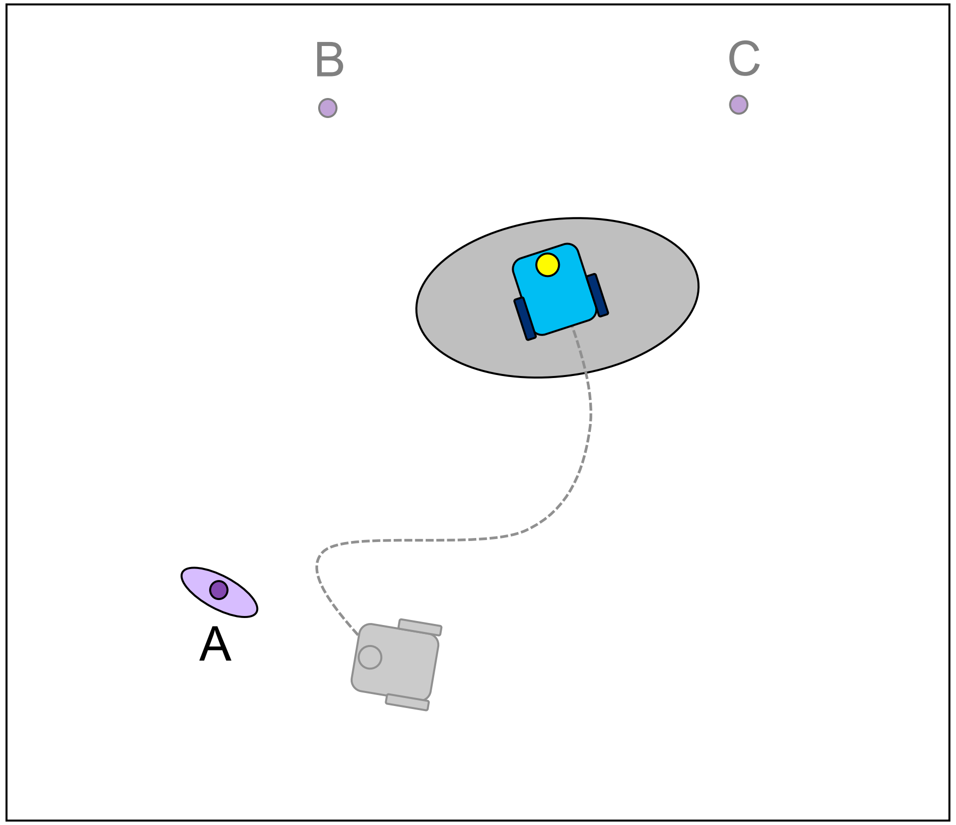


On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

how to do SLAM | with a Gaussian Filter

- As the robot moves, its pose uncertainty increases, obeying the robot's **motion model**.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

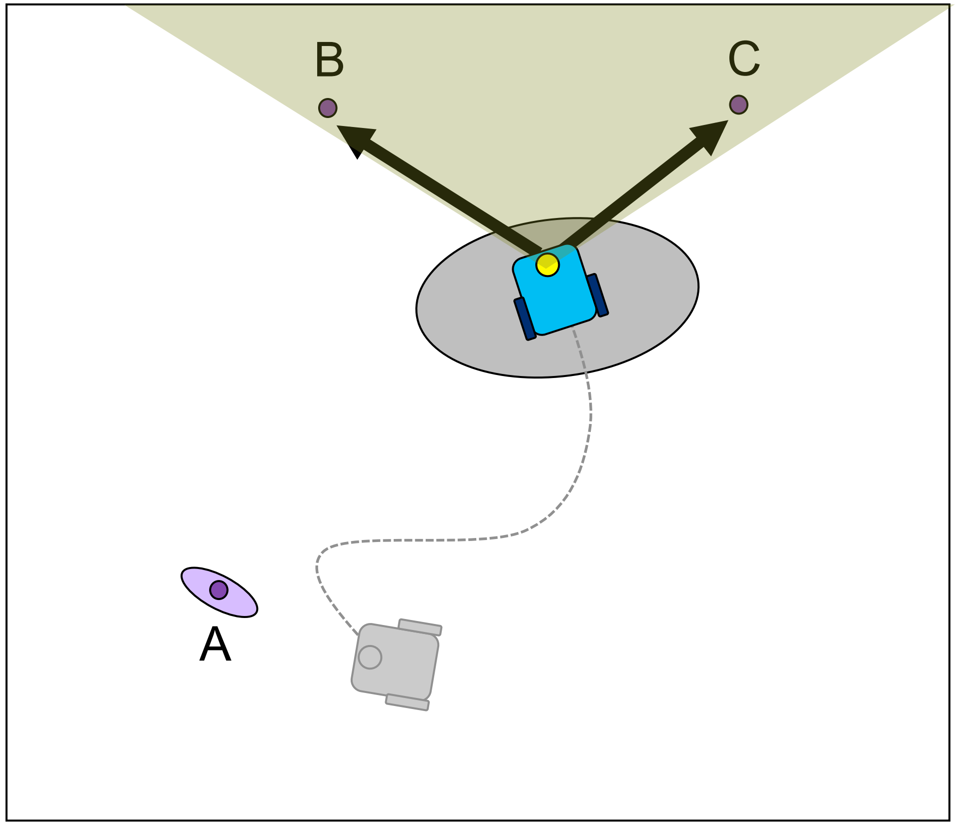


Robot moves forwards: uncertainty grows

how to do SLAM | with a Gaussian Filter

- Robot observes two new features.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



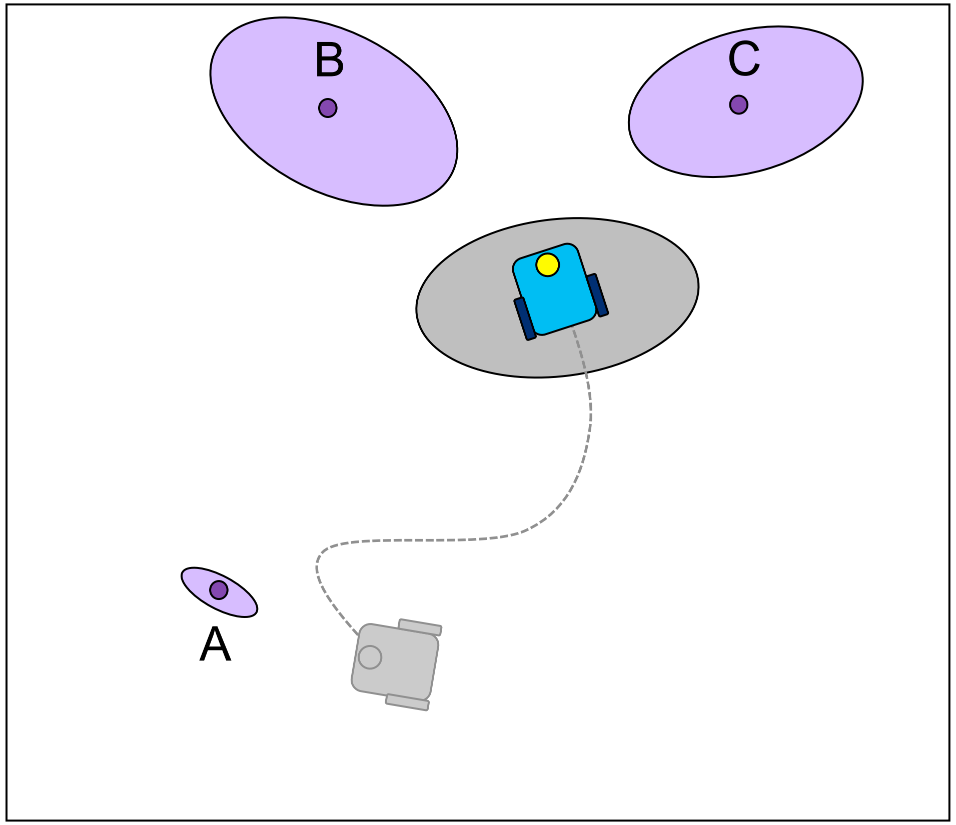
Robot makes first measurements of B & C

how to do SLAM | with a Gaussian Filter

- Their position uncertainty results from the **combination** of the measurement error with the robot pose uncertainty.

⇨ map becomes **correlated** with the robot pose estimate.

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



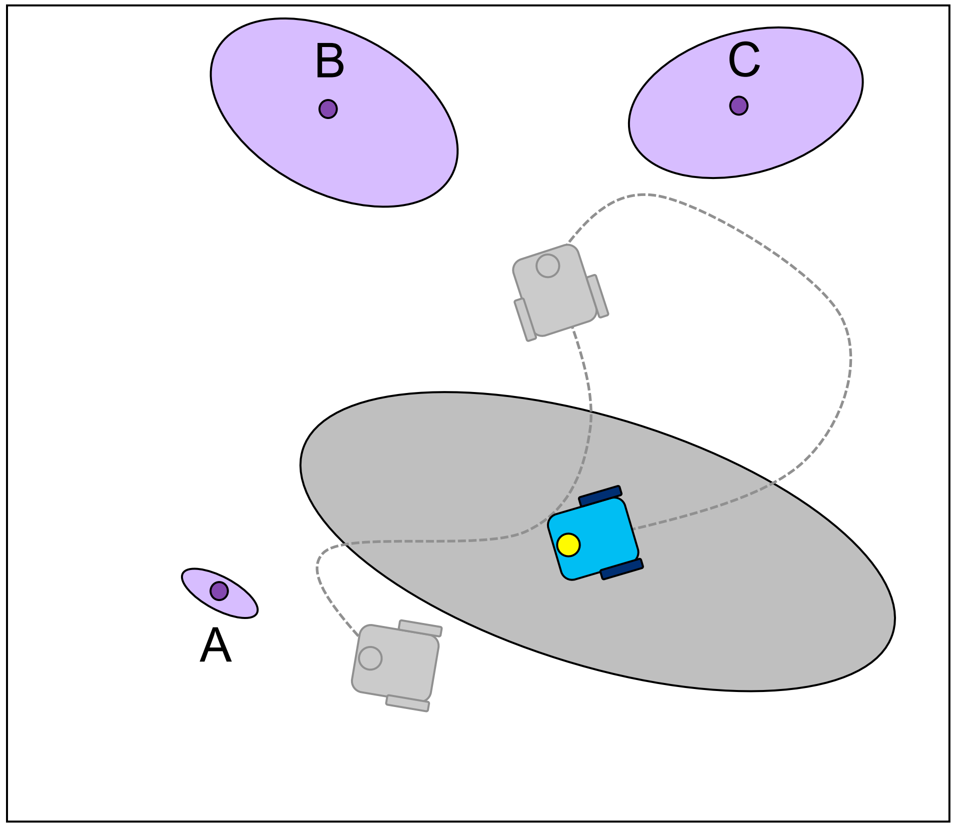
Robot makes first measurements of B & C

how to do SLAM | with a Gaussian Filter

- Robot moves again and its uncertainty increases (motion model)

On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

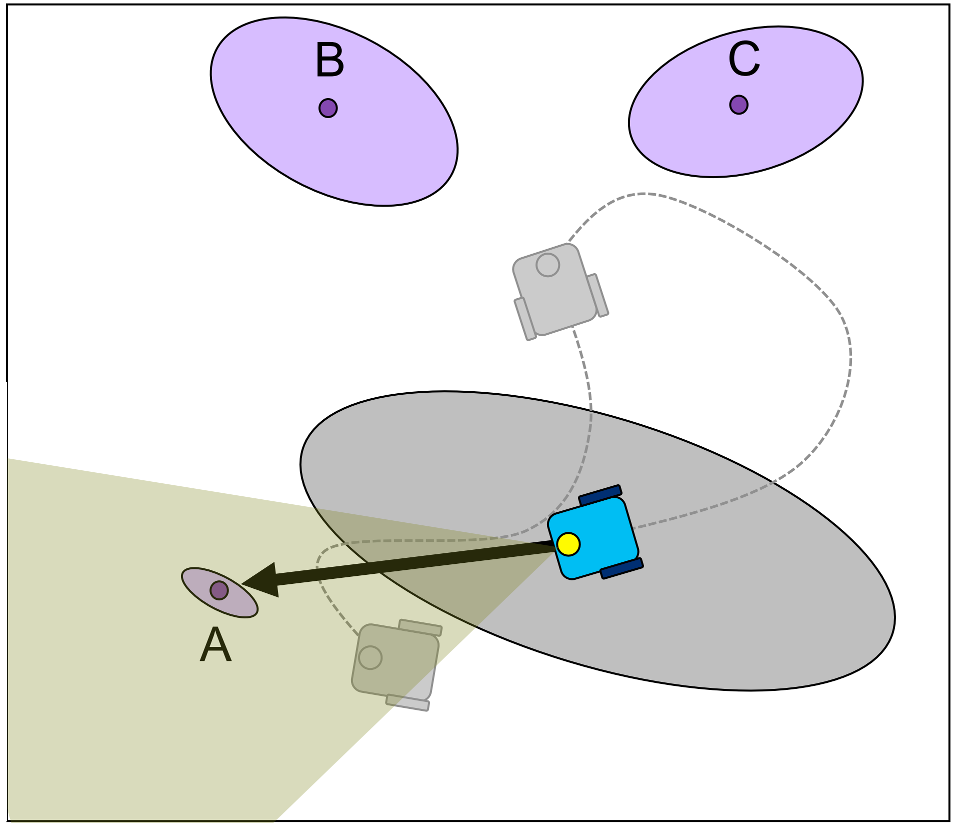


Robot moves again: uncertainty grows more

how to do SLAM | with a Gaussian Filter

- Robot re-observes an old feature
  ➪ **Loop closure** detection

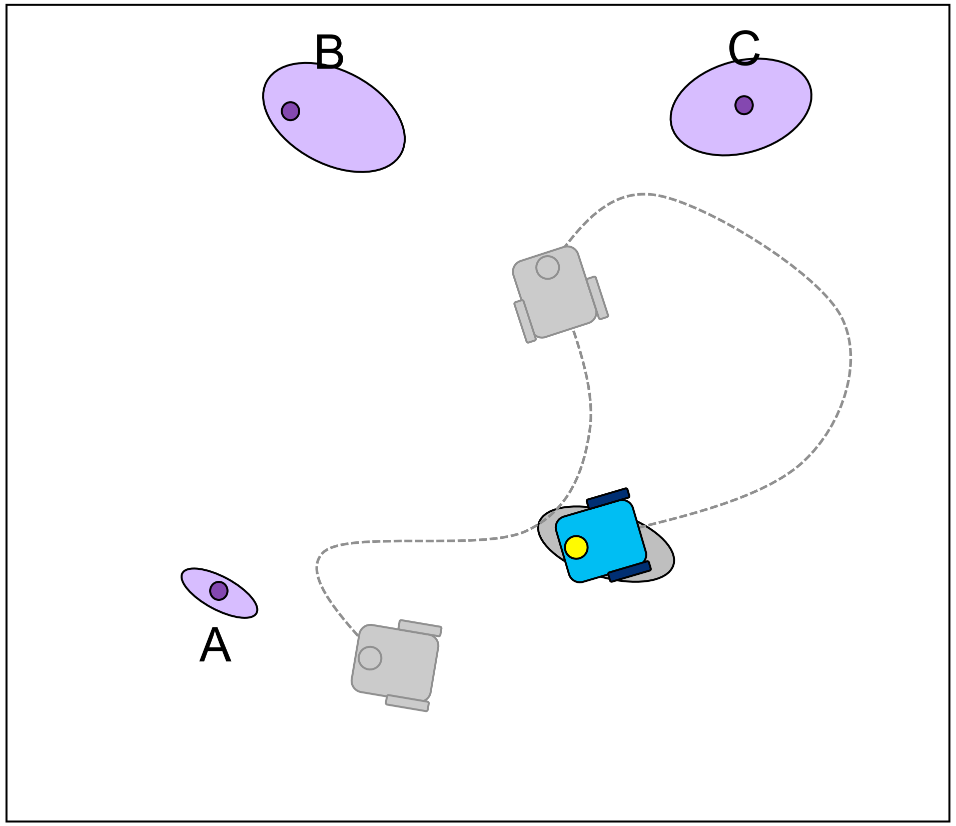
On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



how to do SLAM | with a Gaussian Filter

- Robot updates its position: the resulting **pose** estimate becomes **correlated** with the feature **location estimates**.
- Robot's uncertainty **shrinks** and so does the uncertainty in the rest of the map

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

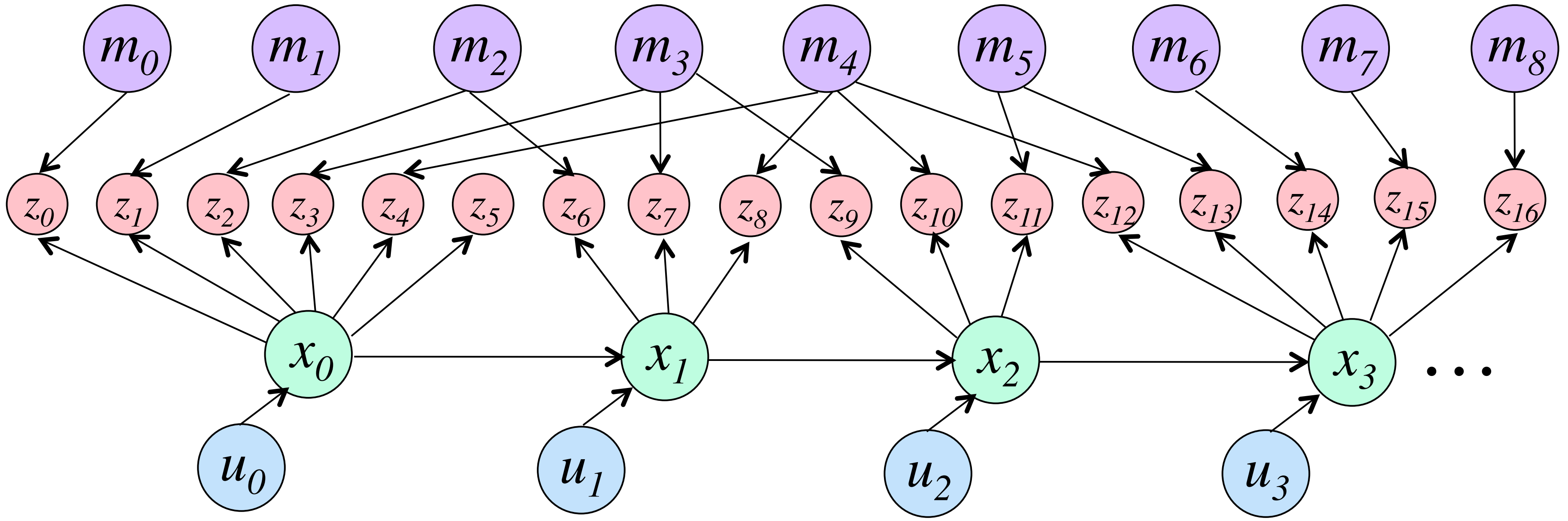


Robot re-measures A: "**loop closure**"
uncertainty shrinks

SLAM | probabilistic formulation

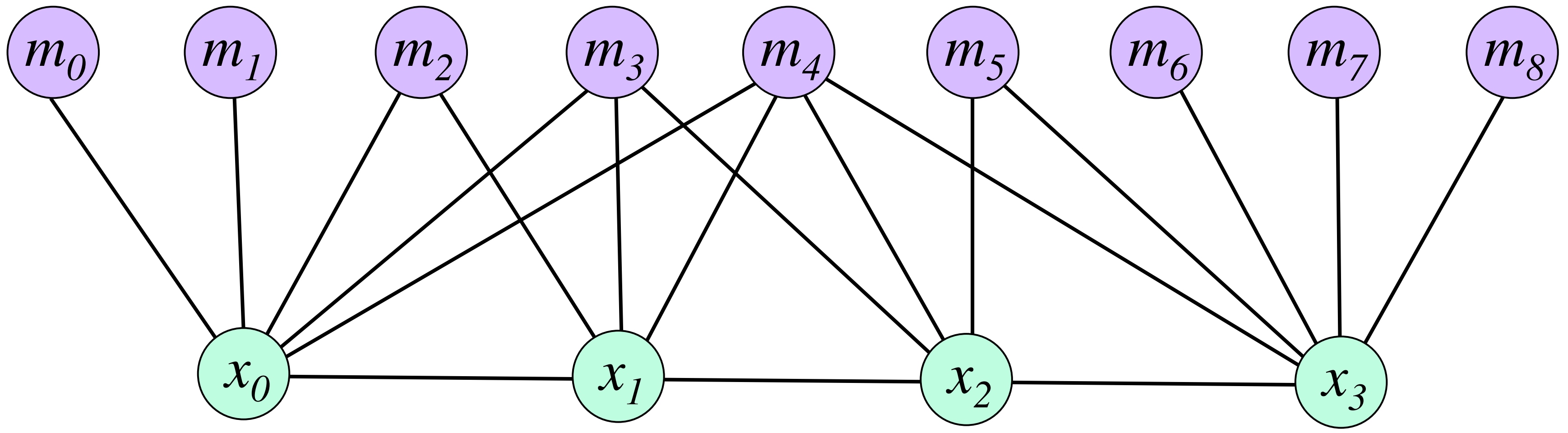
- Using the notation of [Davison et al., PAMI 2007]
- Robot **pose** at time t : x_t ⇨ Robot **path** up to this time: $\{x_0, x_1, \ldots, x_t\}$
- Robot **motion** between time $t\text{-}1$ and t : u_t (control inputs / proprioceptive sensor readings) ⇨ Sequence of robot relative motions: $\{u_0, u_1, \ldots, u_t\}$
- The **true map** of the environment: $\{m_0, m_1, \ldots, m_N\}$
- At each time t the robot makes measurements z_i
  ⇨ Set of all measurements (observations): $\{z_0, z_1, \ldots, z_k\}$
- The Full SLAM problem: estimate the posterior ➡ $p(x_{0:t}, m_{0:N} \mid z_{0:k}, u_{0:t})$
- The Online SLAM problem: estimate the posterior ➡ $p(x_t, m_{0:N} \mid z_{0:k}, u_{0:t})$

SLAM | graphical representation


m_0 m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8
z_0 z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 z_9 z_{10} z_{11} z_{12} z_{13} z_{14} z_{15} z_{16}
x_0 x_1 x_2 x_3 …
u_0 u_1 u_2 u_3

SLAM | approaches to SLAM

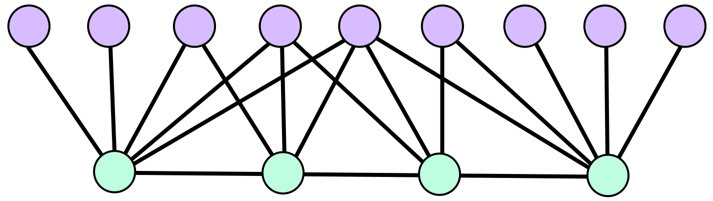
Full graph optimization (Bundle Adjustment)



- Eliminate observations & control-input nodes and solve for the constraints between poses and landmarks.
- Globally consistent solution, but infeasible for large-scale SLAM

⇨ If real-time is a requirement, we need to **sparsify** this graph

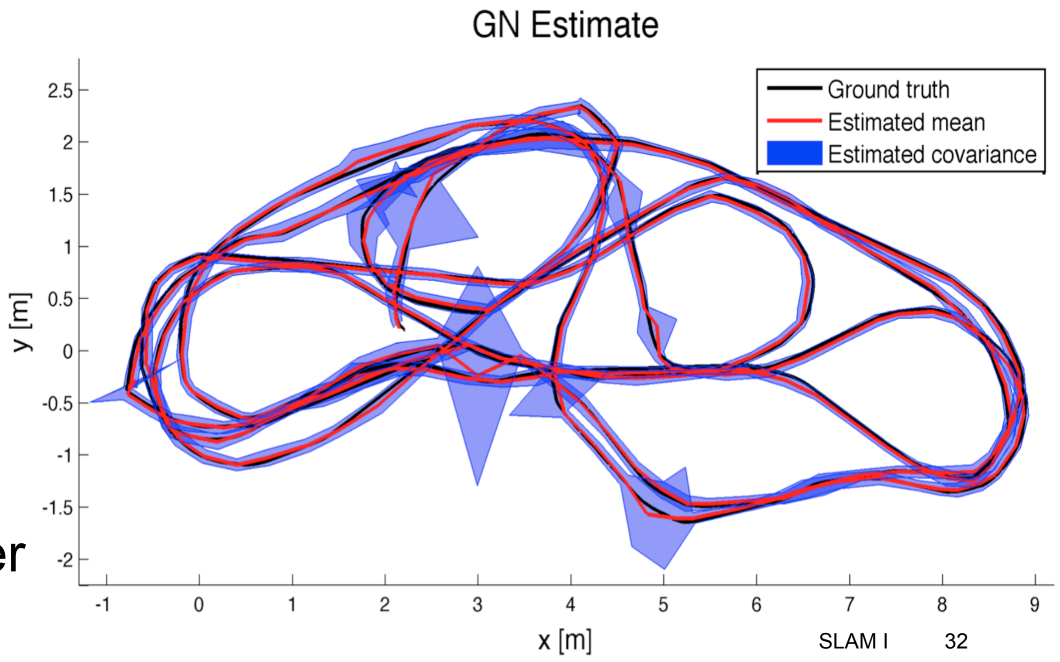
SLAM | full graph optimization



Full graph optimization (Bundle Adjustment)

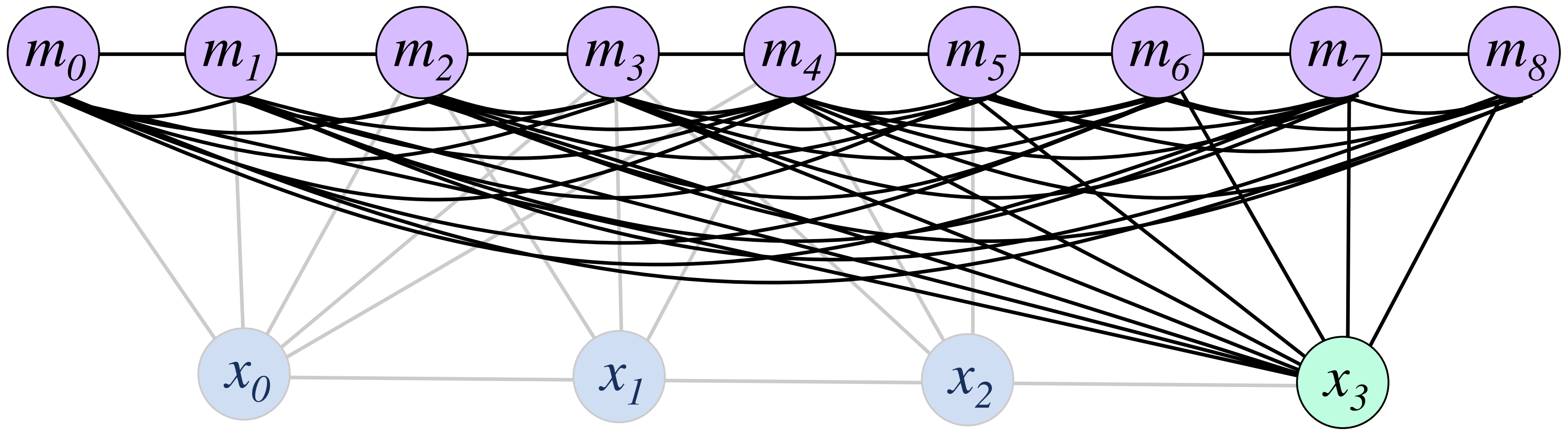
- Minimize the total least-squares cost function – the reprojection error
- Use a batch Maximum Likelihood approach
- Assume Gaussian noise densities

- Pros
  - ✓ Information can move backward in time
  - ✓ Trajectories can be very smooth
  - ✓ Best possible (most likely) estimate given the data and models
  - ✓ Exploitation of matrix sparsity leads to more efficient solutions
- Cons
  - x Computationally demanding
  - x Difficult to provide the online estimates for a controller



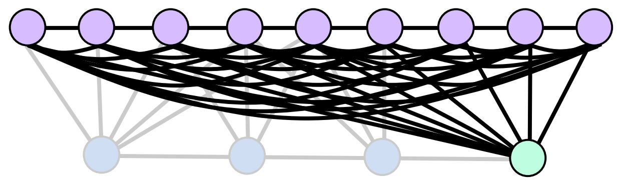
SLAM | approaches to SLAM

Filtering



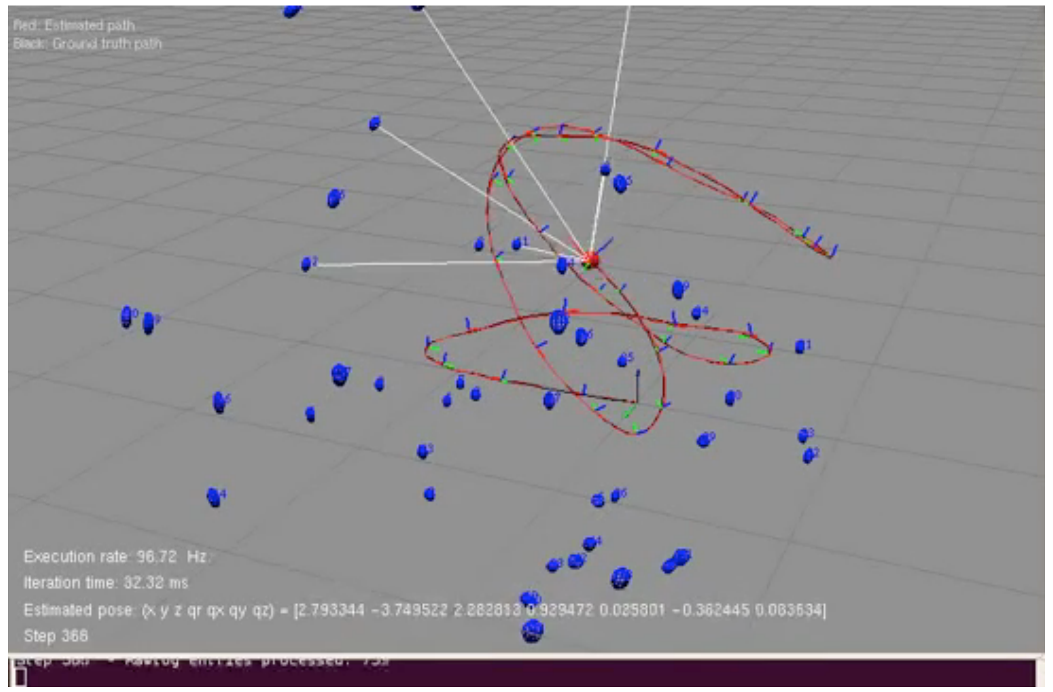
- Eliminate all past poses: ‘summarize’ all experience with respect to the last pose, using a **state vector** and the associated **covariance matrix**

SLAM | filtering



- **Gaussian Filtering** (EKF, UKF)
  - Tracks a Gaussian belief of the state/landmarks
  - Assumes all noise is Gaussian
  - Follows the “predict/measure/update” approach
- Pros
  - ✓ Can run online
  - ✓ Works well for problems experiencing expected perturbations/uncertainty
- Cons
  - x Unimodal estimate
  - x States must be well approximated by a Gaussian
  - x The vanilla implementation does not scale very well with larger maps

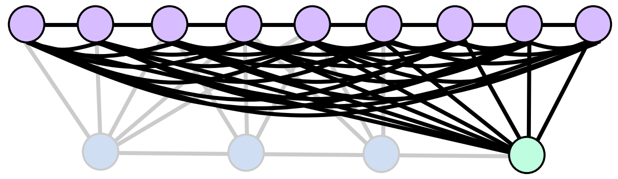
Courtesy of P. Furgale



```
Execution rate: 96.72  Hz
Iteration time: 32.32 ms
Estimated pose: (x y z qr qx qy qz) = [2.793344 -3.749522 2.282813 0.929472 0.025801 -0.362445 0.083634]
Step 368
```

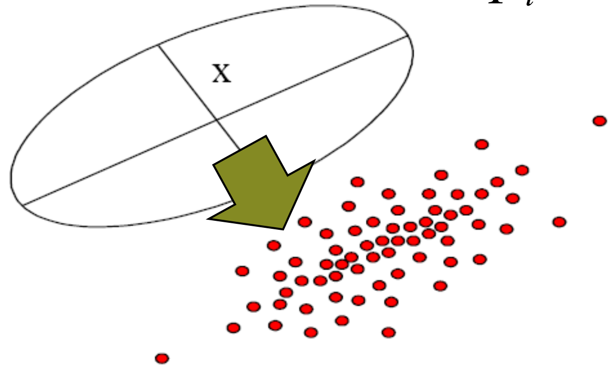
```
Step 368 - Rawlog entries processed: 735
```

SLAM | filtering



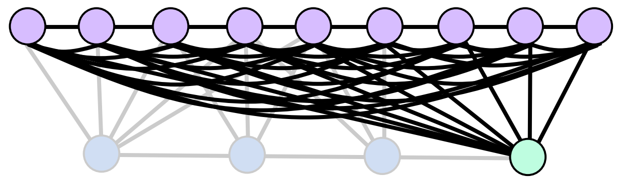
- **Particle Filtering**
  - Represents belief by a series of samples
  - **Each Particle** = a hypothesis of the state (= a suggested pose & map) with an associated weight
    (all weights should add up to 1)
  - Follows the "predict/measure/update" approach

$$p_i = \{y_{t_i}, w_i\}$$

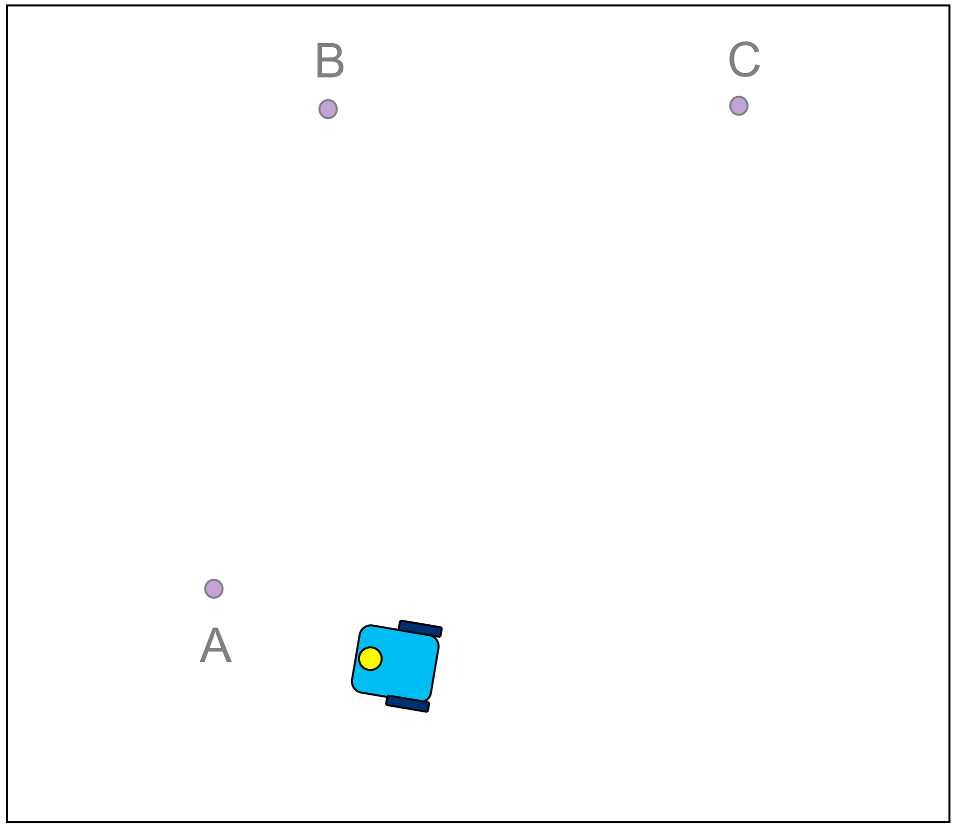


probability distribution (ellipse) as particle set (red dots)

how to do SLAM | with a Particle Filter

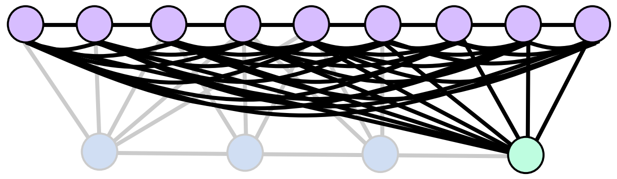


- Use internal representations for
  - the positions of landmarks (: map)
  - the camera parameters
- Assumption:
  Robot's uncertainty at starting position is zero
- Initialize N particles at the origin, each with weight 1/N



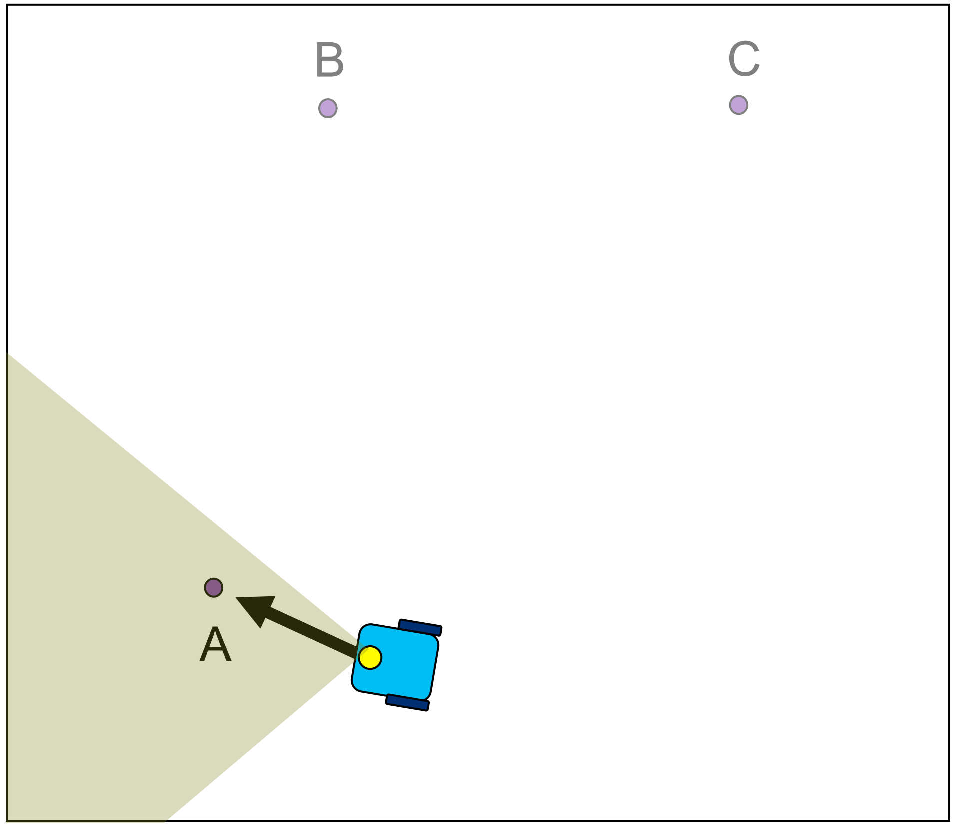
Start: robot has zero uncertainty

how to do SLAM | with a Particle Filter



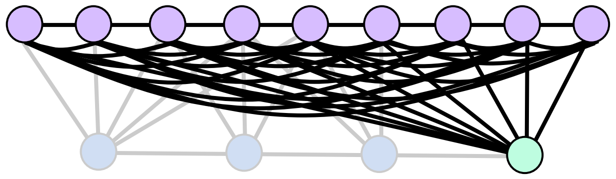
On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

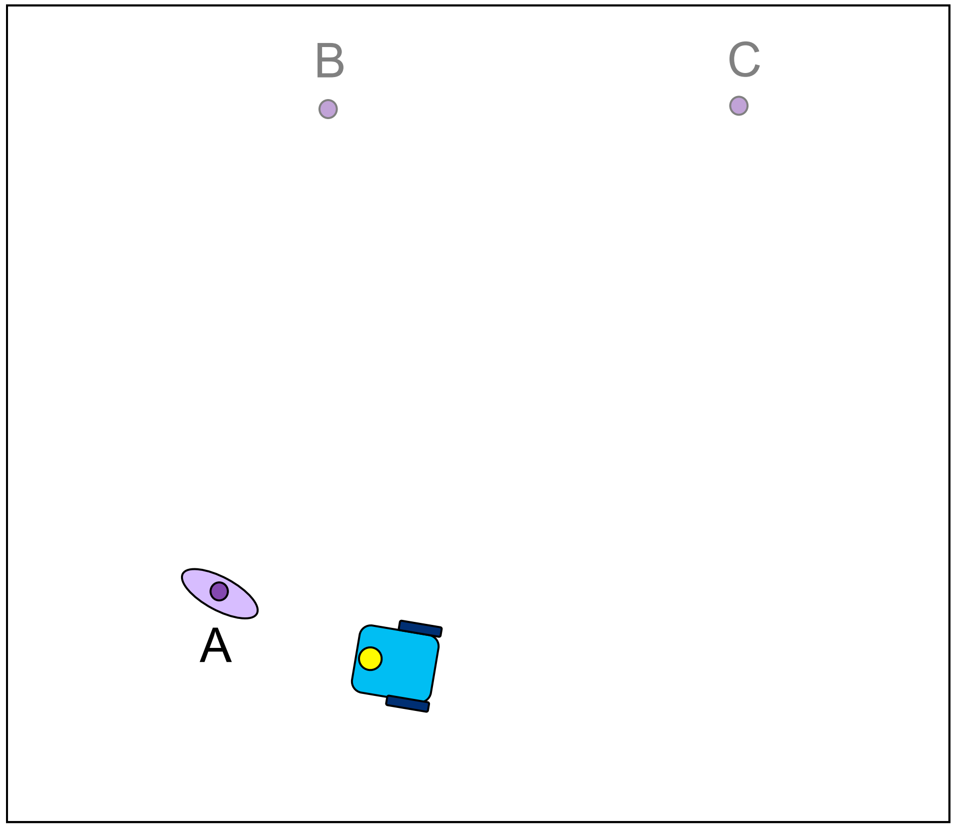


First measurement of feature A

how to do SLAM | with a Particle Filter



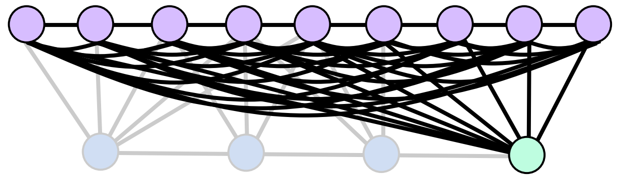
- The robot observes a feature which is mapped with an uncertainty related to the **measurement model**



On every frame:

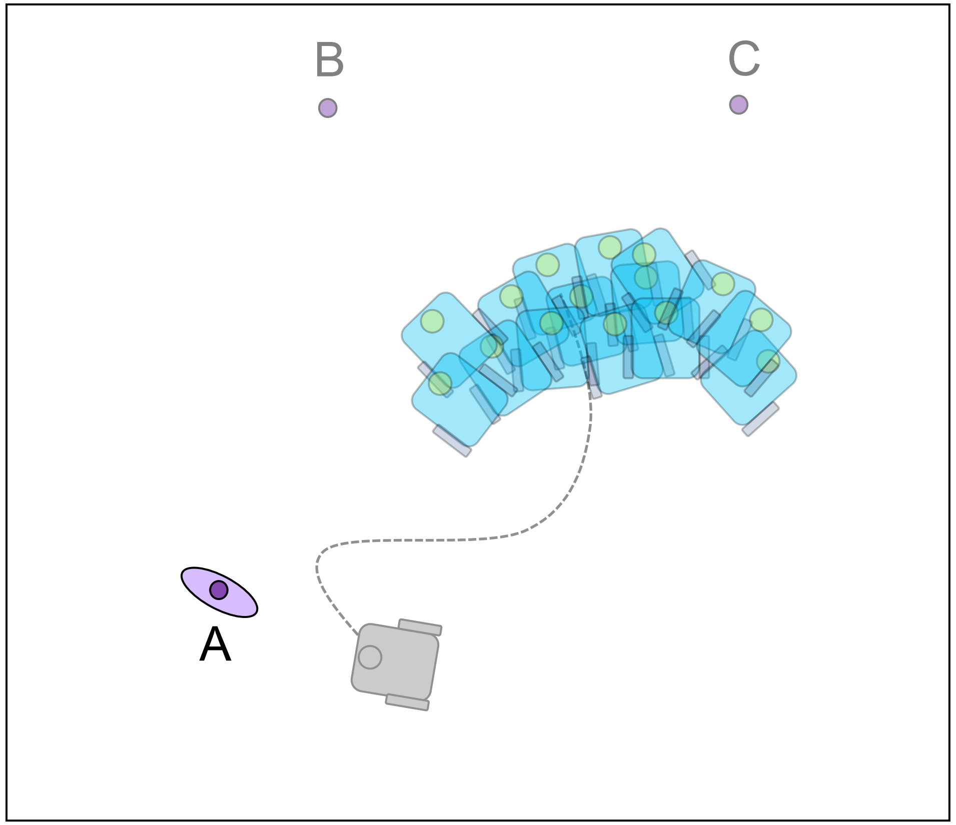
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

how to do SLAM | with a Particle Filter



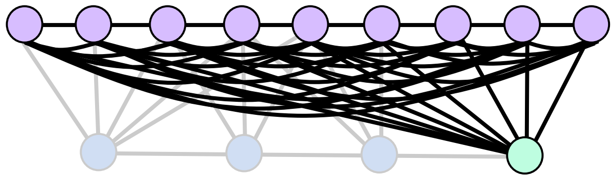
- As the robot moves, its pose uncertainty increases, obeying the robot's **motion model**.
- Apply motion model to each particle

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot moves forwards: uncertainty grows

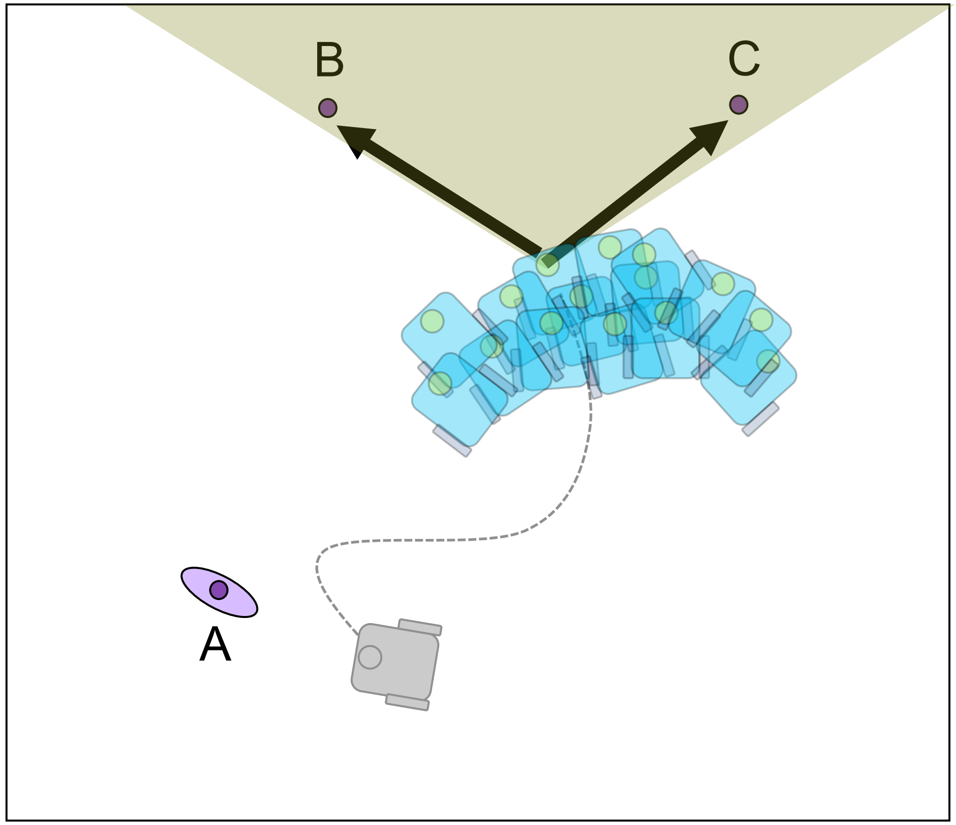
how to do SLAM | with a Particle Filter



- Robot observes two new features.

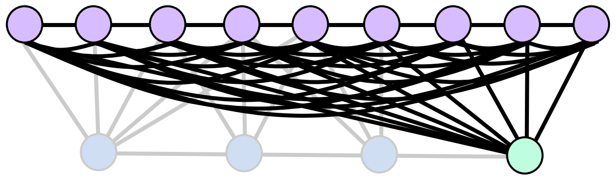
On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot makes first measurements of B & C

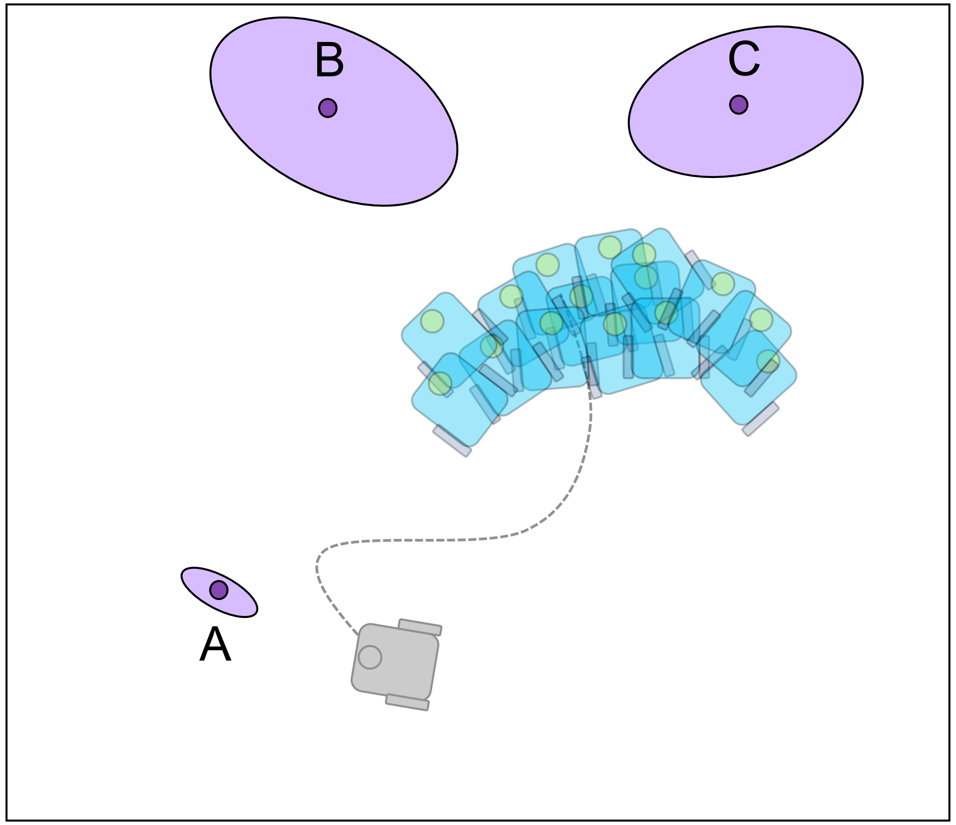
how to do SLAM | with a Particle Filter



- Their position uncertainty is encoded for each particle individually

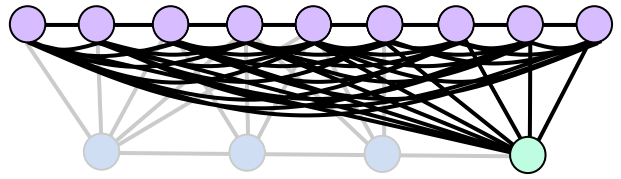
  For each particle:
  - Compare the particle's predicted measurements with the obtained measurements
  - Re-weigh such that particles with good predictions get higher weight & re-normalize particle weights
  - Re-sample according to likelihood

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



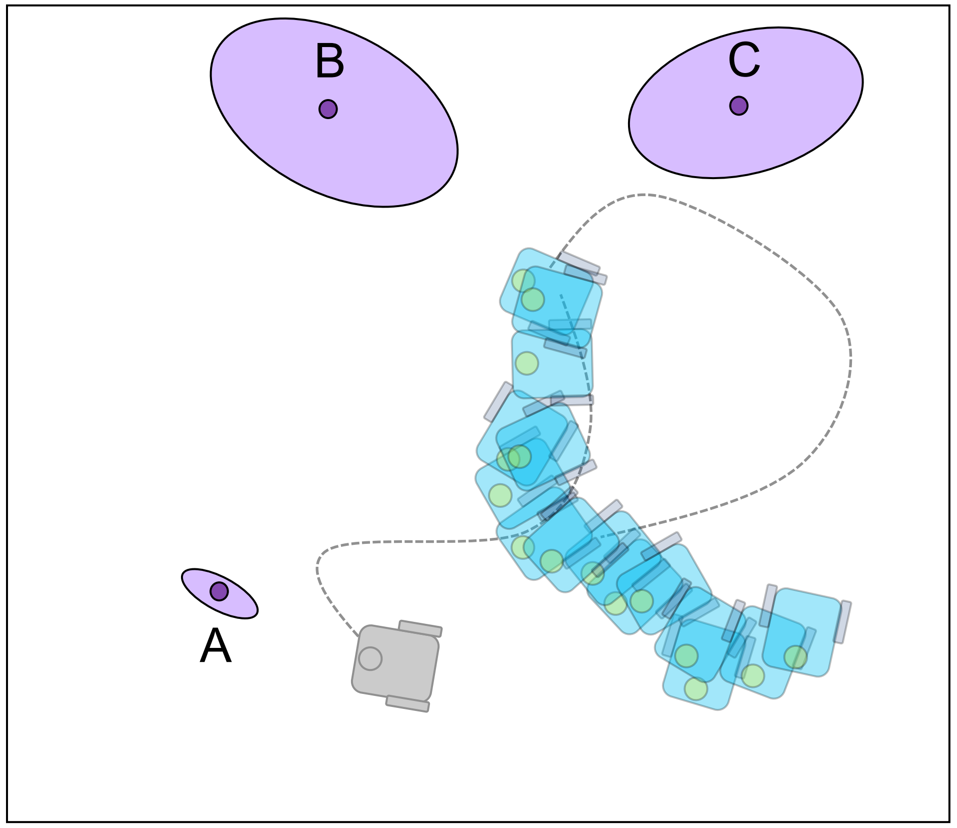
Robot makes first measurements of B & C

how to do SLAM | with a Particle Filter



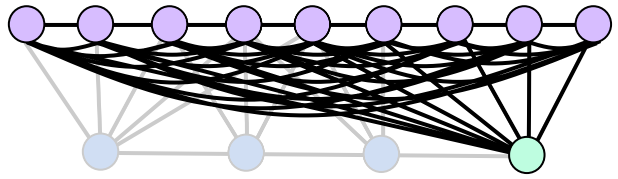
- Robot moves again and its uncertainty increases (motion model)
- Apply motion model to each particle

On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

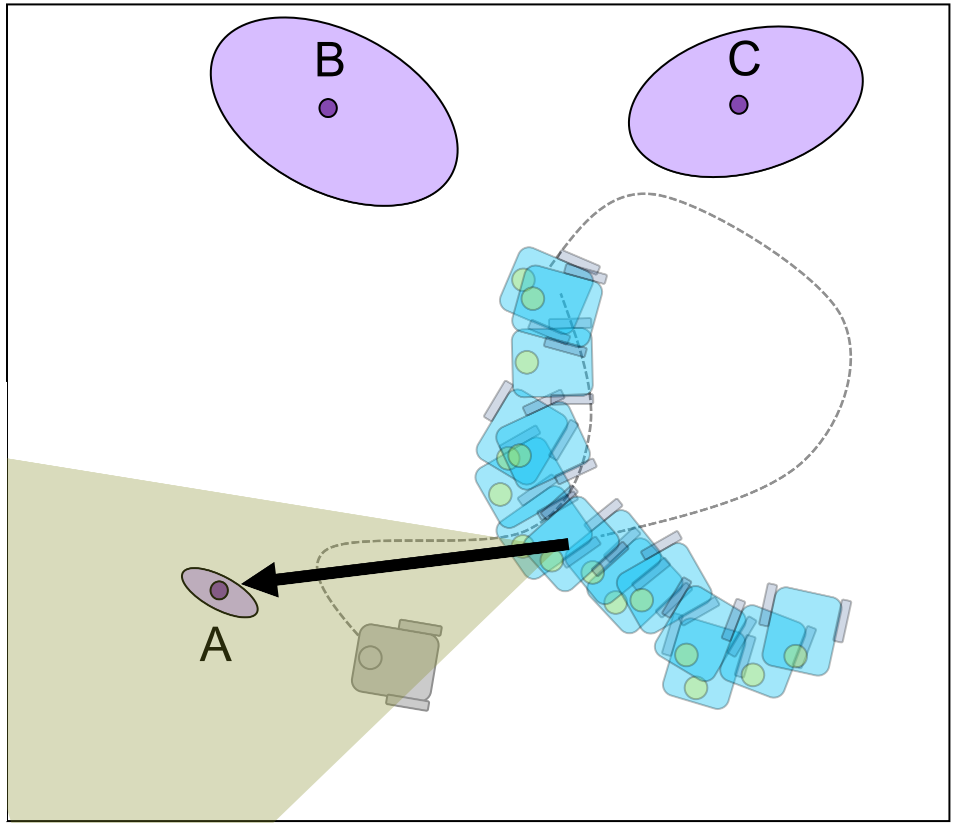


Robot moves again: uncertainty grows more

how to do SLAM | with a Particle Filter

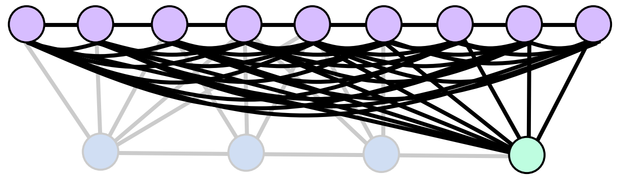


- Robot moves again and its uncertainty increases (motion model)

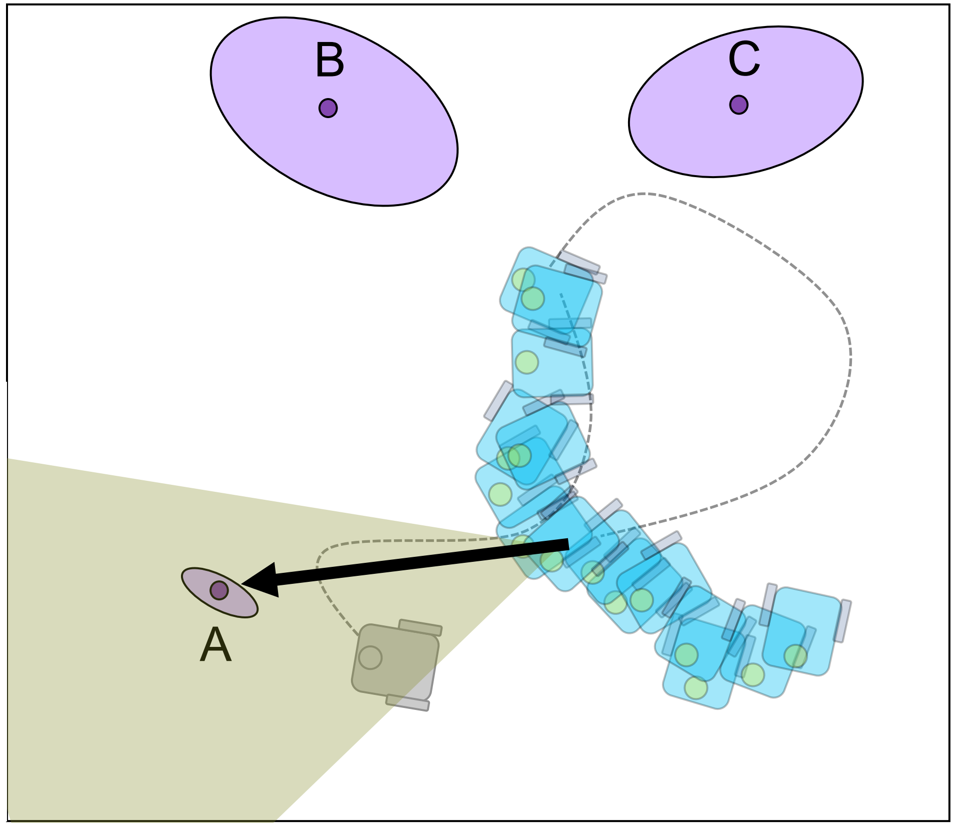


On every frame:
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

how to do SLAM | with a Particle Filter



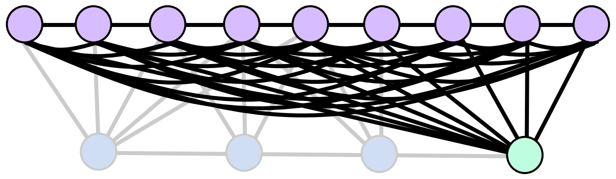
- Robot re-observes an old feature
  ➪ **Loop closure** detection



On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

how to do SLAM | with a Particle Filter

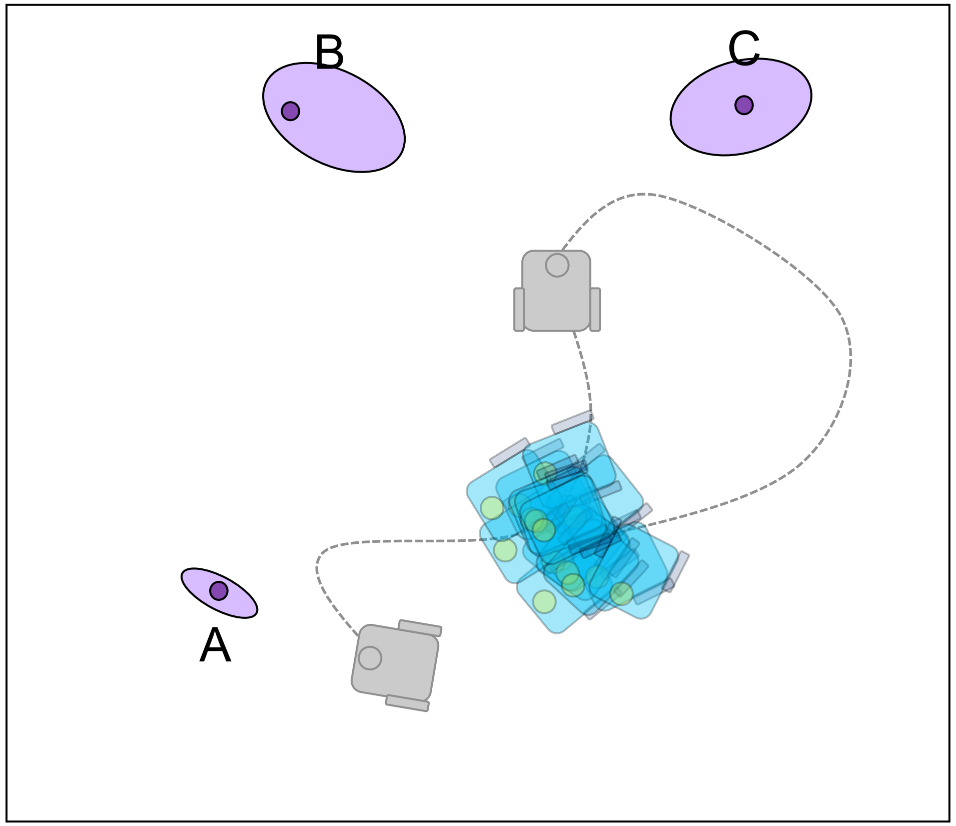


For each particle:

- Compare the particle's predicted measurements with the obtained measurements
- Re-weigh such that particles with good predictions get higher weight & re-normalize particle weights
- Re-sample according to likelihood

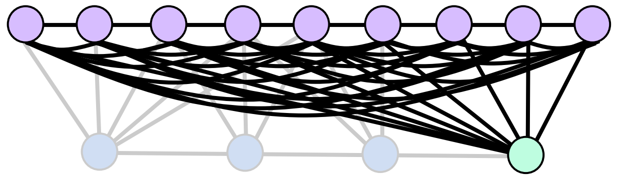
On every frame:

- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



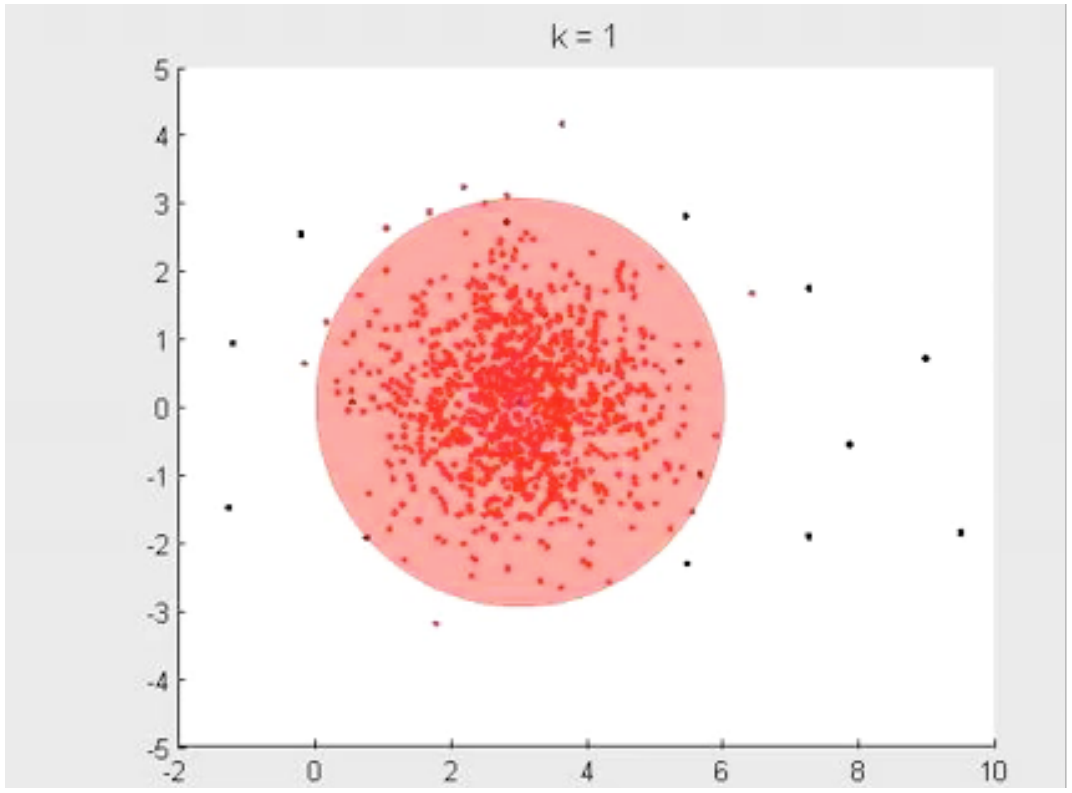
Robot re-measures A: "loop closure"
uncertainty shrinks

SLAM | filtering



- **Particle Filtering**
  - Represents belief by a series of samples
  - **Each Particle** = a hypothesis of the state with an associated weight (all weights should add up to 1)
  - Follow the "predict/measure/update" approach
- Pros
  - ✓ Noise densities can be from any distribution
  - ✓ Works for multi-modal distributions
  - ✓ Easy to implement
- Cons
  - x Does not scale to high-dimensional problems
  - x Requires many particles to have good convergence

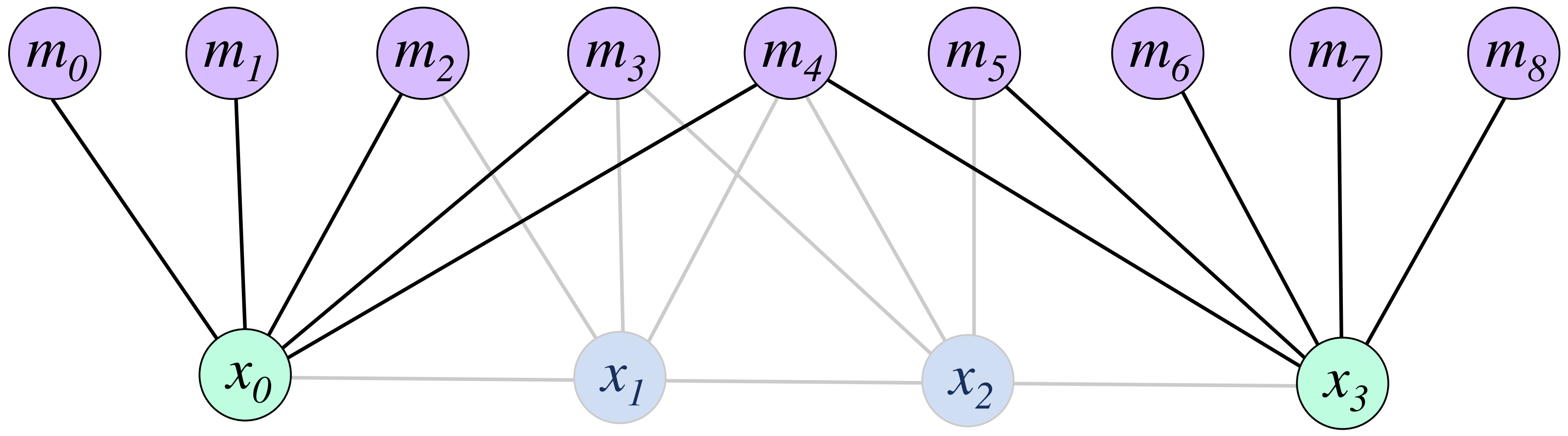
Courtesy of P. Furgale



Distribution in the robot's position estimate:
- red dots – particle filtering
- red ellipse – EKF filtering

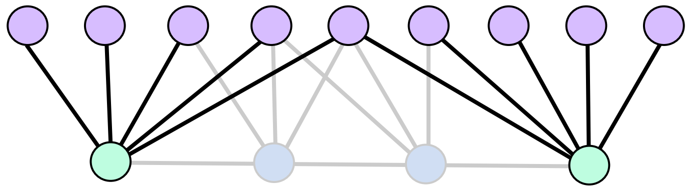
SLAM | approaches to SLAM

Key-frames



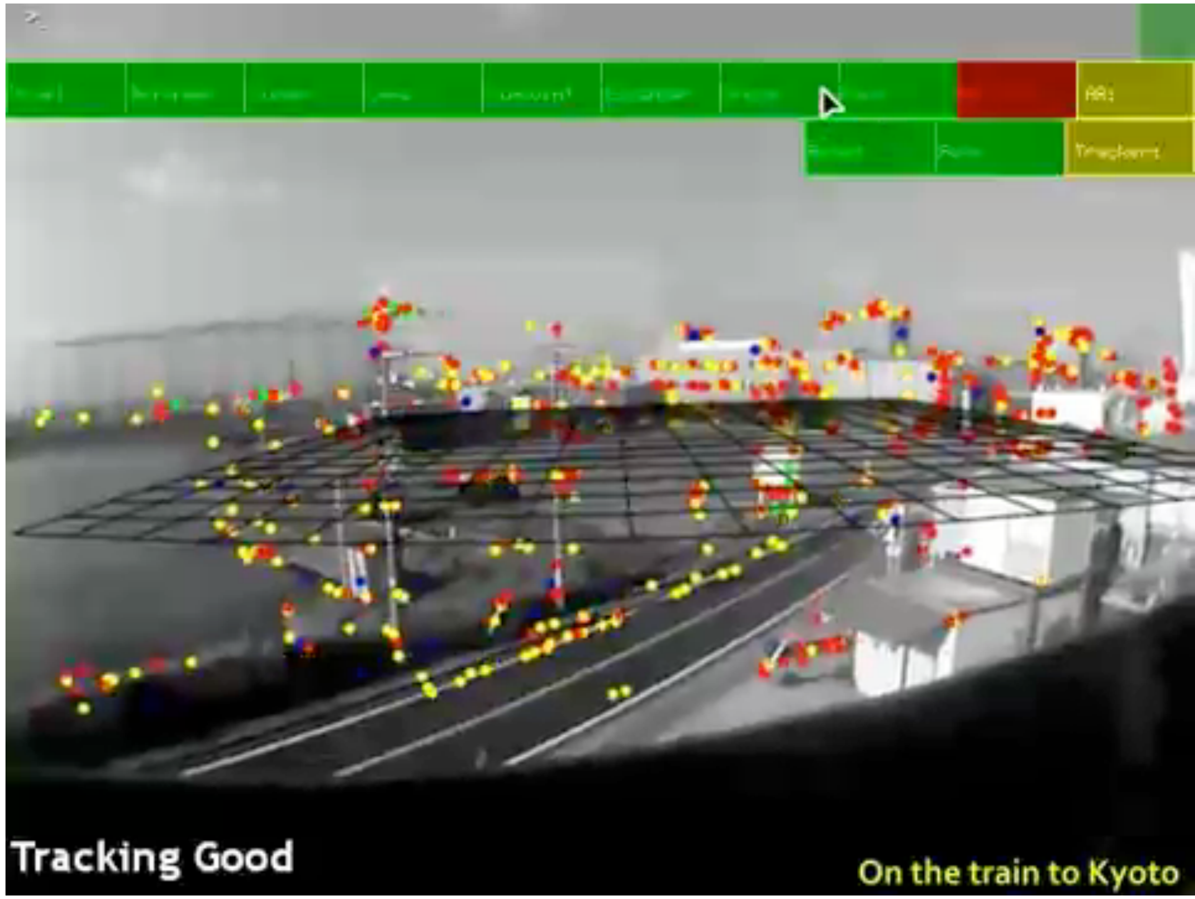
- Retain the most 'representative' poses (key-frames) and their dependency links ➪ optimize the resulting graph
- Examples: PTAM [Klein & Murray, ISMAR 2007] ,
  ORB-SLAM [Mur-Artal et al., TRO 2015]

SLAM | keyframes



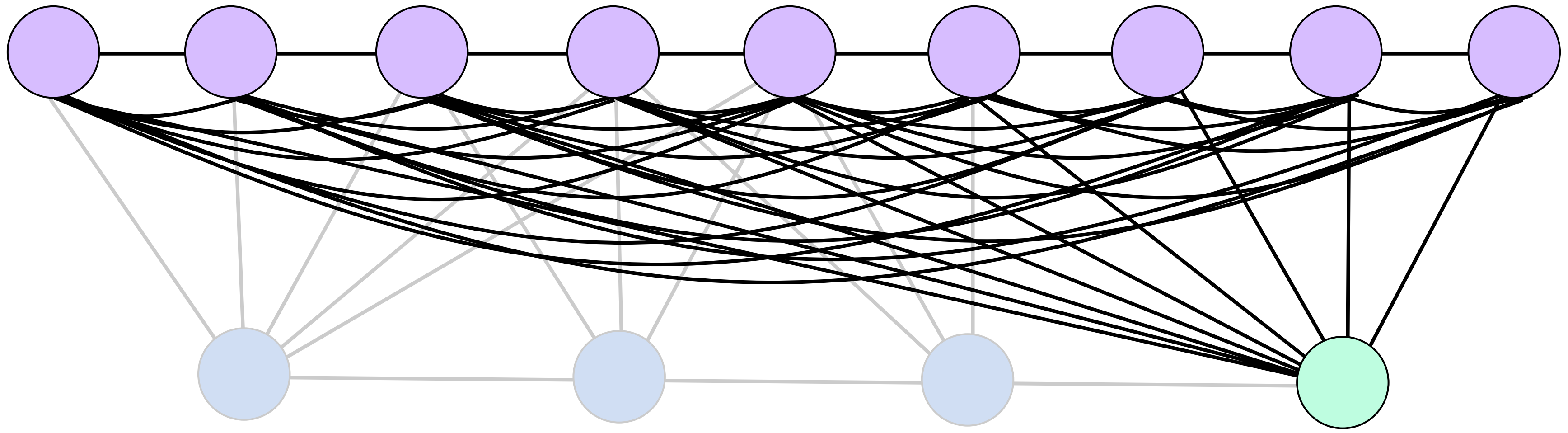
- **Keyframe-based SLAM**
  - Minimizes the least-squares cost function
  - Typically optimizes over a window of recent keyframes for efficiency
  - Assumes Gaussian noise densities
- Pros
  - ✓ Known to provide better balance between accuracy & efficiency than filtering
  - ✓ Permits processing of many more features per frame than filtering
- Cons
  - x Size of optimization window affects scalability and convergence

[PTAM, Klein & Murray, ISMAR 2007]



EKF SLAM | overview

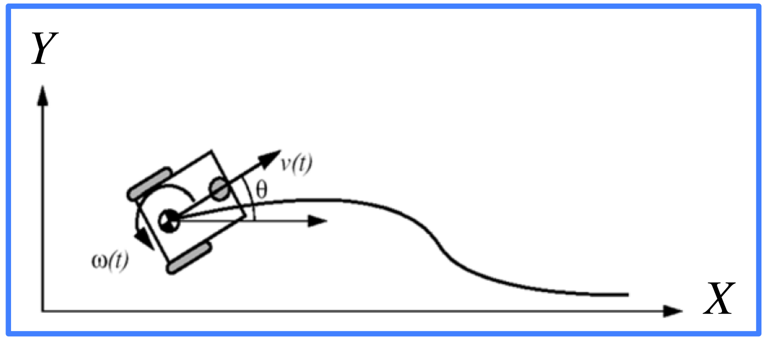
- SLAM using an Extended Kalman Filter (EKF)



EKF SLAM | overview

- EKF SLAM summarizes all past experience in an **extended state vector** $\boldsymbol{y_t}$ comprising of the robot pose $\boldsymbol{x_t}$ and the position of all the features $\boldsymbol{m_i}$ in the map, and an associated **covariance matrix** $\boldsymbol{P_{y_t}}$:

$$y_t = \begin{bmatrix} x_t \\ m_1 \\ \ldots \\ m_{n-1} \end{bmatrix} , \quad P_{y_t} = \begin{bmatrix} P_{xx} & P_{xm_1} & .. & P_{xm_{n-1}} \\ P_{m_1x} & P_{m_1m_1} & .. & P_{m_1m_{n-1}} \\ .. & .. & .. & .. \\ P_{m_{n-1}x} & P_{m_{n-1}m_1} & .. & P_{m_{n-1}m_{n-1}} \end{bmatrix}$$



- If we sense 2D line-landmarks, the size of $\boldsymbol{y_t}$ is $3+2n$ (and size of $\boldsymbol{P_{y_t}}$: $(3+2n)(3+2n)$)
  - 3 variables to represent the robot pose and
  - $2n$ variables for the n line-landmarks with state components

  Hence, $y_t = [X_t, Y_t, \theta_t, \alpha_0, r_0, \ldots, \alpha_{n-1}, r_{n-1}]^T$

- As the robot moves and makes measurements, $\boldsymbol{y_t}$ and $\boldsymbol{P_{y_t}}$ are updated using the **standard EKF equations**

EKF SLAM | prediction step

- The predicted robot pose $\hat{x}_t$ at time-stamp t is computed using the estimated pose x_{t-1} at time-stamp t-1 and the odometric control input $u_t = \{\Delta S_l, \Delta S_r\}$

$$\hat{x}_t = f(x_{t-1}, u_t) = \begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} X_{t-1} \\ Y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

Odometry function

$\Delta S_l ; \Delta S_r$: distance travelled by the left and right wheels resp.
b : distance between the two robot wheels
(based on the example of Section 5.8.4 of the AMR book)

- During this step, the position of the features remains unchanged. EKF Prediction Equations:

$$\hat{y}_t = \begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \\ \hat{\alpha}_0 \\ \hat{r}_0 \\ \dots \\ \hat{\alpha}_{n-1} \\ \hat{r}_{n-1} \end{bmatrix} = \begin{bmatrix} X_t \\ Y_t \\ \theta_t \\ \alpha_0 \\ r_0 \\ \dots \\ \hat{\alpha}_{n-1} \\ r_{n-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \end{bmatrix}$$

Jacobians of f

$$\hat{P}_{y_t} = F_y P_{y_{t-1}} F_y^T + F_u Q_t F_u^T$$

Covariance at previous time-stamp

Covariance of noise associated to the motion

EKF SLAM | vs. EKF localization

EKF LOCALIZATION

- The state x_t is **only** the robot configuration:

$$x_t = [X_t, Y_t, \theta_t]^T$$

$$\hat{x}_t = f(x_{t-1}, u_t)$$

$$\begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} X_{t-1} \\ Y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2}\cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2}\sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

$$\hat{P}_{x_t} = F_x P_{x_{t-1}} F_x^T + F_u Q_t F_u^T$$

EKF SLAM

- The state y_t comprises of the robot configuration x_t **and** that of each feature m_i :

$$y_t = [X_t, Y_t, \theta_t, \alpha_0, r_0, \ldots, \alpha_{n-1}, r_{n-1}]^T$$

$$\hat{y}_t = f(y_{t-1}, u_t)$$

$$\hat{y}_t = \begin{bmatrix} \hat{X}_t \\ \hat{Y}_t \\ \hat{\theta}_t \\ \hat{\alpha}_0 \\ \hat{r}_0 \\ \ldots \\ \hat{\alpha}_{n-1} \\ \hat{r}_{n-1} \end{bmatrix} = \begin{bmatrix} X_t \\ Y_t \\ \theta_t \\ \alpha_0 \\ r_0 \\ \ldots \\ \hat{\alpha}_{n-1} \\ r_{n-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2}\cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2}\sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \\ 0 \\ 0 \\ \ldots \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{P}_{y_t} = F_y P_{y_{t-1}} F_y^T + F_u Q_t F_u^T$$

EKF SLAM | measurement prediction & update

- The application of the **measurement model** is the same as in EKF localization. The predicted observation of each feature m_i is:

$$\hat{z}_i = \begin{bmatrix} \hat{\alpha}_i \\ \hat{r}_i \end{bmatrix} = h_i(\hat{x}_t, m_i)$$

The predicted new pose is used to predict where each feature lies in measurement space

$$y_t = \hat{y}_t + K_t \underbrace{(z_{0:n-1} - h_{0:n-1}(\hat{x}_t, m_{0:n-1}))}_{\text{“Innovation” (= observation-prediction)}}$$

$$P_{y_t} = \hat{P}_{y_t} - K_t \Sigma_{IN} K_t^T$$

Jacobian of ***h***

Measurement noise

where

$$\Sigma_{IN} = H \hat{P}_{y_t} H^T + R$$

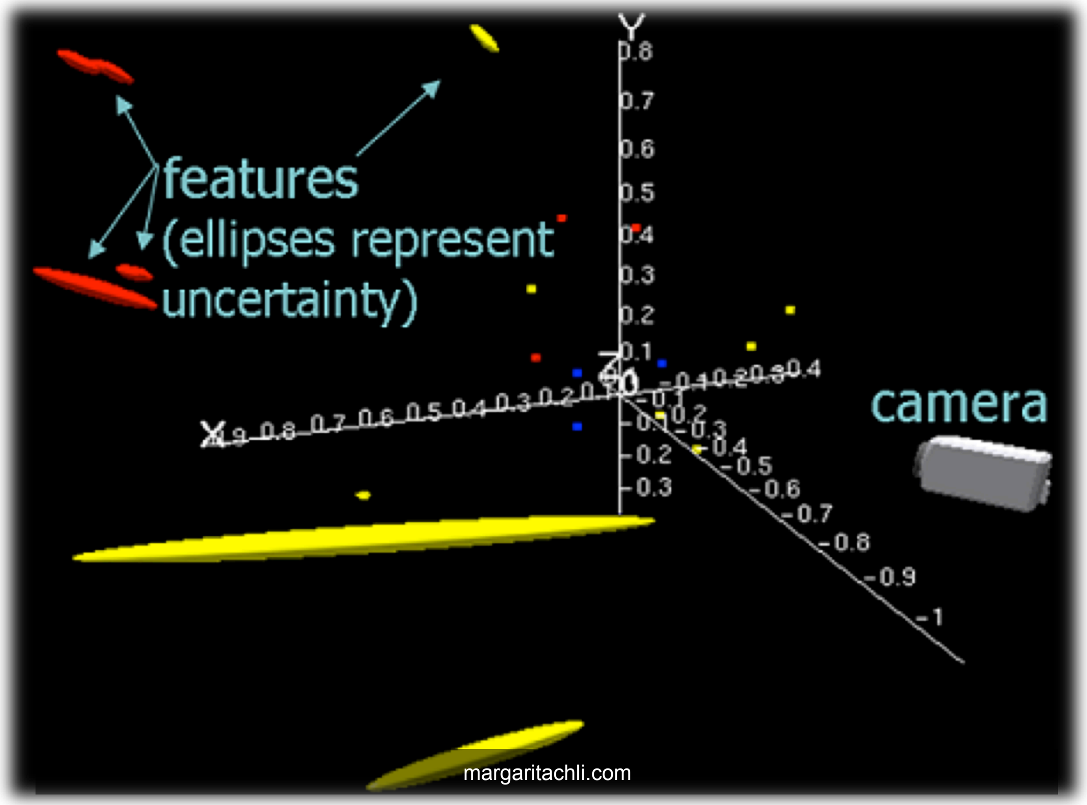
$$K_t = \hat{P}_{y_t} H (\Sigma_{IN})^{-1}$$

Kalman Gain

Innovation Covariance

MonoSLAM

- An example of EKF SLAM: **MonoSLAM**



[Davison, Reid, Molton and Stasse, PAMI 2007]

MonoSLAM | single camera SLAM

Vision
for SLAM

- Images = information-rich snapshots of a scene
- Compactness + affordability of cameras
- HW advances

- SLAM using a single, handheld camera:
- Hard but … (e.g. cannot recover depth from 1 image)
- **very** applicable, compact, affordable, …

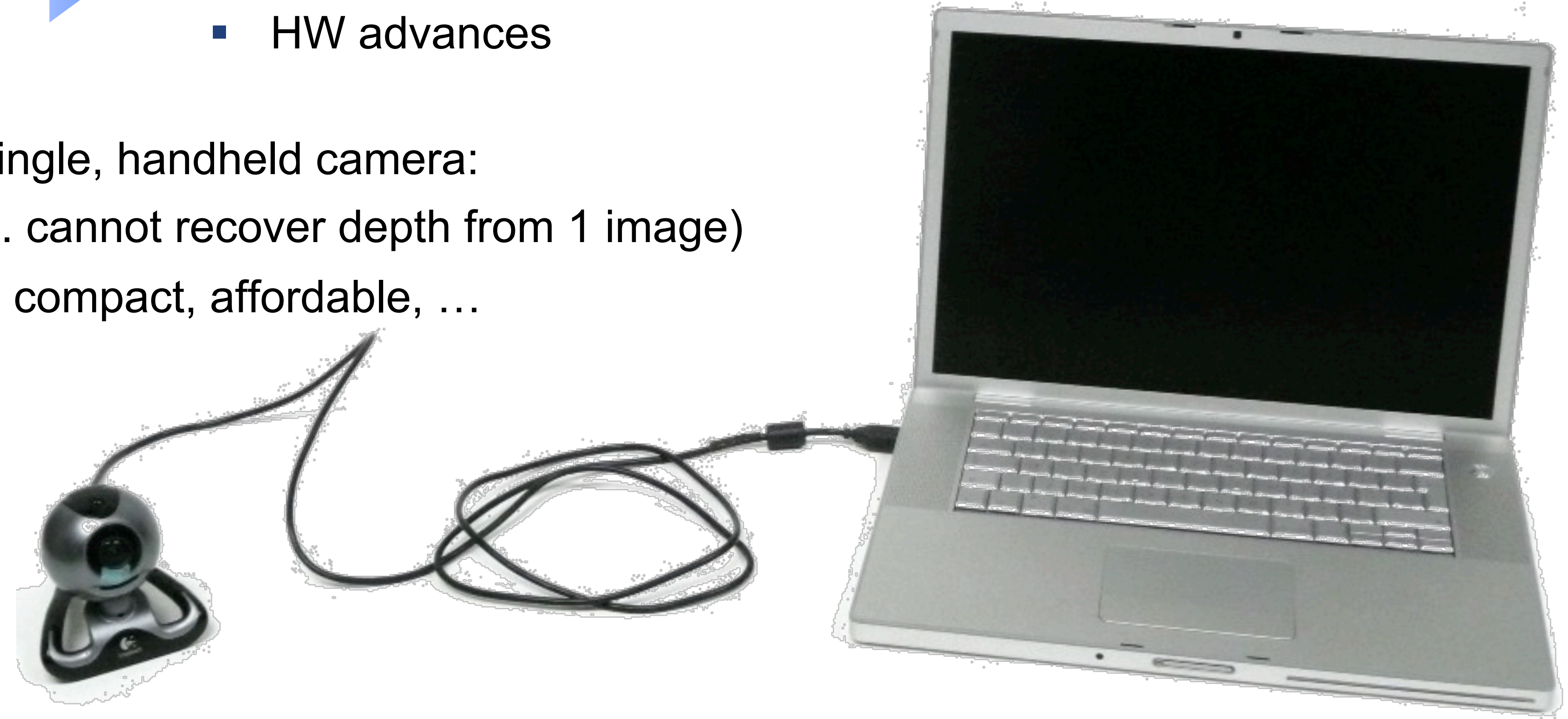


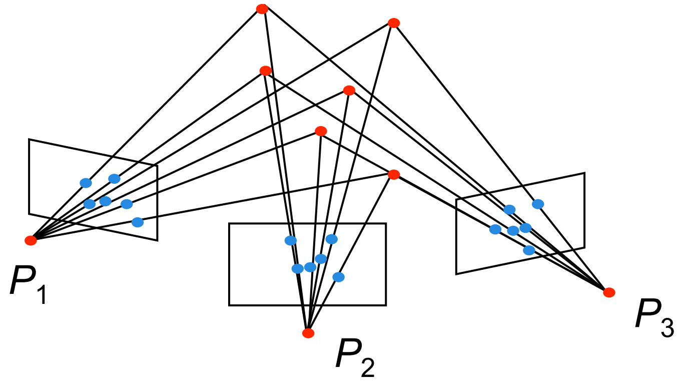
Image Courtesy of G. Klein

MonoSLAM | from SFM to SLAM

Structure from Motion (SFM):

- Take some images of the object/scene to reconstruct
- Features (points, lines, …) are extracted from all frames and matched among them
- Process all images simultaneously
- Optimization to recover both:
  - camera motion and
  - 3D structure

  up to a scale factor
- **Not real-time**



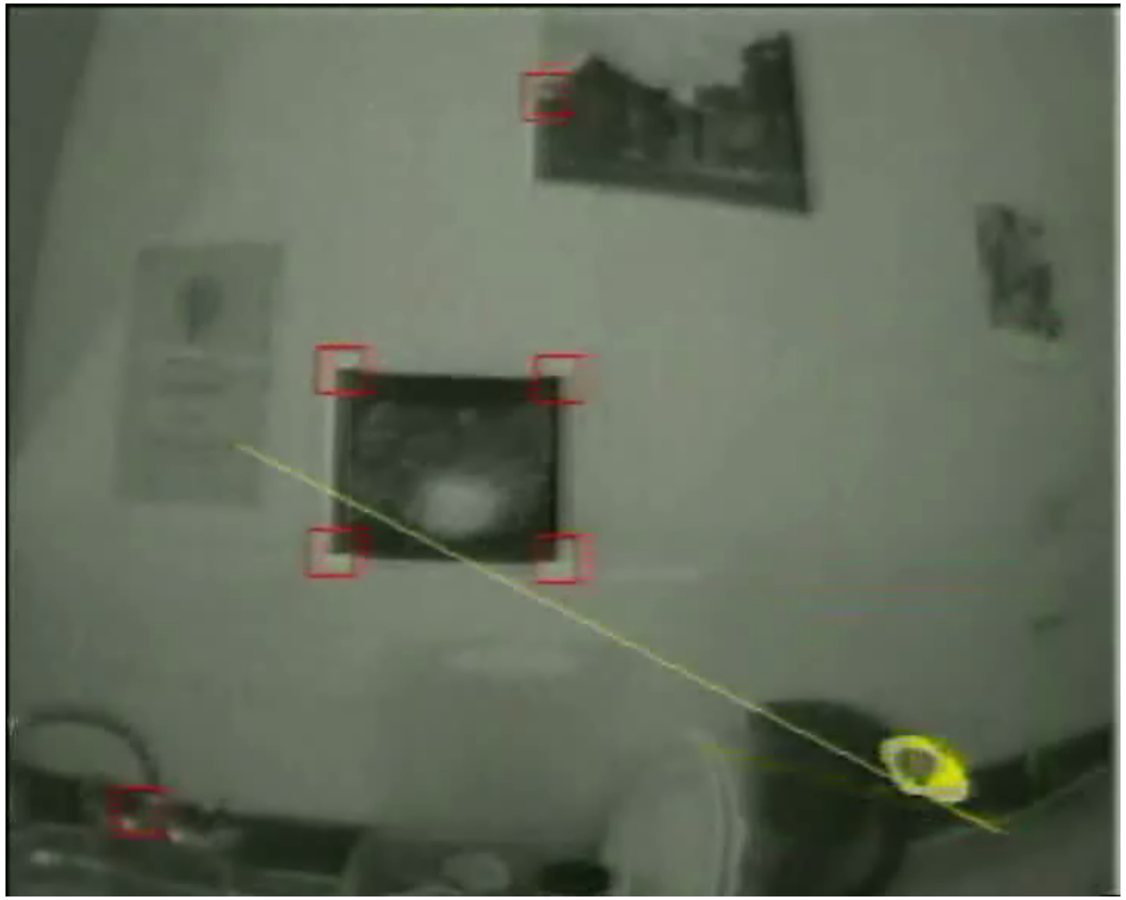
[Agarwal et al, IEEE Computer 2010]

San Marco square, Venice
14,079 images, 4,515,157 points

MonoSLAM | problem statement

- Can we track the motion of a **hand-held** camera while it is moving? i.e. **online**

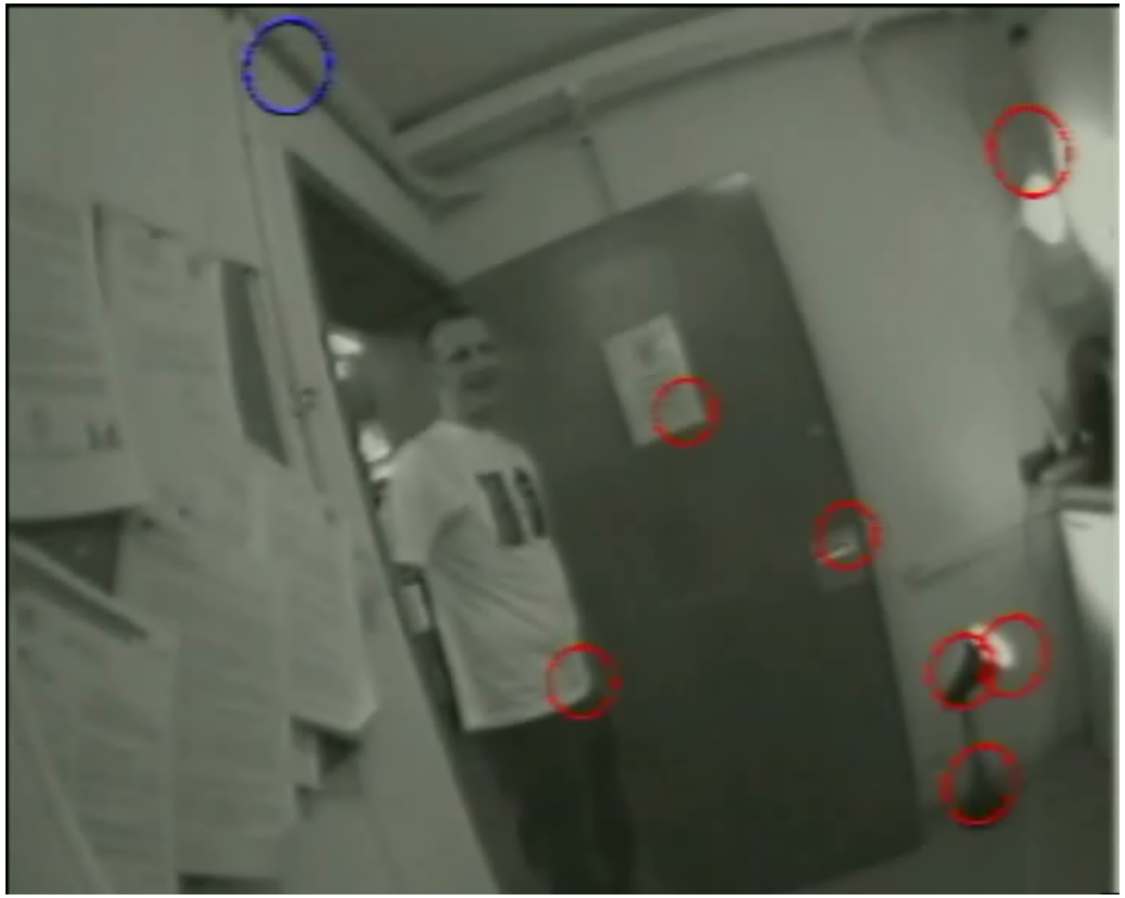
scene view

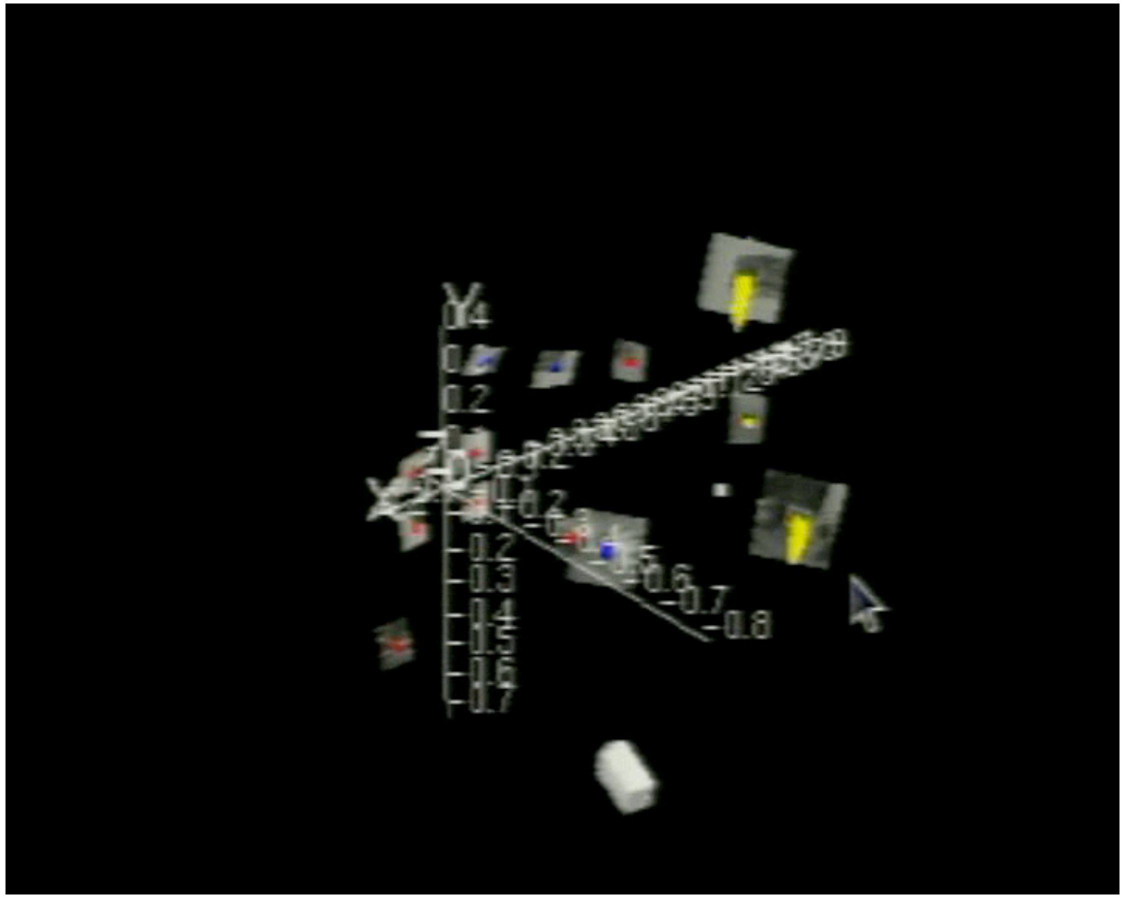

camera view

The videos are courtesy of Andrew J. Davison

MonoSLAM | problem statement

- SLAM using a single camera, grabbing frames at 30Hz
- Ellipses (in camera view) and Ellipsoids (in map view) represent uncertainty


camera view


internal SLAM map

The videos are courtesy of Andrew J. Davison

MonoSLAM | representation of the world

- The belief about the state of the world $\mathbf{x}$ is approximated with a single, multivariate Gaussian distribution:

$$p(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} |\mathrm{P}|^{-\frac{1}{2}} \, exp\{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^\top \mathrm{P}^{-1}(\mathbf{x}-\hat{\mathbf{x}})\}$$

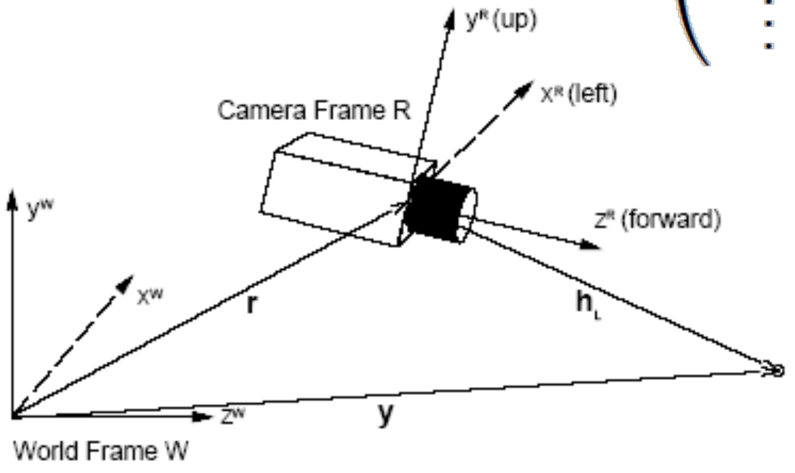
d denotes the dimension of $\hat{\mathbf{x}}$ and P is a square $(d \times d)$ matrix.

Landmark's state
e.g. 3D position for point-features

Mean
(state vector)

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_c \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix}, \quad \mathrm{P} = \begin{bmatrix} \mathrm{P}_{xx} & \mathrm{P}_{xy_1} & \mathrm{P}_{xy_2} & \cdots \\ \mathrm{P}_{y_1x} & \mathrm{P}_{y_1y_1} & \mathrm{P}_{y_1y_2} & \cdots \\ \mathrm{P}_{y_2x} & \mathrm{P}_{y_2y_1} & \mathrm{P}_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \end{bmatrix}$$

Covariance matrix



Camera state

$$\mathbf{x}_c = \begin{pmatrix} \mathbf{r}^W \\ \mathbf{q}^{CW} \\ \mathbf{v}^W \\ \omega^C \end{pmatrix}$$

: Position [3 dim.]
: Orientation using quaternions [4 dim.]
: Linear velocity [3 dim.]
: Angular velocity [3 dim.]

MonoSLAM | motion & probabilistic prediction

- How has the camera moved from frame $t\text{-}1$ to frame t ?

$$\hat{x}_t = f(x_{t-1}, u_t)$$

$$\hat{P}_t = F_y P_{t-1} F_y^T + F_u Q_t F_u^T$$

- The camera is **hand-held** ⇨ no odometry or control input data

⇨ Use a motion model

- But how can we model the unknown intentions of a human carrier?

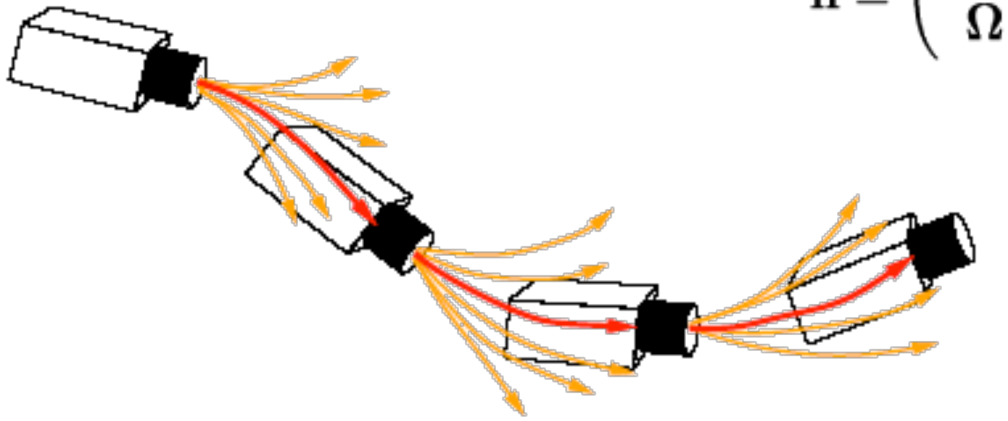
"we assume that the camera moves at a constant velocity over all time , [...] but on average we expect undetermined accelerations occur with a Gaussian profile"

- Davison et al. use a **constant linear velocity, constant angular velocity motion model**:

MonoSLAM | constant linear & angular velocity model

At each time step, the unknown linear **a** and angular ***α*** accelerations (characterized by zero-mean Gaussian distribution) cause an impulse of velocity:

$$\mathbf{n} = \begin{pmatrix} \mathbf{V}^W \\ \mathbf{\Omega}^W \end{pmatrix} = \begin{pmatrix} \mathbf{a}^W \Delta t \\ \alpha^W \Delta t \end{pmatrix}$$

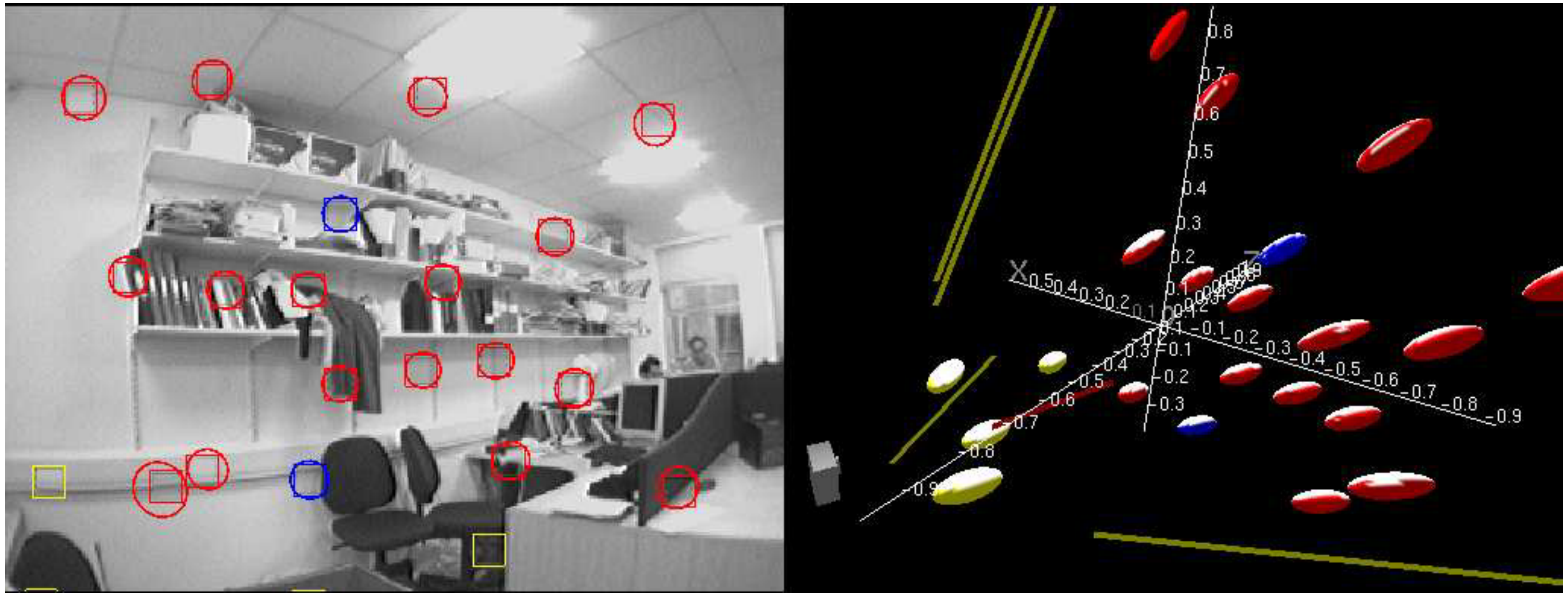


The constant velocity motion model, imposes a certain **smoothness** on the camera motion expected.

$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}^W_{new} \\ \mathbf{q}^{WR}_{new} \\ \mathbf{v}^W_{new} \\ \omega^W_{new} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^W + \mathbf{\Omega}^W)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ \omega^W + \mathbf{\Omega}^W \end{pmatrix}$$

MonoSLAM | motion & probabilistic prediction

- Based on the predicted new camera pose ⇨ predict **which** known features will be visible and **where** they will lie in the image
- Use measurement model h to identify the predicted location $\hat{z}_i = h_i(\hat{x}_t, y_i)$ of each feature and an associated search region (defined in the corresponding diagonal block of $\Sigma_{IN} = H\hat{P}_t H^T + R$)
- Essentially: project the 3D ellipsoids from the SLAM map onto the image space



MonoSLAM | measurement & EKF update steps

- Search for the known feature-patches inside their corresponding search regions to get the set of all observations

- Update the state using the EKF equations

$$x_t = \hat{x}_t + K_t(z_{0:n-1} - h_{0:n-1}(\hat{x}_t, y_{0:n-1}))$$

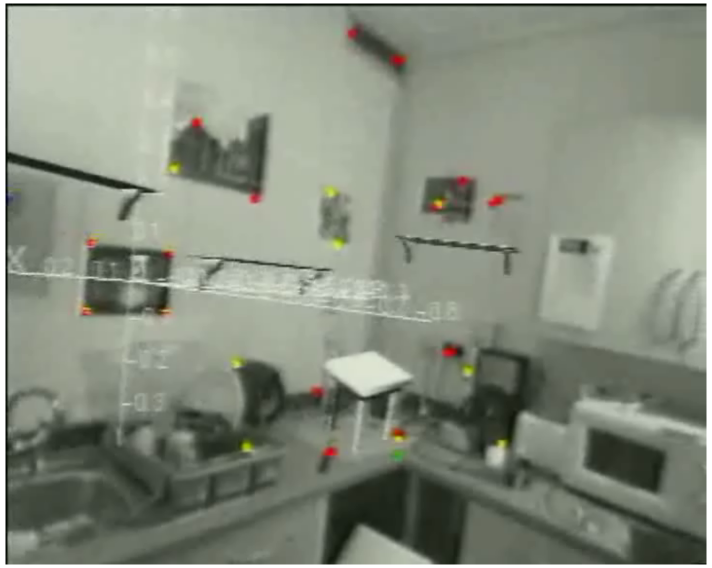
$$P_t = \hat{P}_t - K_t \Sigma_{IN} K_t^T$$

where: $$\Sigma_{IN} = H\hat{P}_t H^T + R$$

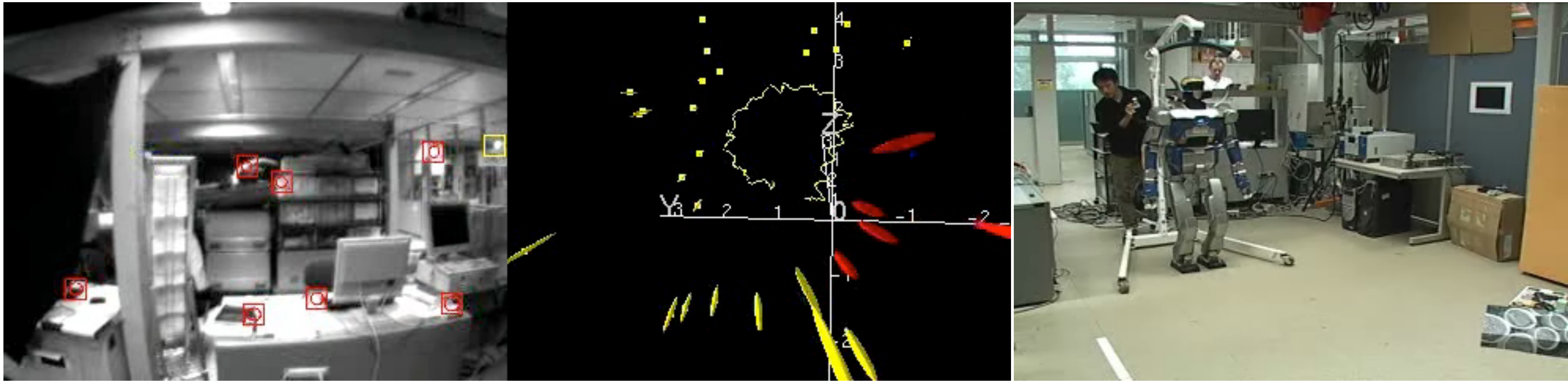
$$K_t = \hat{P}_t H (\Sigma_{IN})^{-1}$$

MonoSLAM | applications

- MonoSLAM for Augmented Reality
- HPR-2 Humanoid at JRL, AIST, Japan



The videos are courtesy of Andrew J. Davison



EKF SLAM | a note on correlations

- At start up: the robot makes the first measurements and the covariance matrix is populated assuming that these (initial) features are uncorrelated ⇨ off-diagonal elements are zero.

$$P_0 = \begin{bmatrix} P_{xx} & 0 & 0 & \dots & 0 & 0 \\ 0 & P_{y_0 y_0} & 0 & \dots & 0 & 0 \\ 0 & 0 & P_{y_1 y_1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & P_{y_{n-2} y_{n-2}} & 0 \\ 0 & 0 & 0 & \dots & 0 & P_{y_{n-1} y_{n-1}} \end{bmatrix}$$

- When the robot starts moving & taking new measurements, both the robot pose and features start becoming correlated.

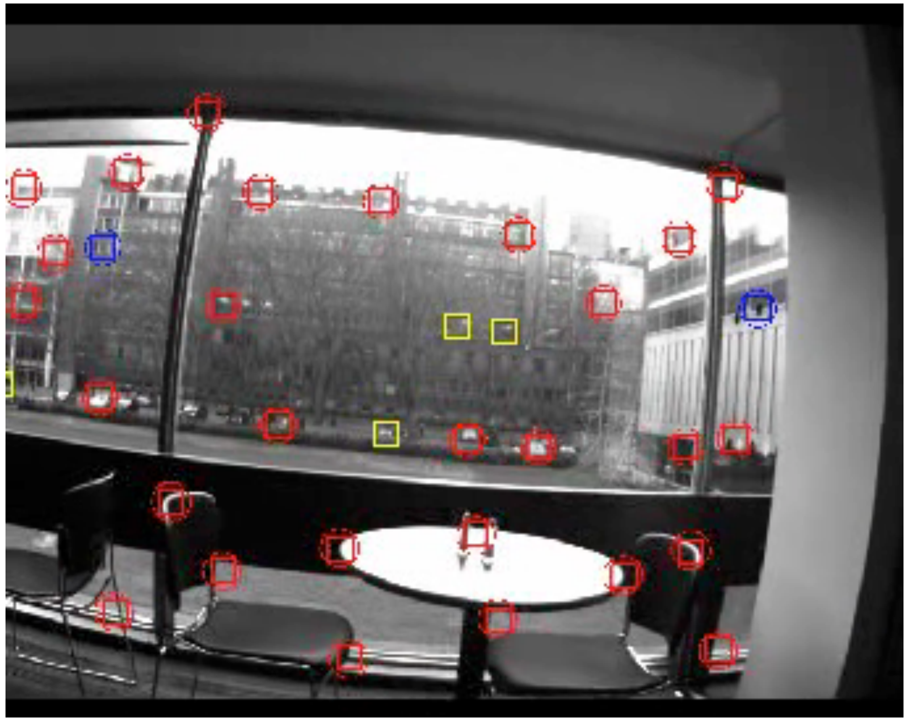
$$\hat{P}_{y_t} = F_y P_{y_{t-1}} F_y^{\ T} + F_u Q_t F_u^{\ T}$$

- Accordingly, the covariance matrix becomes **dense**.

EKF SLAM | a note on correlations

- Correlations arise as
  - the uncertainty in the robot pose is used to obtain the uncertainty of the observed features.
  - the feature measurements are used to update the robot pose.

Chli & Davison, ICRA 2009



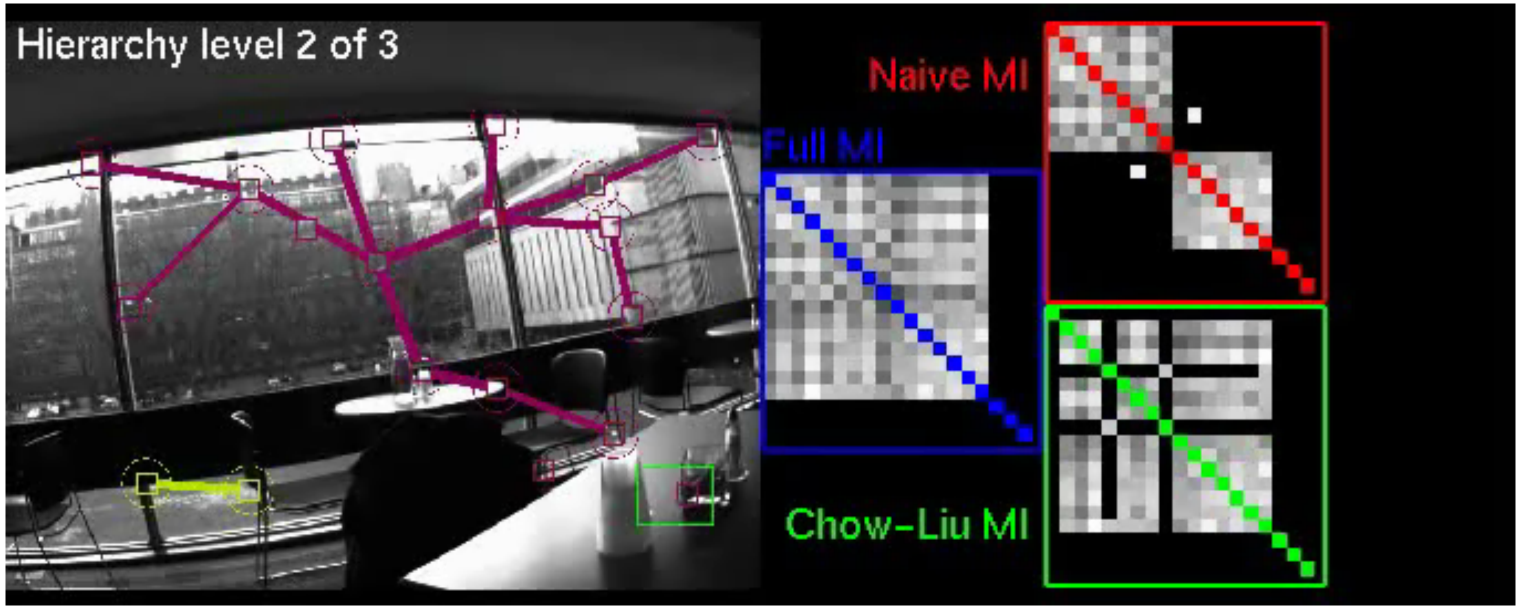
- **Regularly covisible** features become correlated and when their motion is **coherent**, their correlation is even stronger

- Correlations very important for **convergence**: The more observations are made, the more the correlations between the features will grow, the better the solution to SLAM.

EKF SLAM | drawbacks

- The state vector in EKF SLAM is much larger than the state vector in EKF localization
- Newly observed features are added to the state vector ➪ The covariance matrix **grows quadratically** with the no. features ➪ **computationally expensive for large-scale SLAM**.
- Approach to attack this: sparsify the structure of the covariance matrix (via approximations)

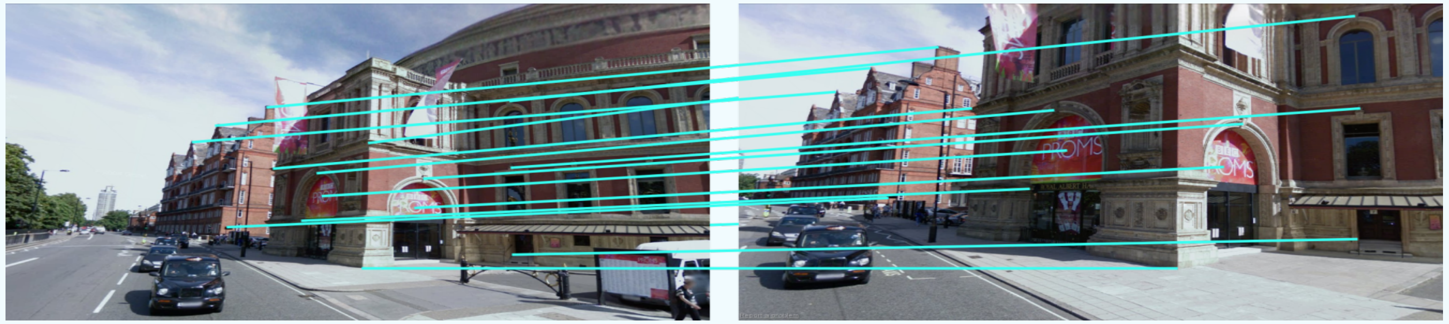
Chli & Davison, ICRA 2009



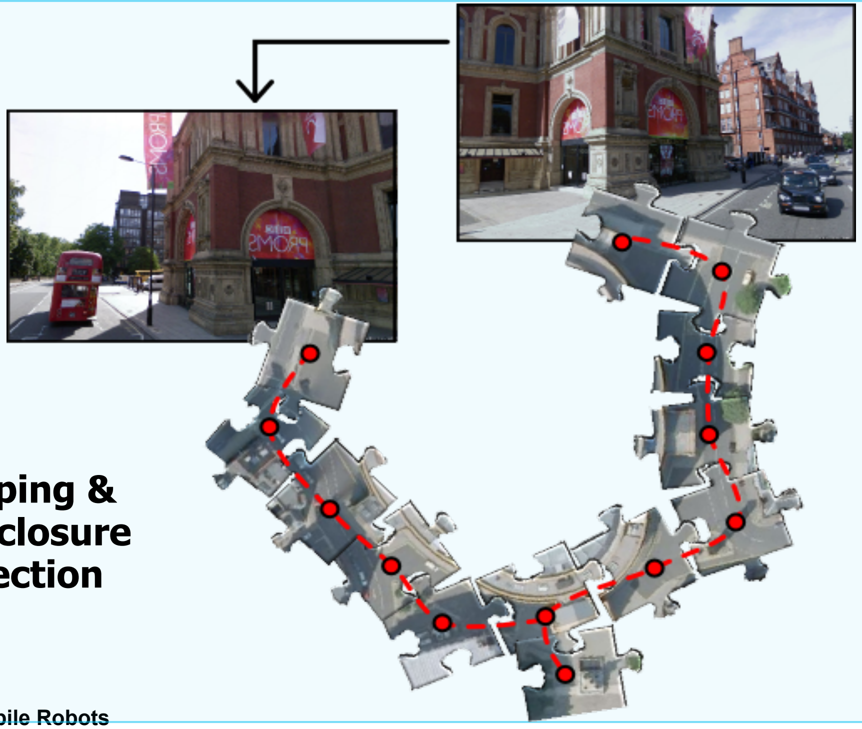
SLAM Challenges | components for scalable SLAM

1

Robust local motion estimation



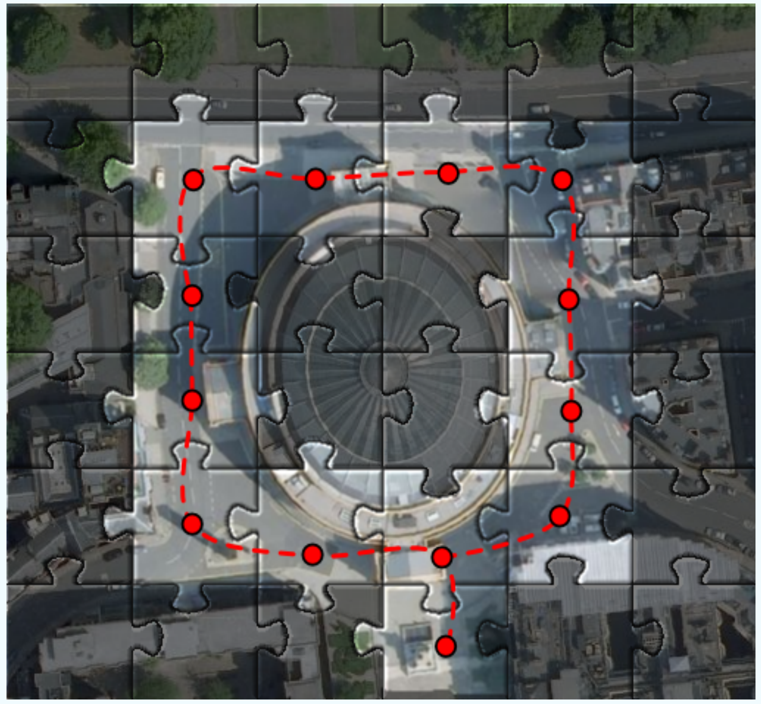
2



Mapping & loop-closure detection

3

Map management & optimisation



[Chli, PhD Thesis, 2009]