Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab

SLAM: Simultaneous Localization And Mapping

Margarita Chli

http://margaritachli.com

Today's lecture

- SLAM: One of the fundamental problems in robotics
- Focus on single-camera SLAM
- EKF SLAM via the MonoSLAM system
- State of the art systems and Current challenges

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 on board sensors only?

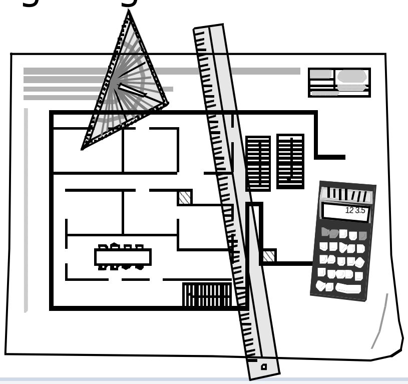
- When is SLAM necessary?
  - When a robot must be truly **autonomous** (no human input)
  - When there is **no prior** knowledge about the environment
  - When we cannot place **beacons** (also in GPS-denied environments)
  - When the robot needs to know where it is

Simultaneous Localization and Mapping

- SLAM: one of the greatest challenges in probabilistic robotics

  - More difficult than **pure localization**: the map is unknown and has to be estimated along the way.

By hand: hard & costly
(e.g. large environment)



Automatic Map Building:

More challenging, but:

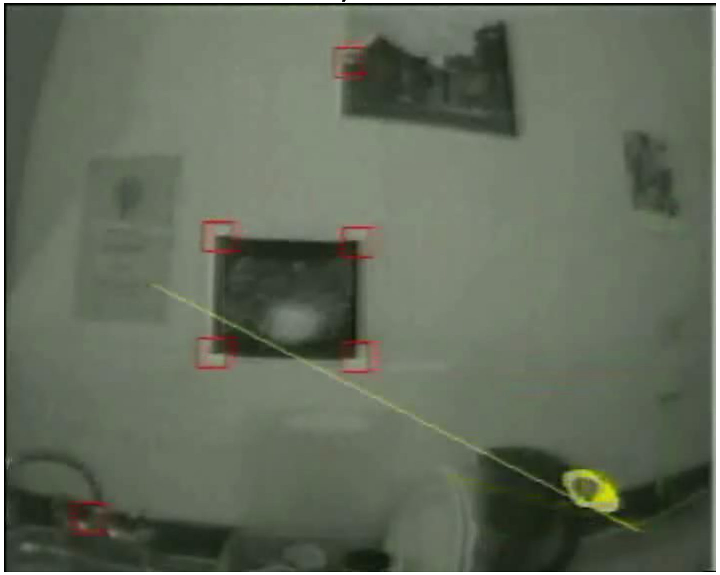
- ✓ Automatic
- ✓ The robot learns its environment
- ✓ Can adapt to dynamic changes

  - More difficult than **mapping with known poses**: the poses are unknown and have to be estimated along the way.

Features for SLAM

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

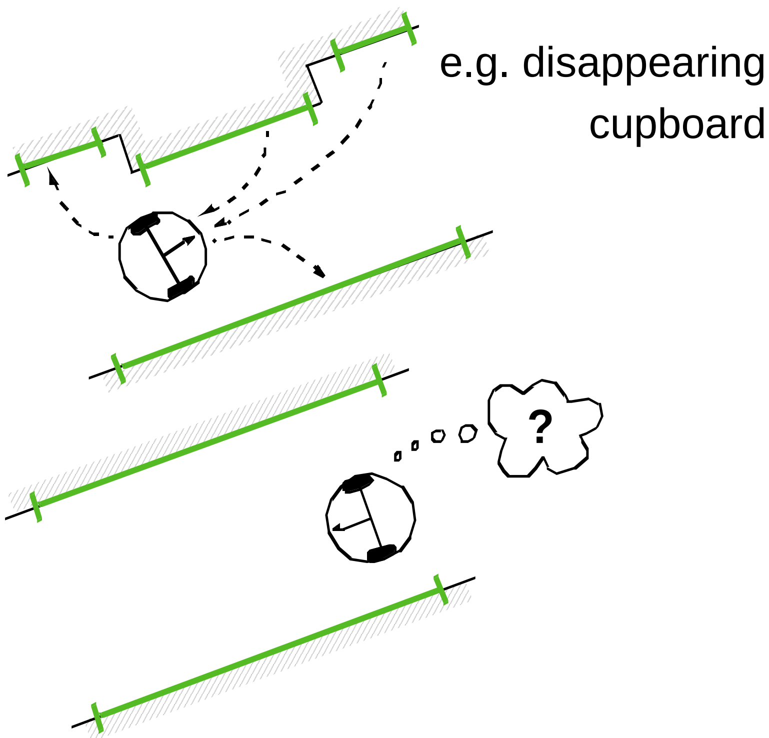
Courtesy of Andrew J. Davison



- 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

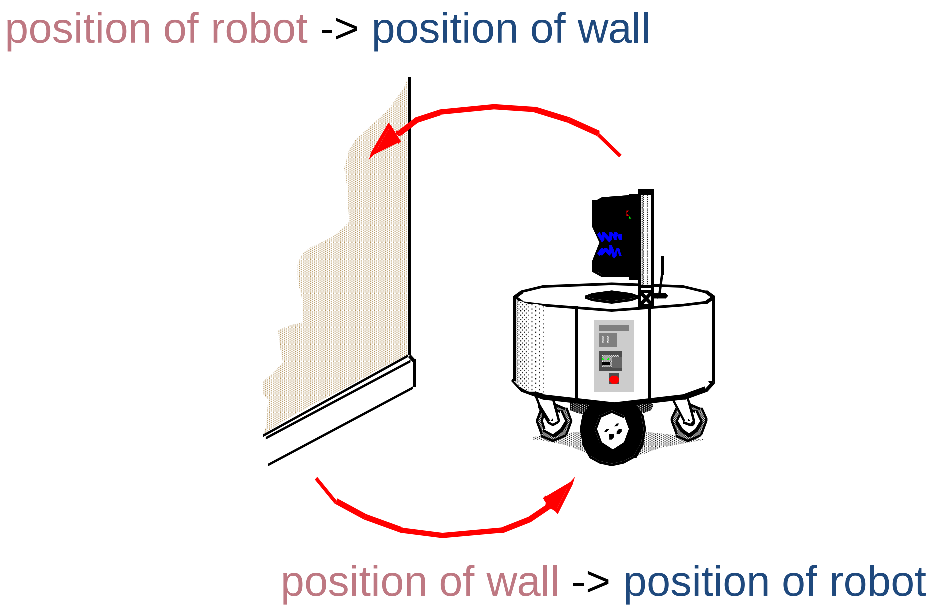
Map Building

1. Map Maintenance: Keeping track of changes in the environment



- e.g. measure of belief of each environment feature
- Generally assume static environment

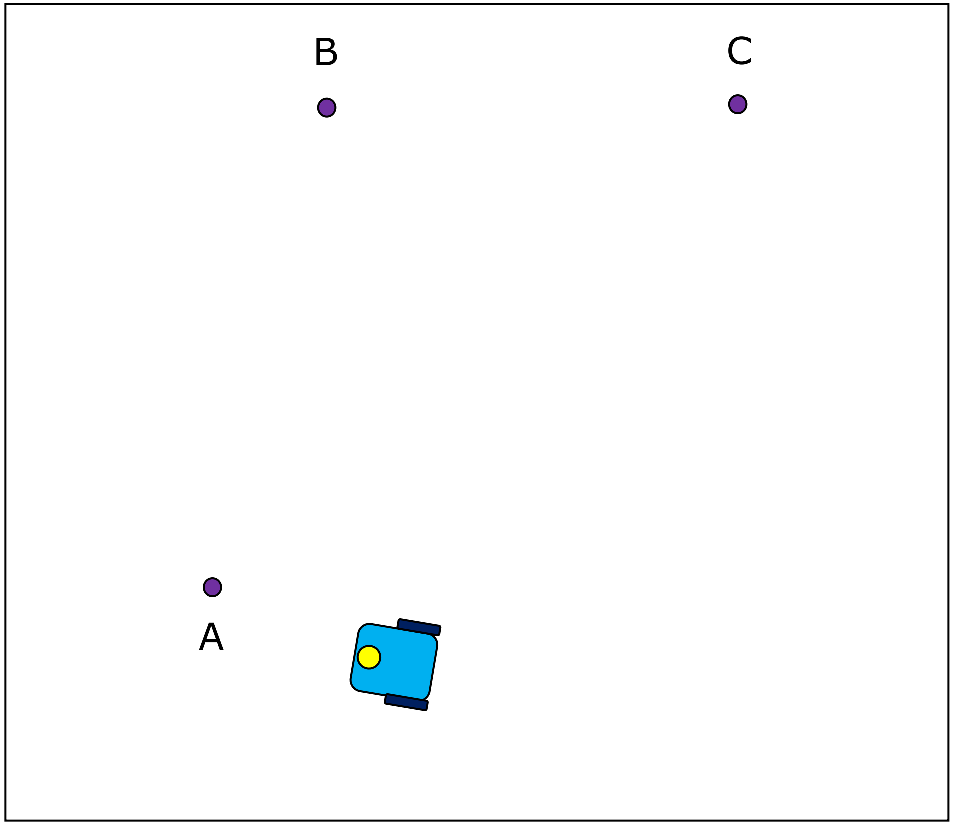
2. Representing and Propagating Uncertainty



- probability densities for feature positions
- Map can become inconsistent due to erroneous measurements / motion drift

How to do SLAM

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

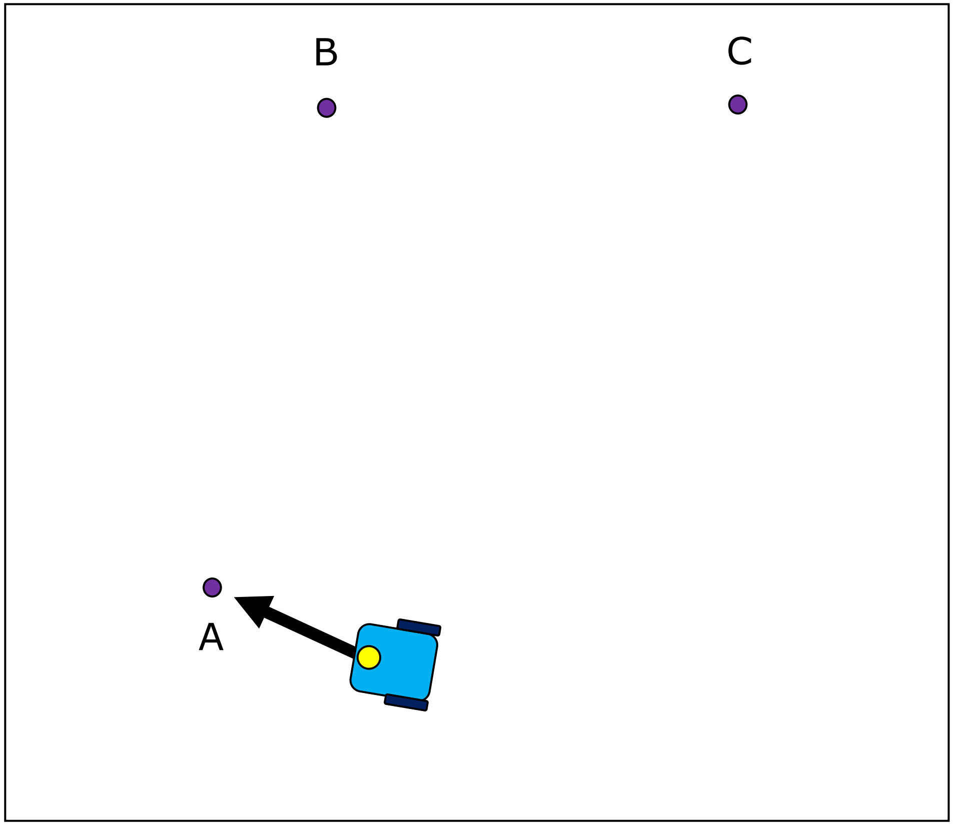


Start: robot has zero uncertainty

How to do SLAM

On every frame:

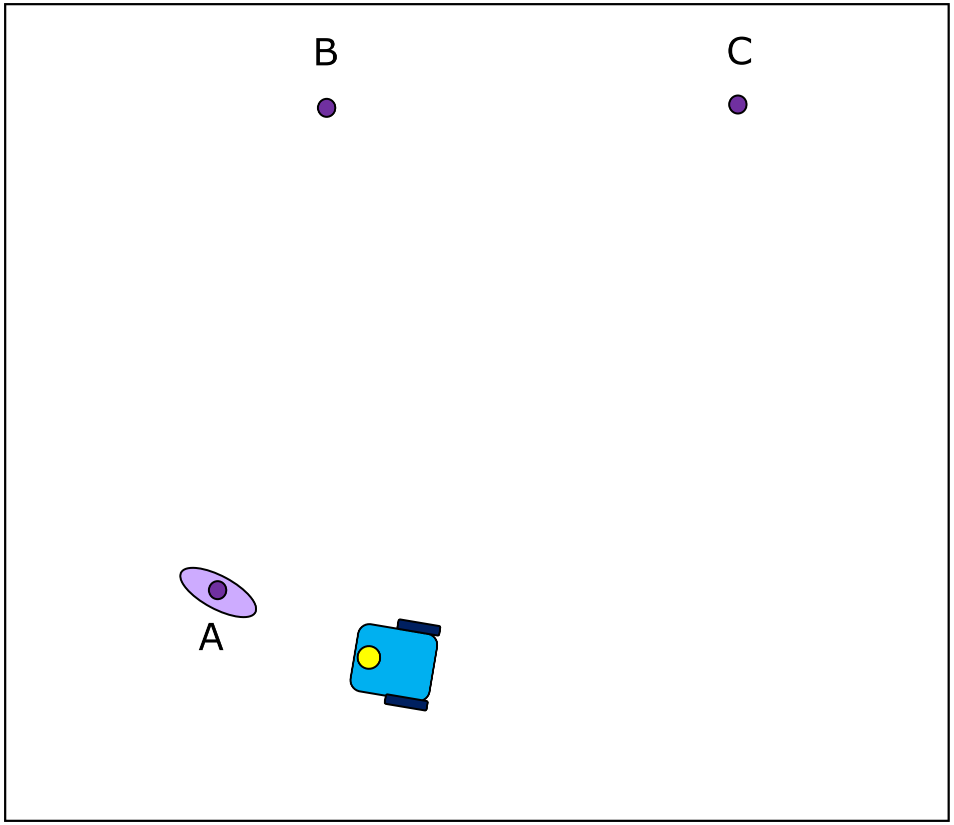
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



First measurement of feature A

How to do SLAM

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

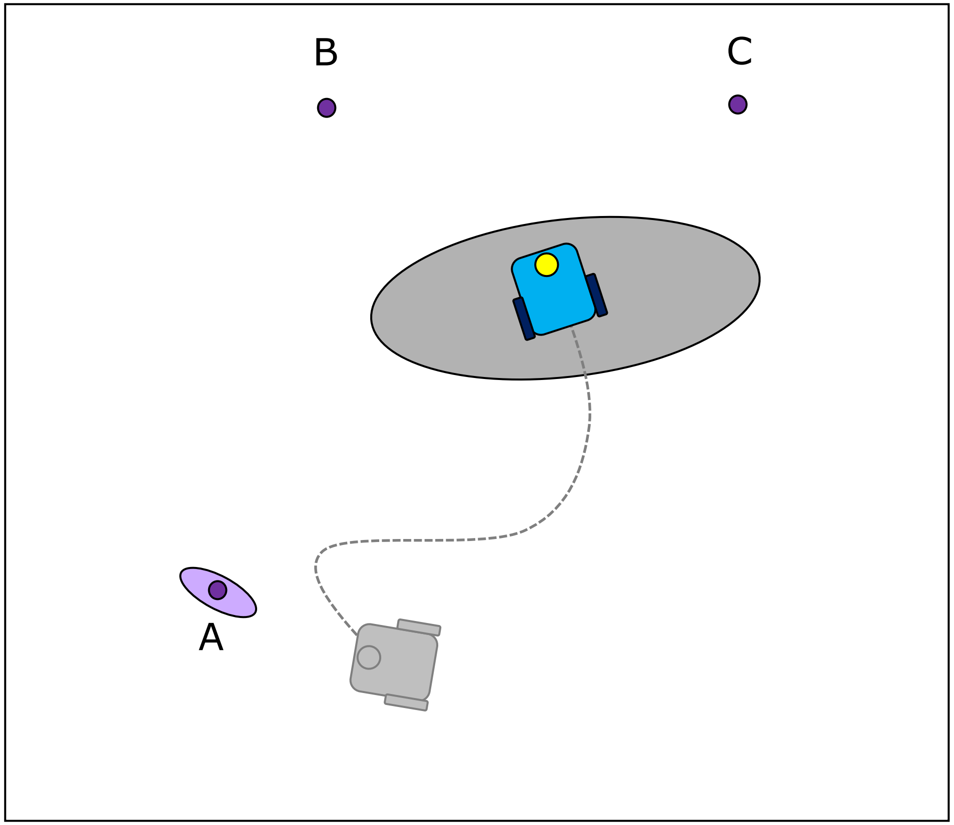


On every frame:

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

How to do SLAM

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

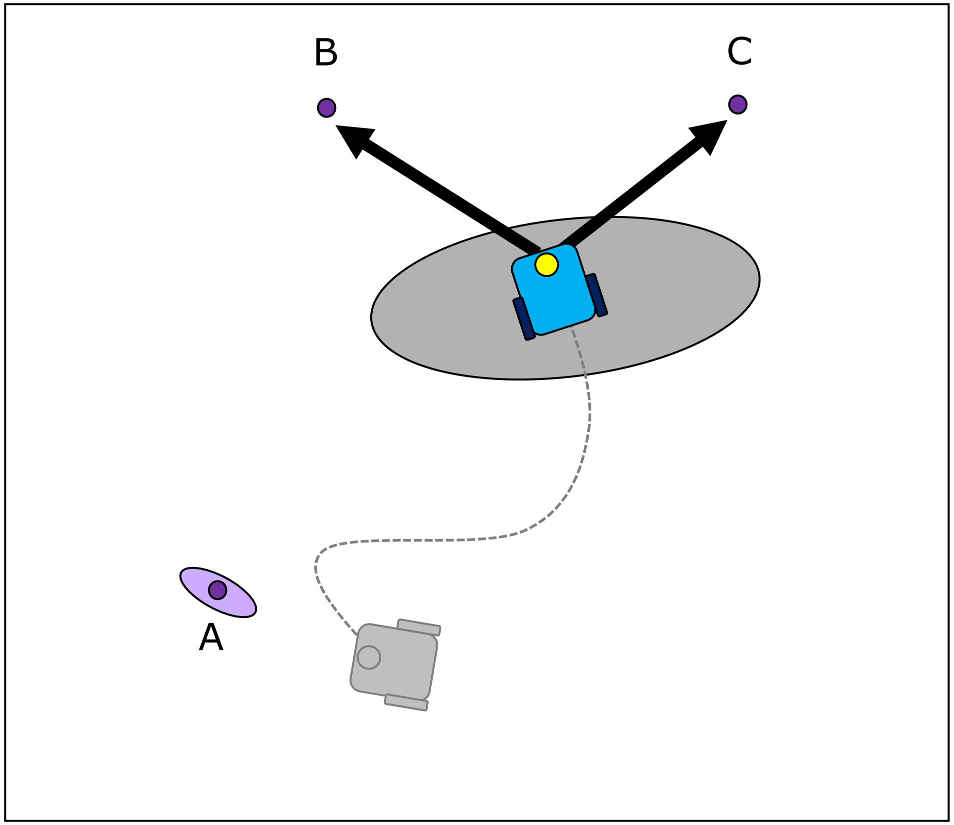


Robot moves forwards: uncertainty grows

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

How to do SLAM

- Robot observes two new features.



Robot makes first measurements of B & C

On every frame:

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

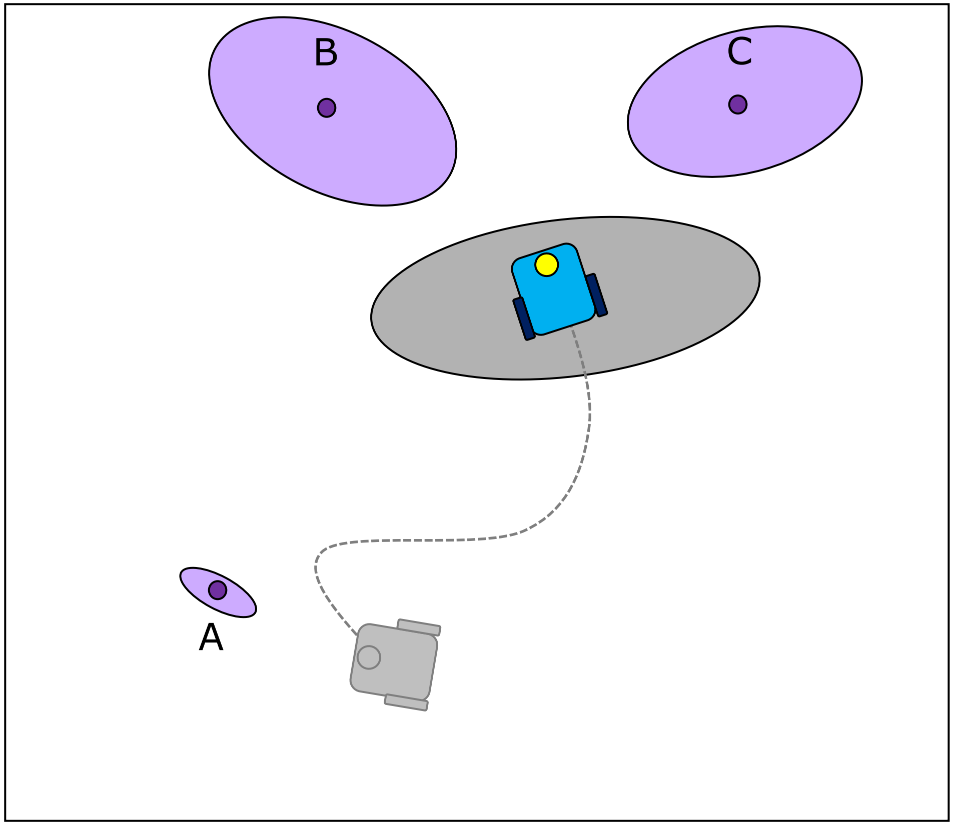
How to do SLAM

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

⇨ map becomes correlated with the robot position estimate.

On every frame:

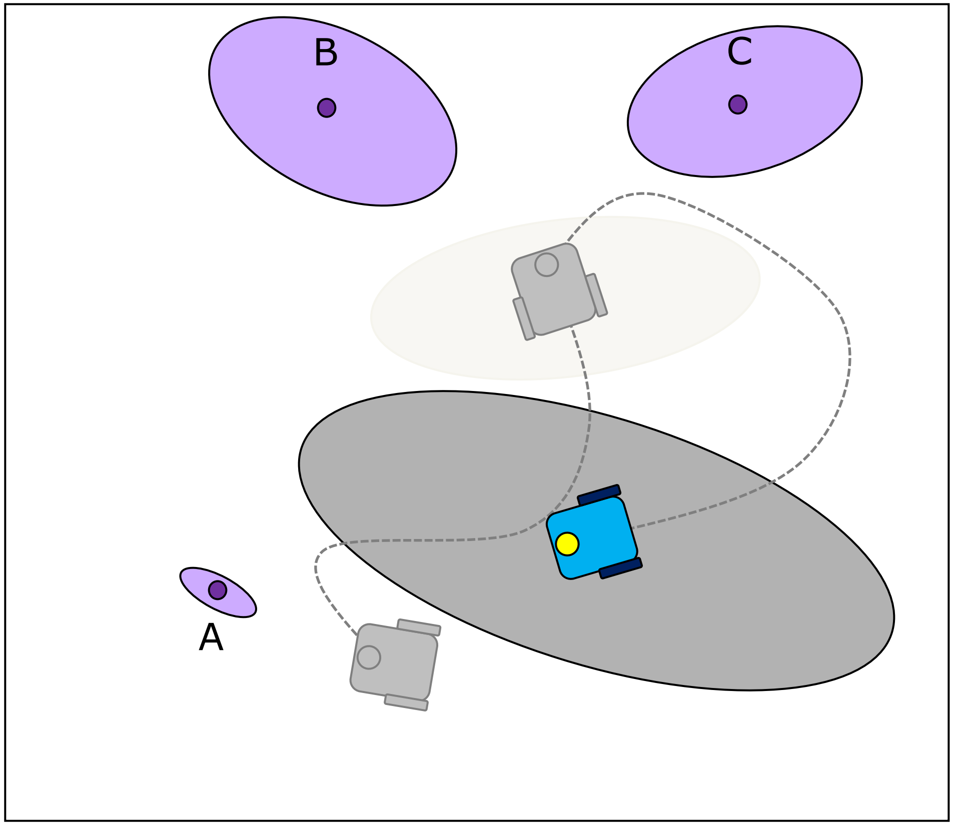
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations



Robot makes first measurements of B & C

How to do SLAM

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



Robot moves again: uncertainty grows more

On every frame:

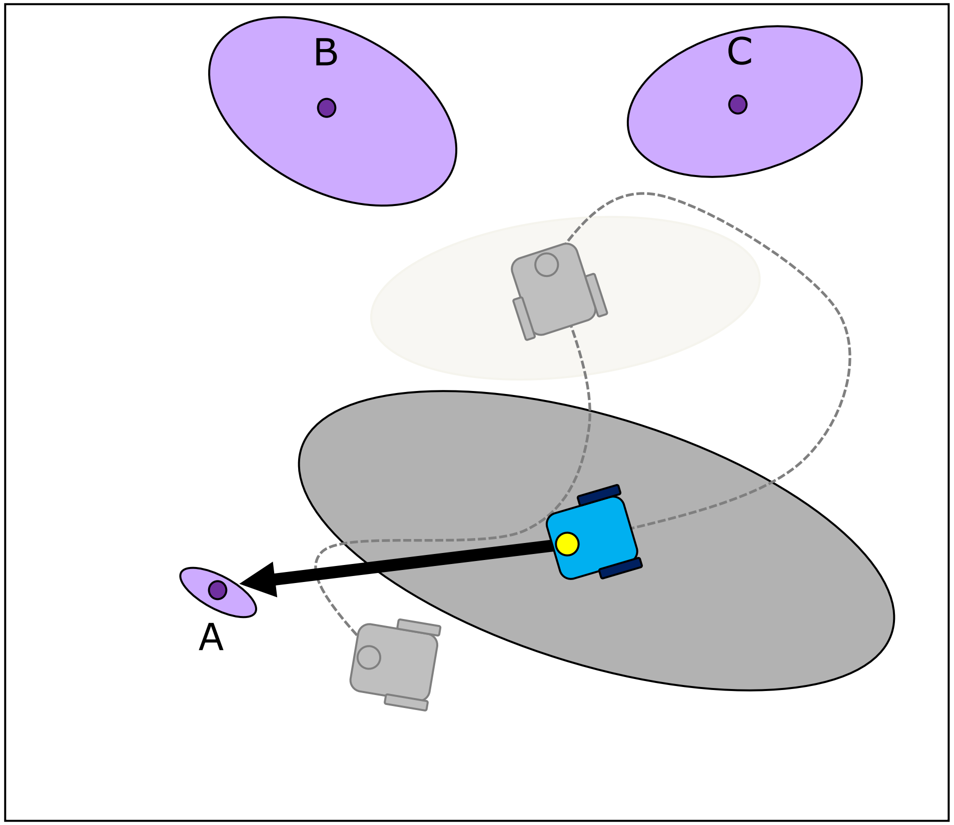
- **Predict** how the robot has moved
- **Measure**
- **Update** the internal representations

How to do SLAM

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

On every frame:

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

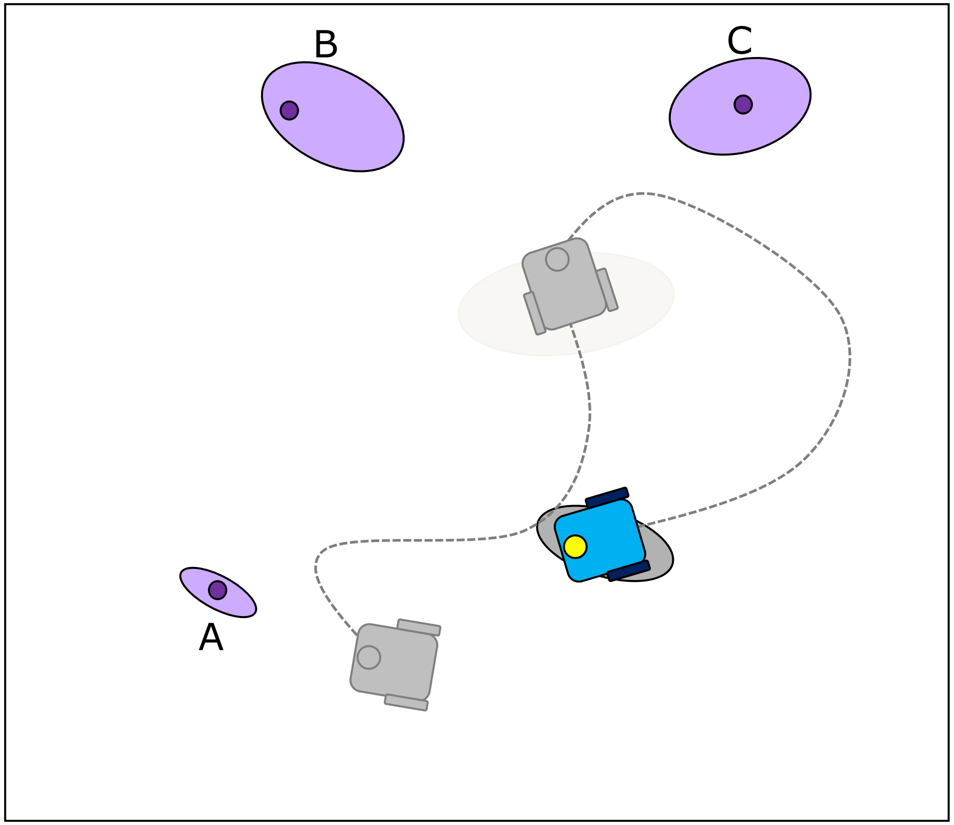


Robot re-measures A: “loop closure”

How to do SLAM

- Robot updates its position: the resulting position 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

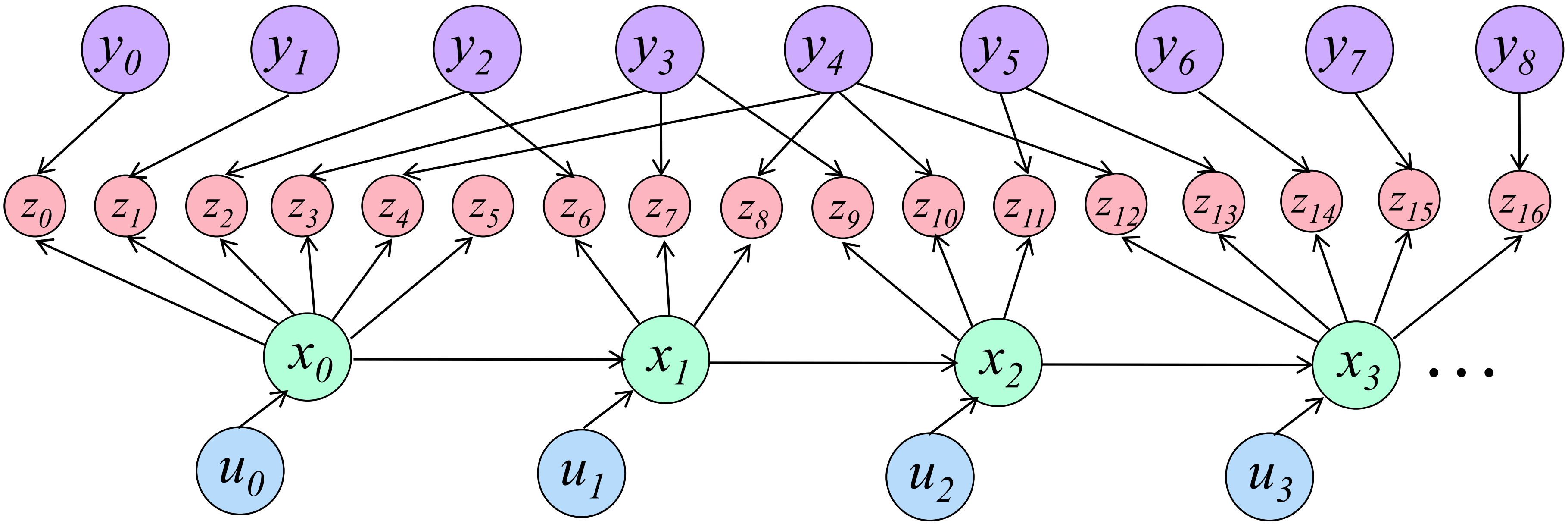


Robot re-measures A: "loop closure"
uncertainty shrinks

SLAM: Probabilistic Formulation

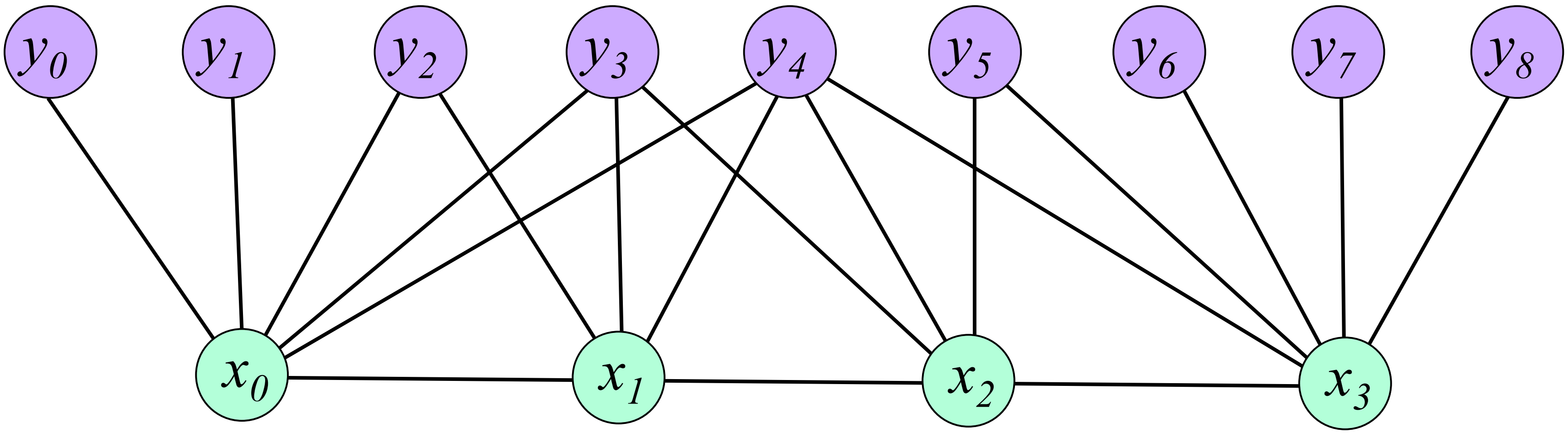
- Robot **pose** at time t : x_t ⇨ Robot **path** up to this time: $\{x_0, x_1, \ldots, x_t\}$
- Robot **motion** between time $t\text{-}1$ and t : u_t (control inputs/proprioceptive sensor readings)
  ⇨ Sequence of robot relative motions: $\{u_0, u_1, \ldots, u_t\}$
- The **true map** of the environment: $\{y_0, y_1, \ldots, y_N\}$
- 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
  $p(x_{0:t}, y_{0:n} \mid z_{0:k}, u_{0:t})$
- The Online SLAM problem: estimate the posterior
  $p(x_t, y_{0:n} \mid z_{0:k}, u_{0:t})$

SLAM: graphical representation



SLAM Approaches

- **Full graph optimization** (bundle adjustment)



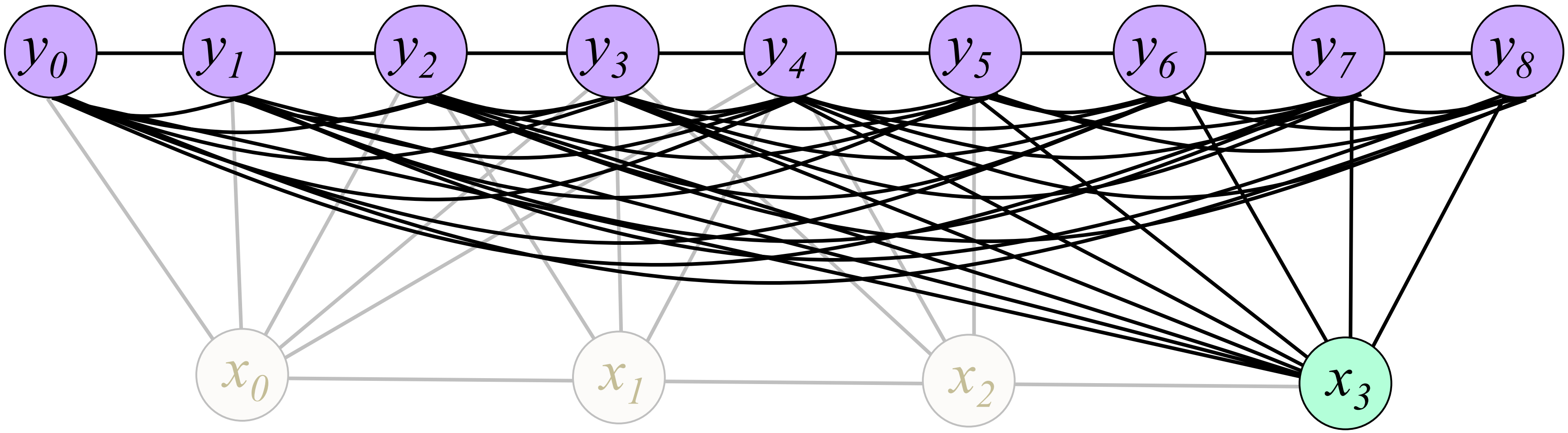
- Eliminate observations & control-input nodes and solve for the constraints between poses and landmarks.

- Globally consistent solution, but infeasible for large-scale SLAM

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

SLAM Approaches

- **Filtering**

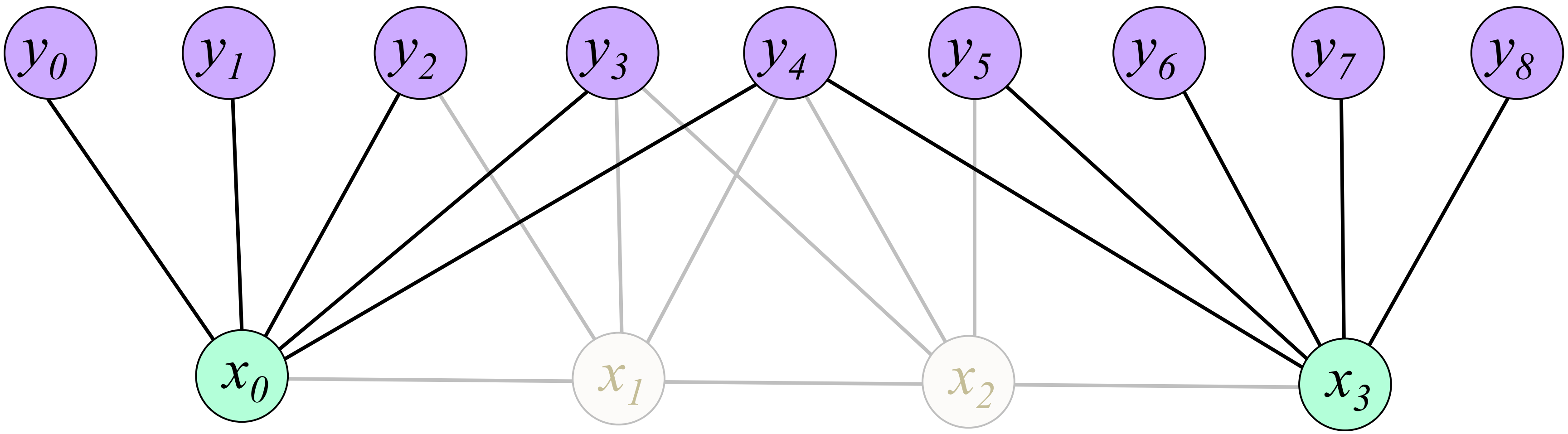


- Eliminate all past poses: 'summarize' all experience with respect to the last pose, using a **state vector** and the associated **covariance matrix**

- We're going to look at filtering in more detail…

SLAM Approaches

- **Key-frames**



- Retain the most 'representative' poses (key-frames) and their dependency links ⇨ optimize the resulting graph
- Example: PTAM [Klein & Murray, ISMAR 2007]

PTAM: Parallel Tracking and Mapping

Courtesy of Georg Klein

Example of EKF SLAM

MonoSLAM: Real-Time Single Camera SLAM

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

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, …

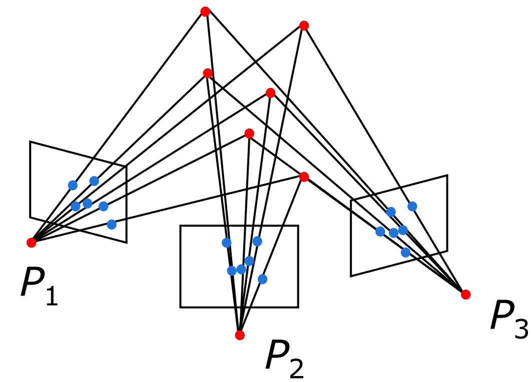
Image Courtesy of G. Klein

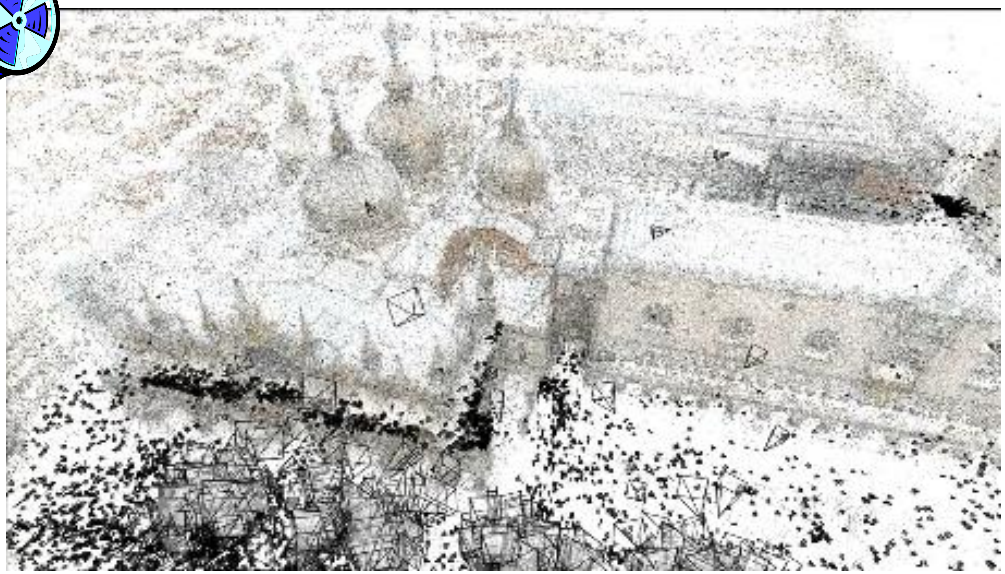
From SFM to SLAM

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
- Optimisation to recover both:
  - camera motion and
  - 3D structure

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



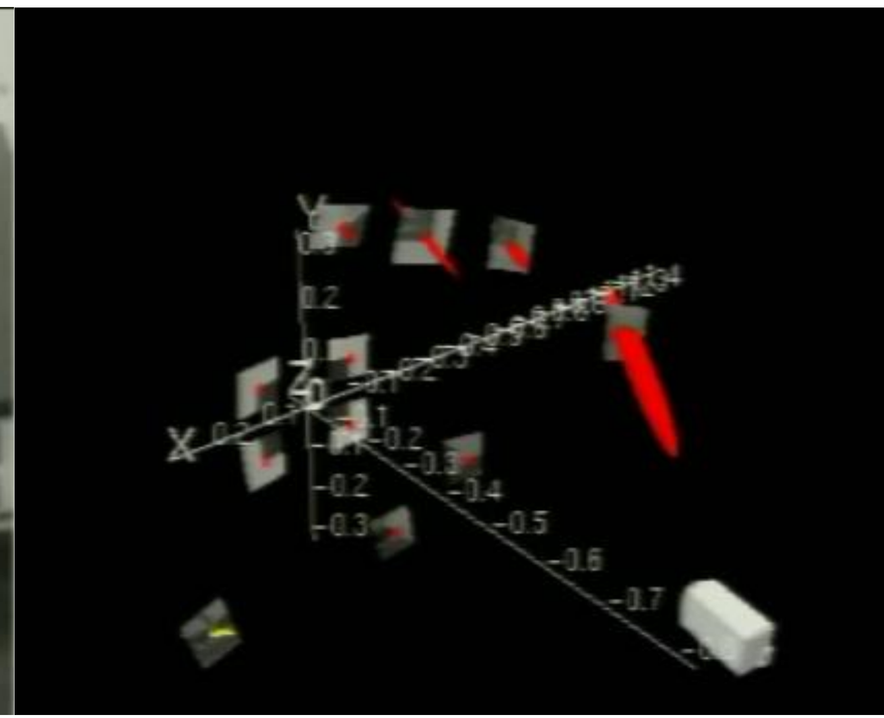


San Marco Square: 14,079 images, 4,515,157 points
[Agarwal et al., ICCV 2009]

MonoSLAM

- Can we track the motion of a **hand-held** camera while it is moving?
  i.e. **online**
- SLAM using a single camera, grabbing frames at 30Hz
- Ellipses (in camera view) and Bubbles (in map view) represent uncertainty

camera view



internal SLAM map

Representation of the world

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

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

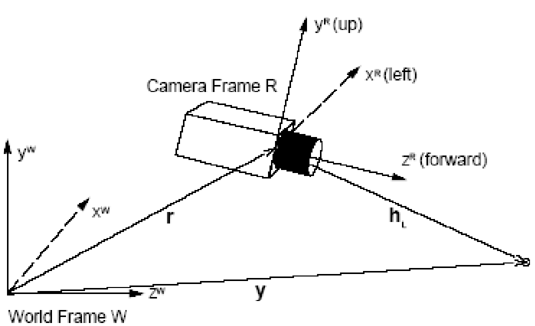
d denotes the dimension of $\hat{\mathbf{x}}$ and P is a square $(d \times d)$ matrix

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.]

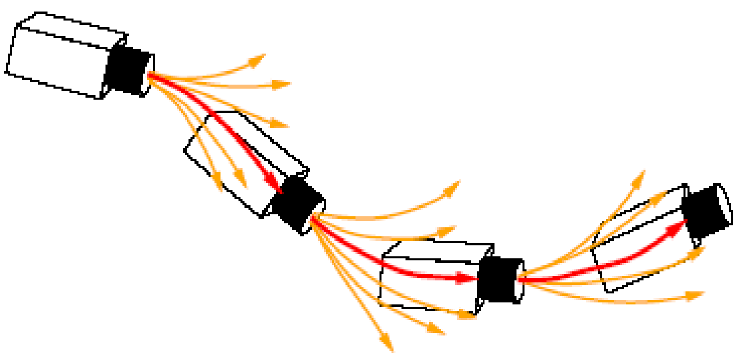
Motion & Probabilistic Prediction

- How has the camera moved from frame *t-1* to frame *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?
- Davison et al. use a **constant velocity motion model**:

"on average we expect undetermined accelerations occur with a Gaussian profile"

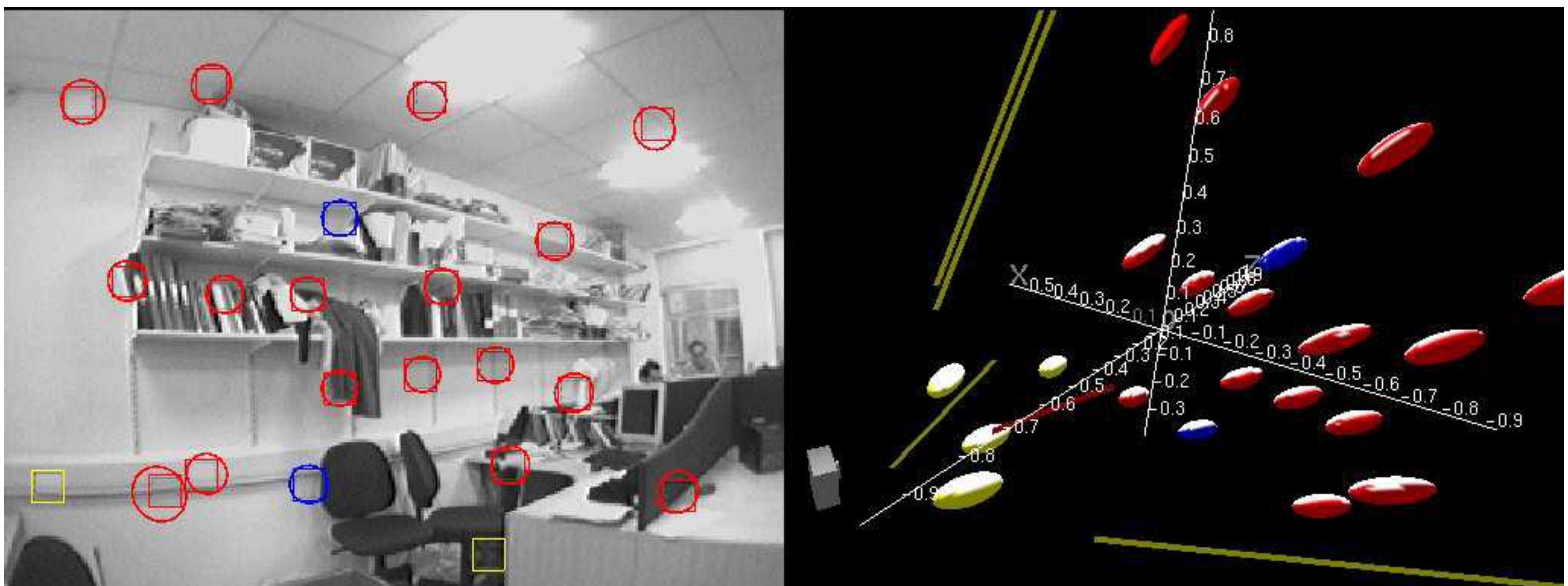


$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}^W_{new} \\ \mathbf{q}^{WR}_{new} \\ \mathbf{v}^W_{new} \\ \omega^W_{new} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^W + \mathbf{\Omega}^W)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W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

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

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 in image space



Measurement & EKF update

- 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:

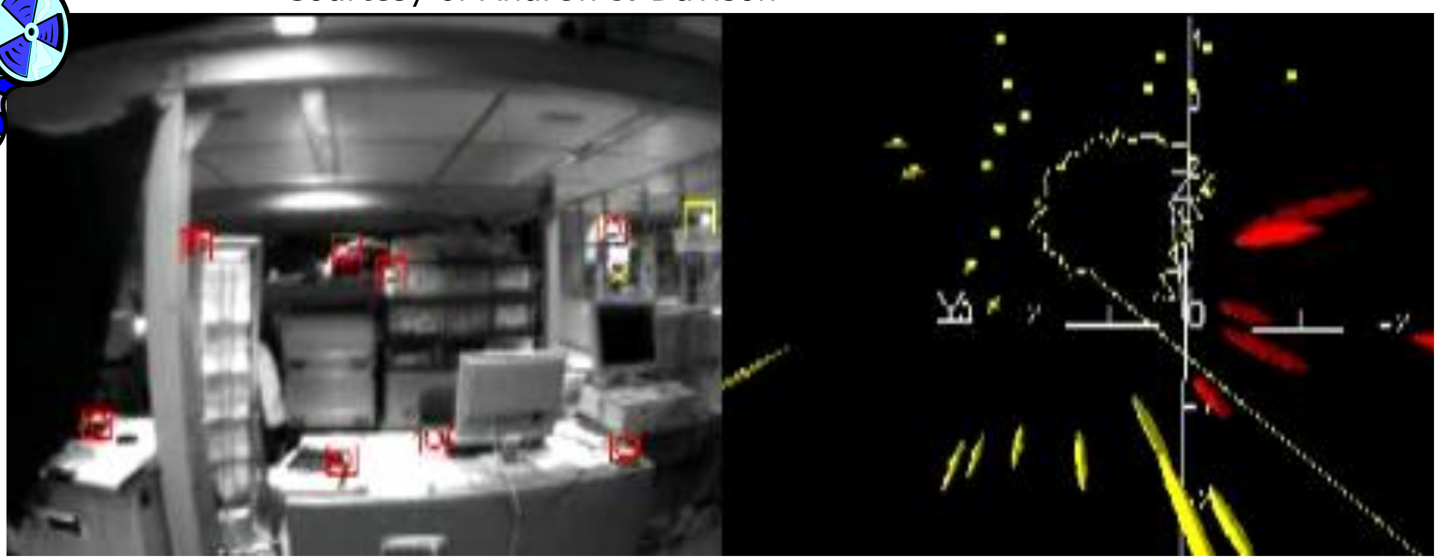
$$\Sigma_{IN} = H\hat{P}_t H^T + R$$

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

Application:
HPR-2 Humanoid at JRL, AIST, Japan

- Small circular loop within a large room
- No re-observation of 'old' features until closing of large loop

Courtesy of Andrew J. Davison



EKF SLAM: Correlations

- At start, when the robot makes the first measurements, the covariance matrix is populated by 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 & making new measurements, both the robot pose and features start becoming correlated.

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

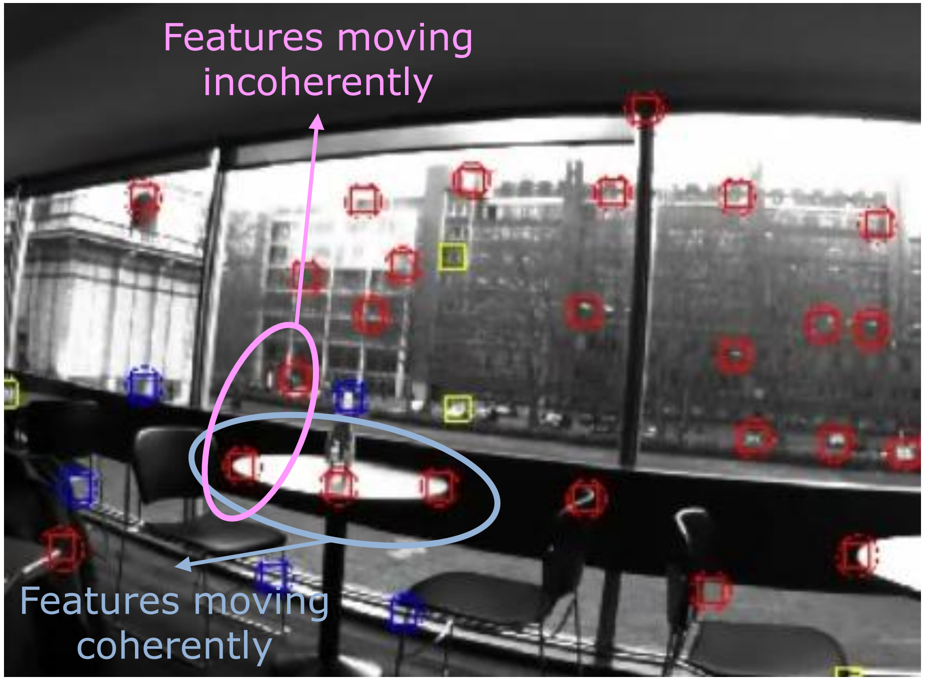
EKF SLAM: Correlations

- 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.

- **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.



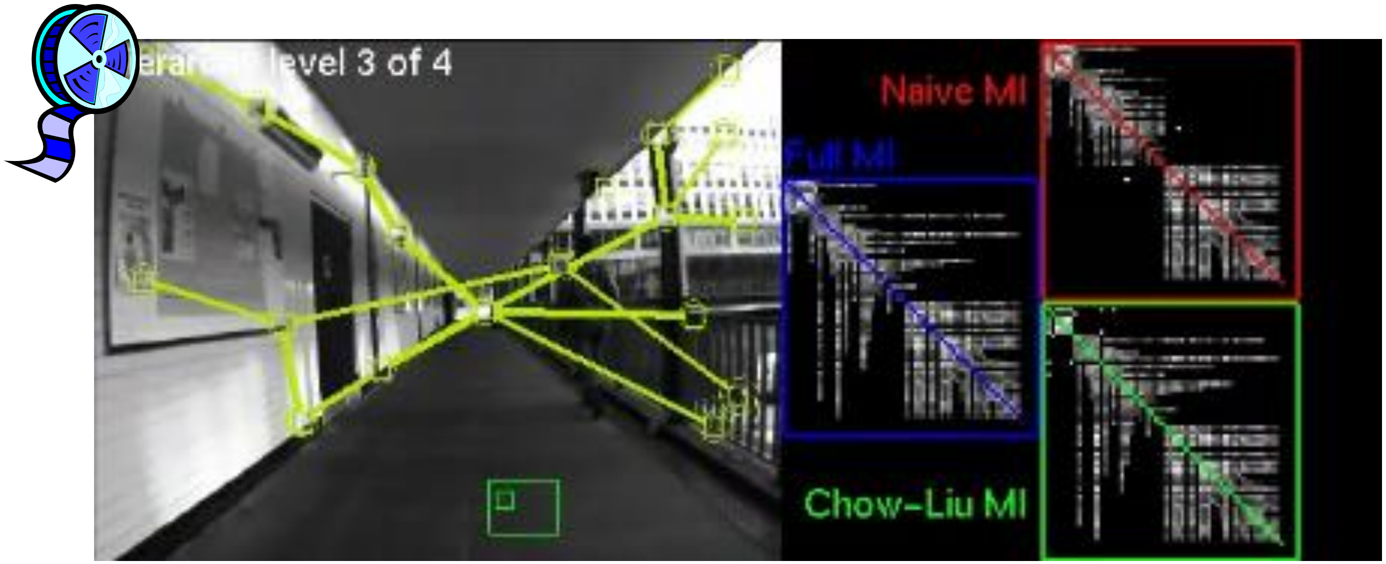
Chli & Davison, ICRA 2009

Drawbacks of EKF SLAM

- 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**.

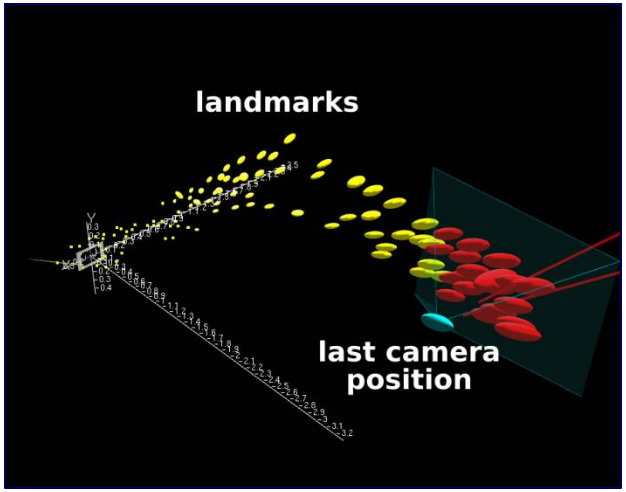
- Approach to attack this: sparsify the structure of the covariance matrix (via approximations)



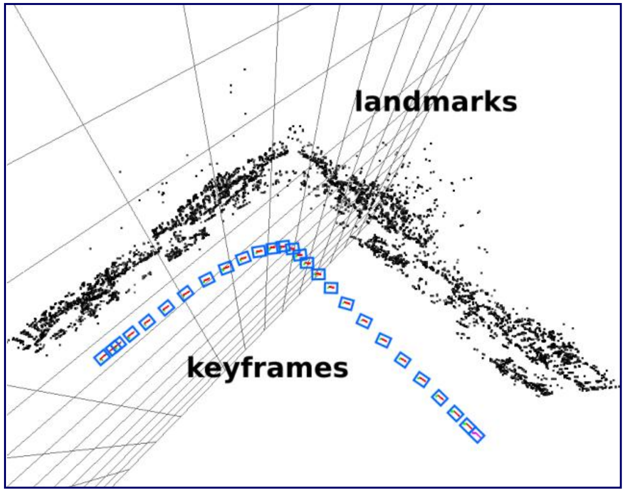
Chli & Davison, ICRA 2009

Real-time Single Camera SLAM systems

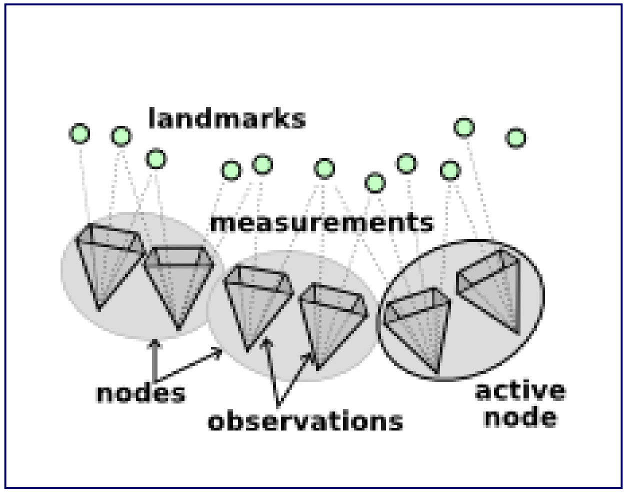
MonoSLAM is computationally expensive with increasing no. features



MonoSLAM
[Davison et al. 2007]



PTAM
[Klein, Murray 2008]

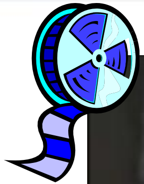


Graph-SLAM
[Eade, Drummond 2007]

✘ not ready yet to leave the lab & perform everyday tasks

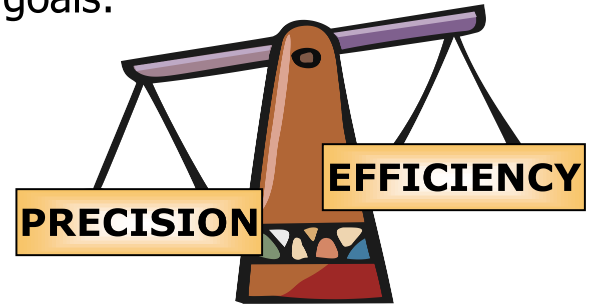
On-going Challenges

- Fast motion
- Large scales
- Robustness
- Rich maps
- Low computation for embedded apps



Ishikawa Komuro Lab.

- Handle **larger** amounts of data more **effectively**
- Competing goals:



key: agile manipulation of information

- Is it worth processing an extra piece of data?

- Employ **Information Theory** to
  - Quantify amount of information gained
  - Understand the problems we are trying to solve
  - Develop robust + dynamic algorithms

[Chli, PhD Thesis 2009]

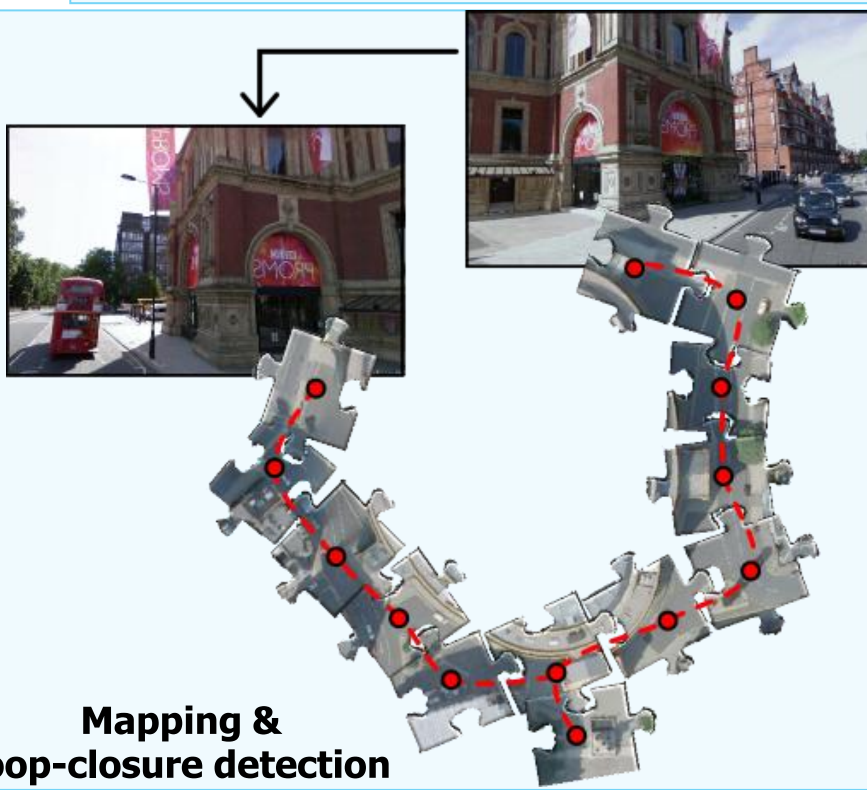
SLAM
#36

Essential components of a scalable SLAM system

1

Robust local motion estimation

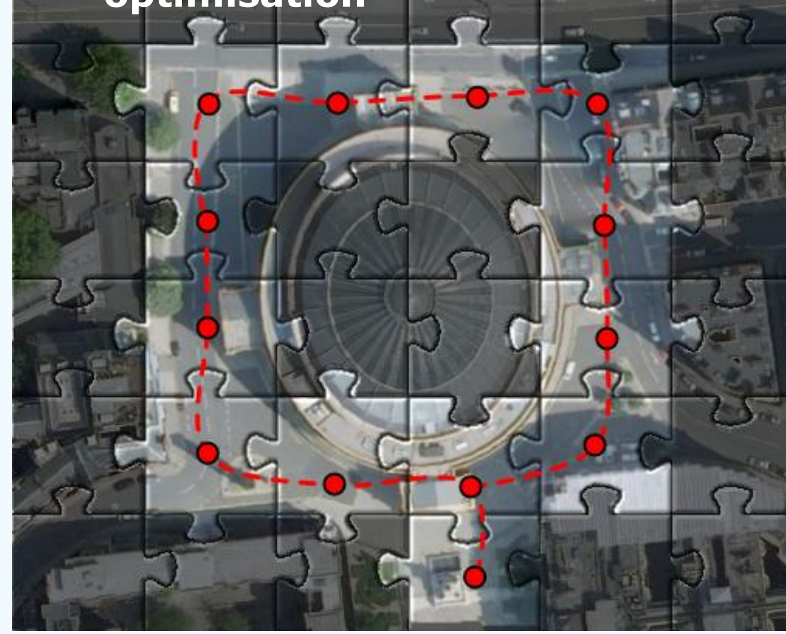
2



Mapping & loop-closure detection

3

Map management & optimisation

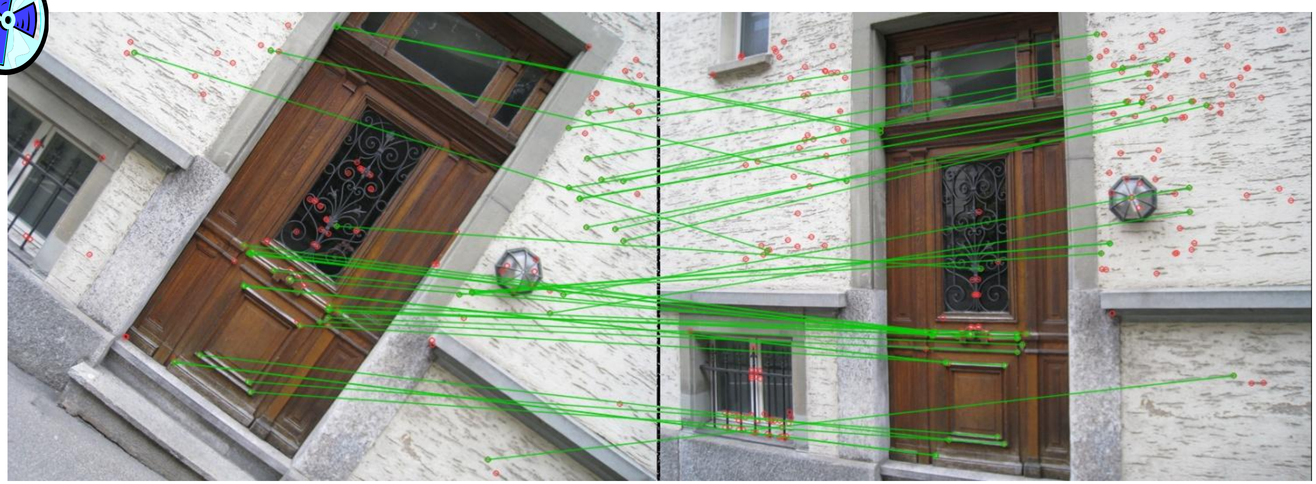


[Chli, PhD Thesis 2009]

Margarita Chli, July 2011

BRISK:
Binary Robust Invariant Scalable Keypoints

- Type of features: essential for robust tracking
- SIFT, SURF: High- performance **BUT**, inappropriate for real-time, low-computation applications.
- Maybe a combination of FAST+BRIEF is more appropriate ➪ affects robustness

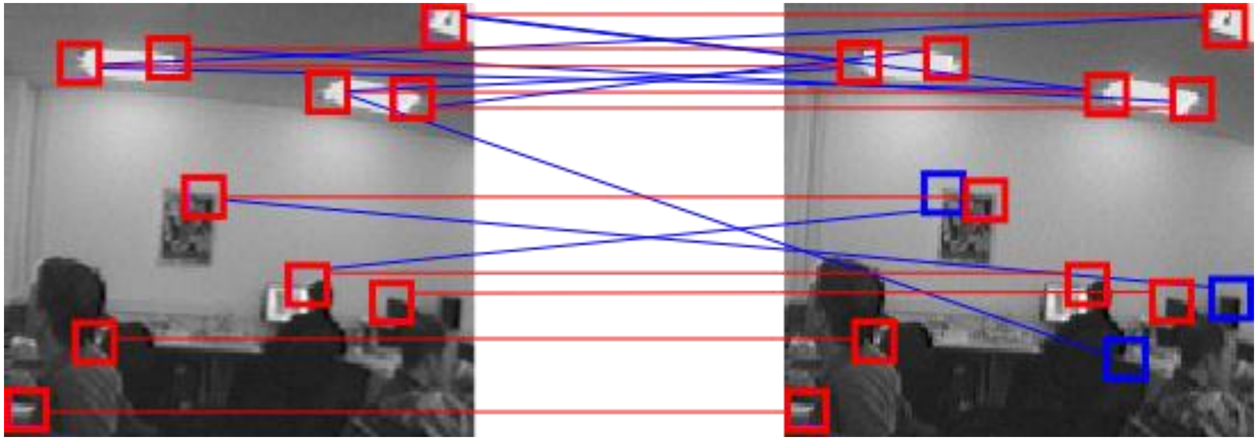


BRISK: Real-time high quality features, [Leutenegger et al, ICCV 2011]

- Comparable performance to SURF
- Much faster (an order of magnitude faster in some cases)

Efficient & Robust Matching

- First cue for matching: similar appearance of features

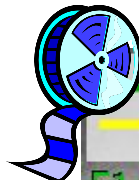


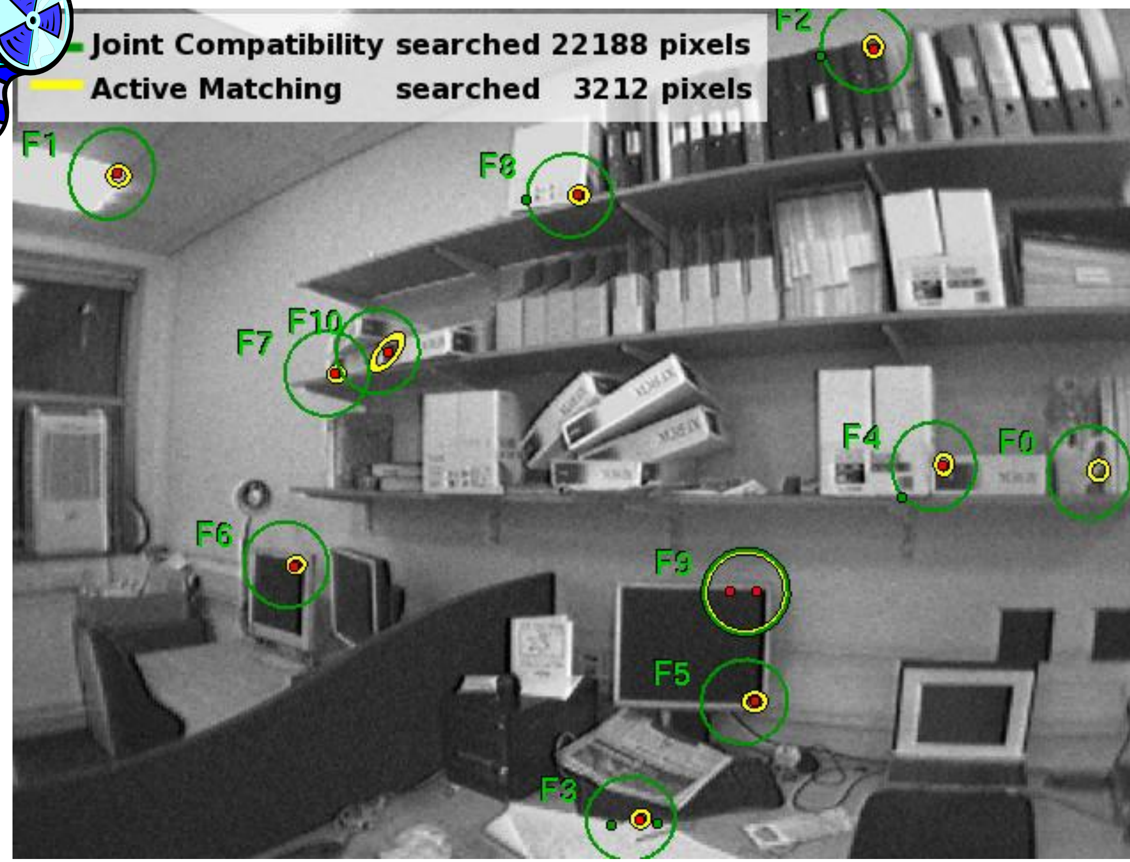
- Resolve inevitable **mismatches** by searching for consensus (agreement with global model)
- Example:
  RANSAC - randomly choose a set, hypothesize a solution & check and count
    - Fast
    - Sensitive to high data contamination
    - Relies on application specific thresholds

Active Matching: an example

Active Matching:

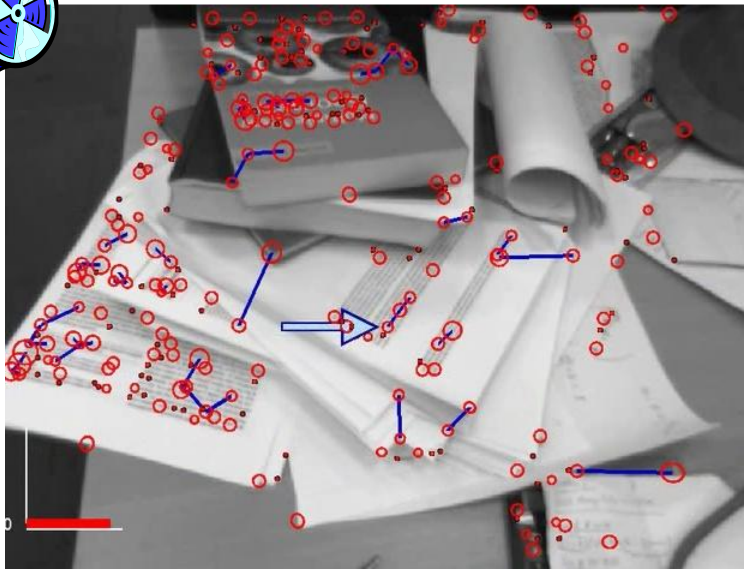
- Multi-hypothesis
- Step-by-step search for global consensus
- Choose to measure the most informative feature at each stage





[Chli,Davison, ECCV 2008]

Denser, real-time matching



[Handa et al., CVPR 2010]

- **Why?** ⇨ need bigger + better solutions (e.g. scene reconstruction)
  denser data = more info = more processing

- **How?** ⇨ brute-force algorithms like **RANSAC**:
  - ✓ can handle large amounts of data efficiently
  - ✗ rely on randomness and ad-hoc thresholds (e.g. no. iterations)
    ⇨ gain ground on speed, sacrificing valued cues

- **Fully probabilistic methods**: (e.g. JCBB, AM)
  - ✓ robustness to dynamic conditions
  - ✗ yet to prove their ability in real-time, dense matching

- **Goal**: bring fully probabilistic solutions on the same basis of applications as RANSAC-based techniques ⇨ approach: sparsify priors for real-time, robust dense matching

Live Dense Reconstruction

- During live camera tracking, perform dense per-pixel surface reconstruction
- Relies heavily on GPU processing for dense image matching

[Newcombe, Davison CVPR 2010]