VR Basics - SLAM

- Srikrishna Sadula

Contents

Brief !

Self-Driving Cars

Geo-mapping

Aerial vehicles

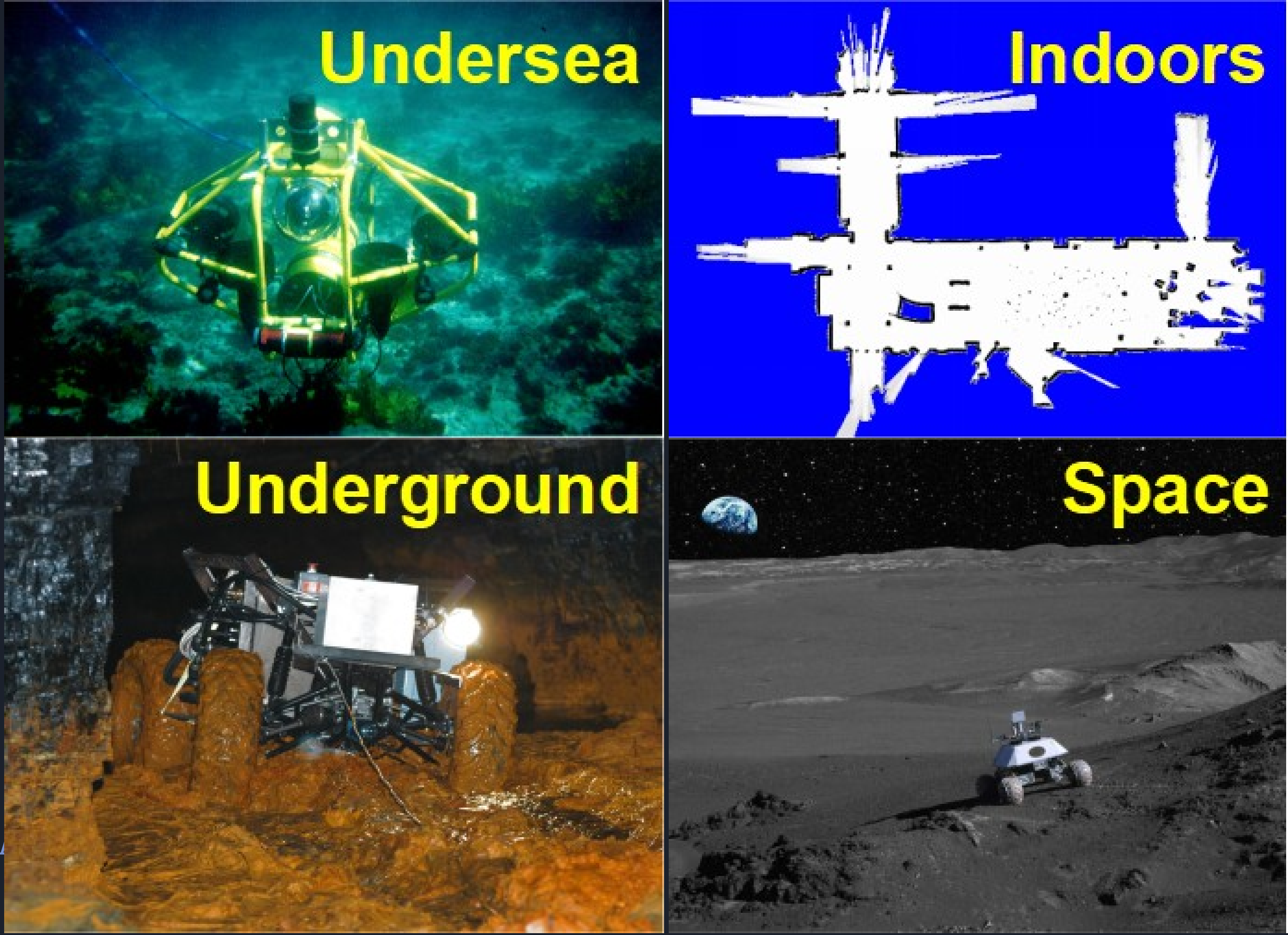
Domestic robots

inside human body

Planetary rovers

At AR/VR?

https://www.youtube.com/



Problem definition and solution approach

Given a series of sensor observations o_t over discrete time steps t, the SLAM problem is to compute an estimate of the agent's location x_t and a map of the environment m_t. All quantities are usually probabilistic, so the objective is to compute:

$$P(m_t, x_t | o_{1:t})$$

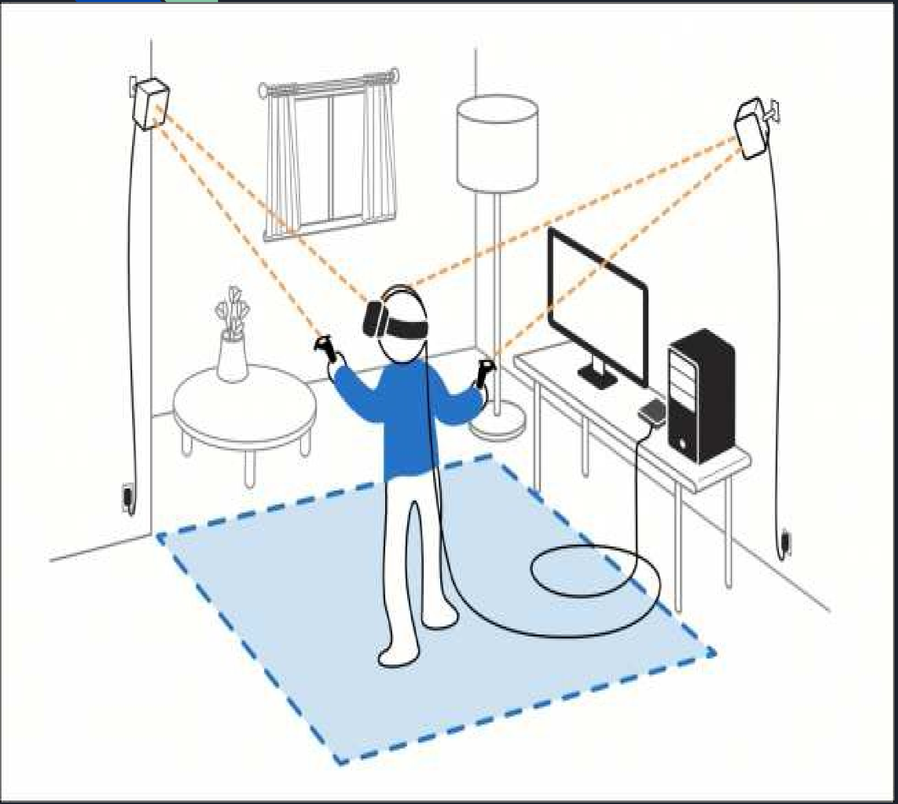
Applying Bayes' rule gives a framework for sequentially updating the location posteriors, given a map and a transition function $P(x_t | x_{t-1})$,

$$P(x_t | o_{1:t}, m_t) = \sum_{m_{t-1}} P(o_t | x_t, m_t) \sum_{x_{t-1}} P(x_t | x_{t-1}) P(x_{t-1} | m_t, o_{1:t-1}) / Z$$

Similarly the map can be updated sequentially by

$$P(m_t | x_t, o_{1:t}) = \sum_{x_t} \sum_{m_t} P(m_t | x_t, m_{t-1}, o_t) P(m_{t-1}, x_t | o_{1:t-1}, m_{t-1})$$

Sensors feeding to SLAM concept

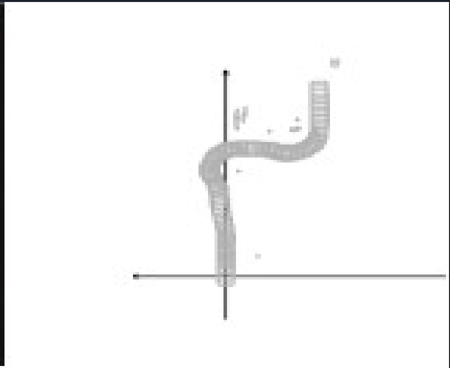
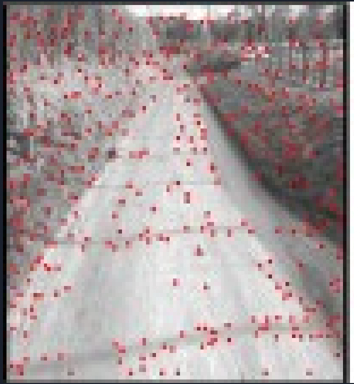
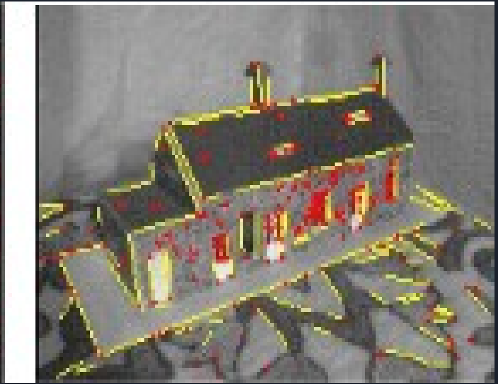


Visual SLAM

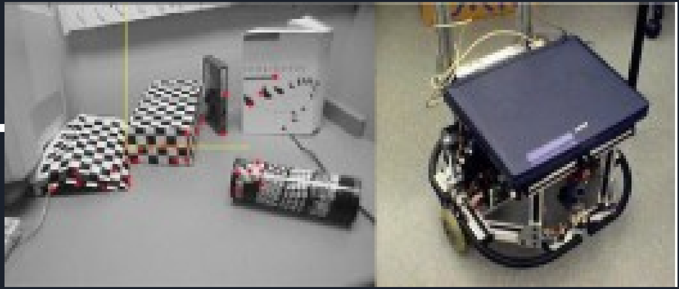
-Closed loop estimation, predictive, efficient.
-Live demos!
-Focus on a single visual sensor in a small area; drift-free, consistent localisation.
-Many possible applications easily apparent.
Commodity hardware (cameras and processors); open source software.
-Research is evolving towards general real-time spatial Perception

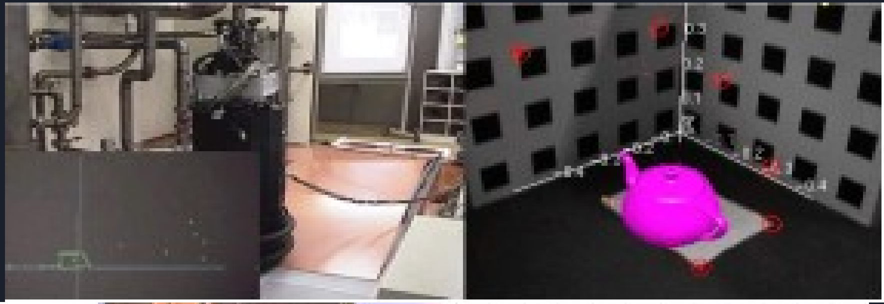
Early research

- SLAM with Active Vision (with David Murray, Oxford). 5Hz real-time loop on a 100MHz PC: Generalised system at AIST, Japan and first SceneLib open source
- DROID (Harris, late 1980s, feature-based VO)
- Off-line SFM moving towards sequence processing (e.g. Fitzgibbon, Pollefeys).
- EKF SLAM with non-visual sensors (Durrant-Whyte, Leonard, etc.).
- Laser scan matching (e.g. Gutmann and Konolige). The mobile robotics community had almost completely **turned away from vision.**
- The computer vision community had almost completely **turned away from real-time and robotics.**

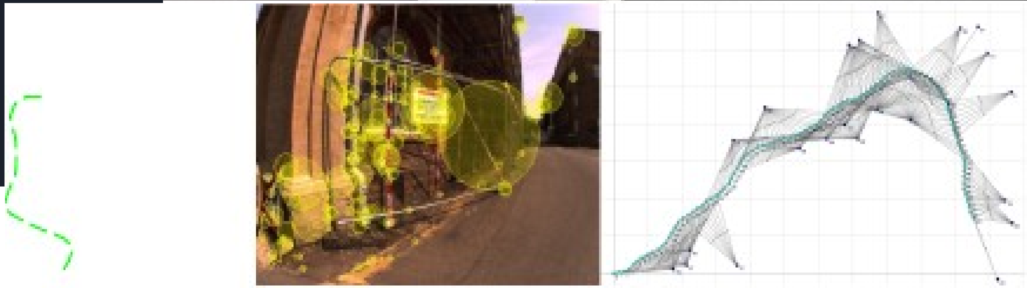





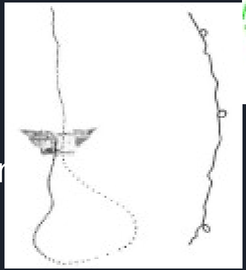
contd..

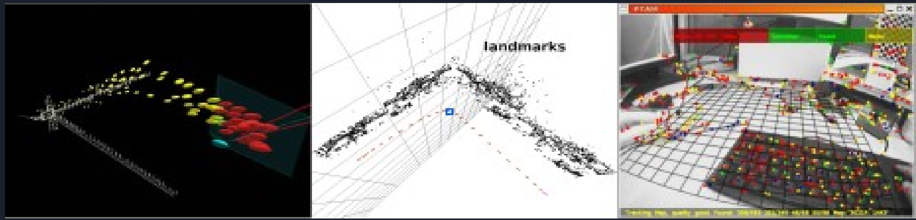


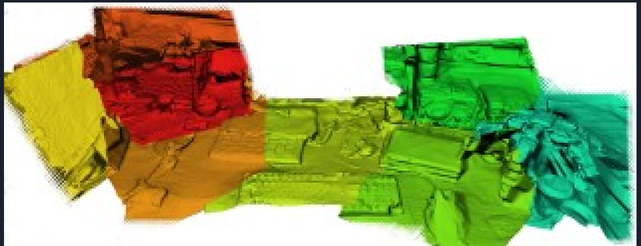


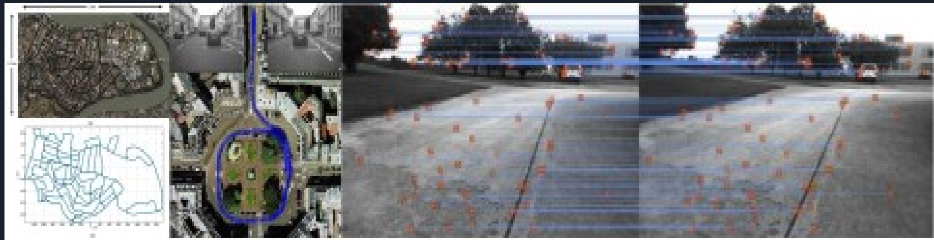
- 3D motion and tracking
- 3D monocular SLAM
- Sparse feature based SLAM EKF
- Aerial SLAM
- Large scale mapping
- Graph SLAM
- Relocalization-SLAM
- Dense tracking and mappin



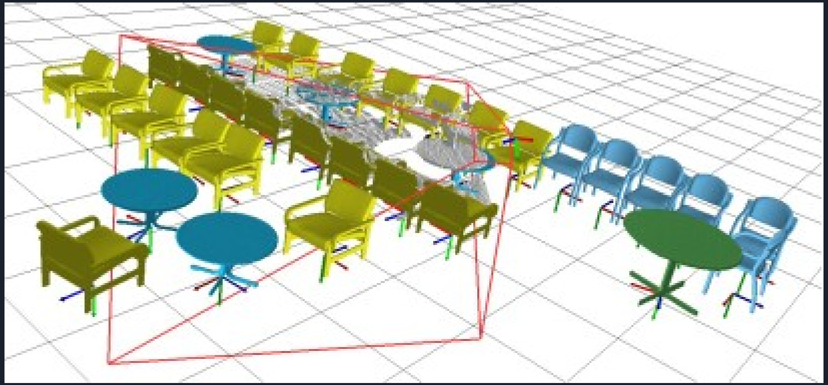








Contd..



-object level slam

{bring object recognition to the front of

SLAM and directly build a map at that

Level to benefit from strong predictions

Immediately..}

Predict, measure, Update will become even stronger with object

Algorithms & Libraries

- Extended Kalman Filter SLAM: (called EKF SLAM)
- Particle Filter SLAM: (called FAST SLAM)
- Graph-Based SLAM

http://openslam.org/

Other open source Libraries

- Robot Operating System (ROS) (slam algos)
- Point cloud (3D maps)
- visual features from Opencv

Where are SDKs??

Leading SLAM SDKs

- KUDAN
- Wikitude
- 13th lab, Metaio
- OSVR (open source VR) and many more !!

Common Features in most SDKs:

- Markerless Instant Tracking
- Object Recognition and Tracking
- Extended Tracking
- Optimal for indoors and outdoors
- iOS Native and JavaScript SDK
- Android Native and JavaScript SDK
- Support for Unity, PhoneGap, Titanium and Xamarin

SDK integration in Unity

Let’s Turn to Unity for a while !!

Examples

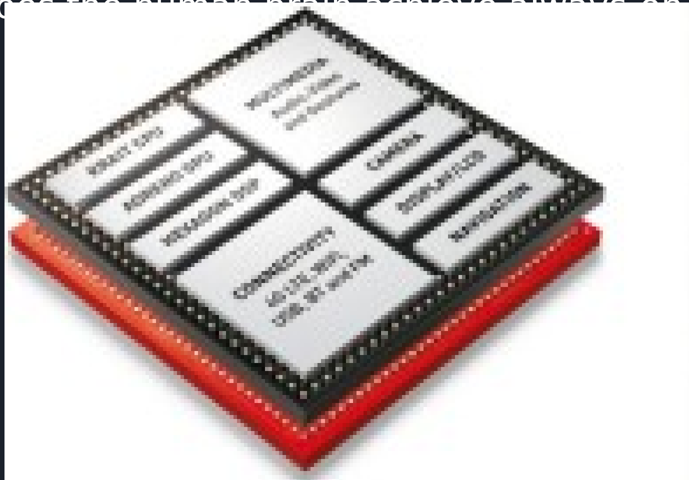
- https://www.wikitude.com/showcase/
- https://www.wikitude.com/showcase/ribena/

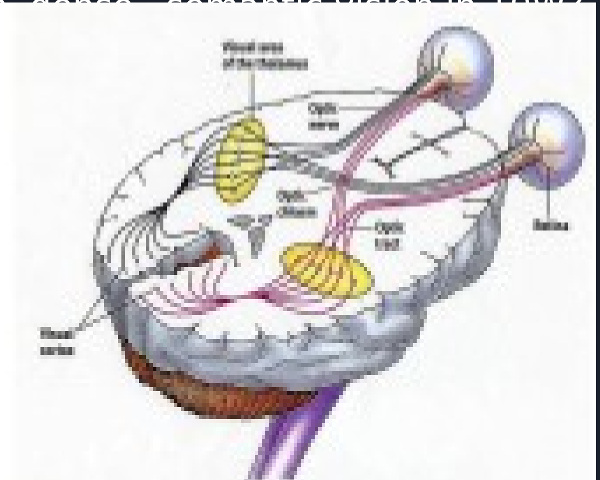
Still there’s lot gap to fill in !!

Smartphone system-on-chip technology will provide the template for low power smart devices — and computer vision will be a major driver.

CPUs, GPUs and increasingly specialised application-specific ‘ASIC’ chips.

But how does the human brain achieve always on, dense , semantic vision in 10W?





Thanks

QA

EKF (extended Kalman Filter)

- Non linear version of Kalman filter
- Linearizes about an estimate of the current mean and covariance

Model

$$\dot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \mathbf{u}(t)\big) + \mathbf{w}(t) \qquad \mathbf{w}(t) \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}(t)\big)$$

$$\mathbf{z}(t) = h\big(\mathbf{x}(t)\big) + \mathbf{v}(t) \qquad \mathbf{v}(t) \sim \mathcal{N}\big(\mathbf{0}, \mathbf{R}(t)\big)$$

Initialize

$$\hat{\mathbf{x}}(t_0) = E\big[\mathbf{x}(t_0)\big], \mathbf{P}(t_0) = Var\big[\mathbf{x}(t_0)\big]$$

Predict-Update

$$\dot{\hat{\mathbf{x}}}(t) = f\big(\hat{\mathbf{x}}(t), \mathbf{u}(t)\big) + \mathbf{K}(t)\Big(\mathbf{z}(t) - h\big(\hat{\mathbf{x}}(t)\big)\Big)$$

$$\dot{\mathbf{P}}(t) = \mathbf{F}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}(t)^\top - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}(t) + \mathbf{Q}(t)$$

$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)^\top\mathbf{R}(t)^{-1}$$

$$\mathbf{F}(t) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(t), \mathbf{u}(t)}$$

$$\mathbf{H}(t) = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(t)}$$

Brute force vision - vision algorithms to be brought into mobile systems, dense reconstruction, random forests