Computer Vision cmput 428/615

Intro to

Practical 3D Vision

Upcoming course project

Martin Jagersand

Where we are in course

- Done:
  - Basics of computer vision, images
  - 2D Point correspondences, visual motion, tracking
  - Euclidean and some projective geometry in 2D and 3D
- Upcoming:
  - More projective geometry in 2D and 3D
  - 3D modeling from 2D image points
  - How surfaces look? Texture, Light models, Reflectance
  - Human vision
  - Applications of computer vision (sci/eng/ind, entertainment, educational, service)

The camera matrix

$$(U, V, W) \rightarrow \left(\frac{U}{W}, \frac{V}{W}\right) = (u, v)$$

- Homogenous coordinates for 3D
  - four coordinates for 3D point, 3 for a 2D

$$\begin{pmatrix} U \\ V \\ W \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ T \end{pmatrix}$$

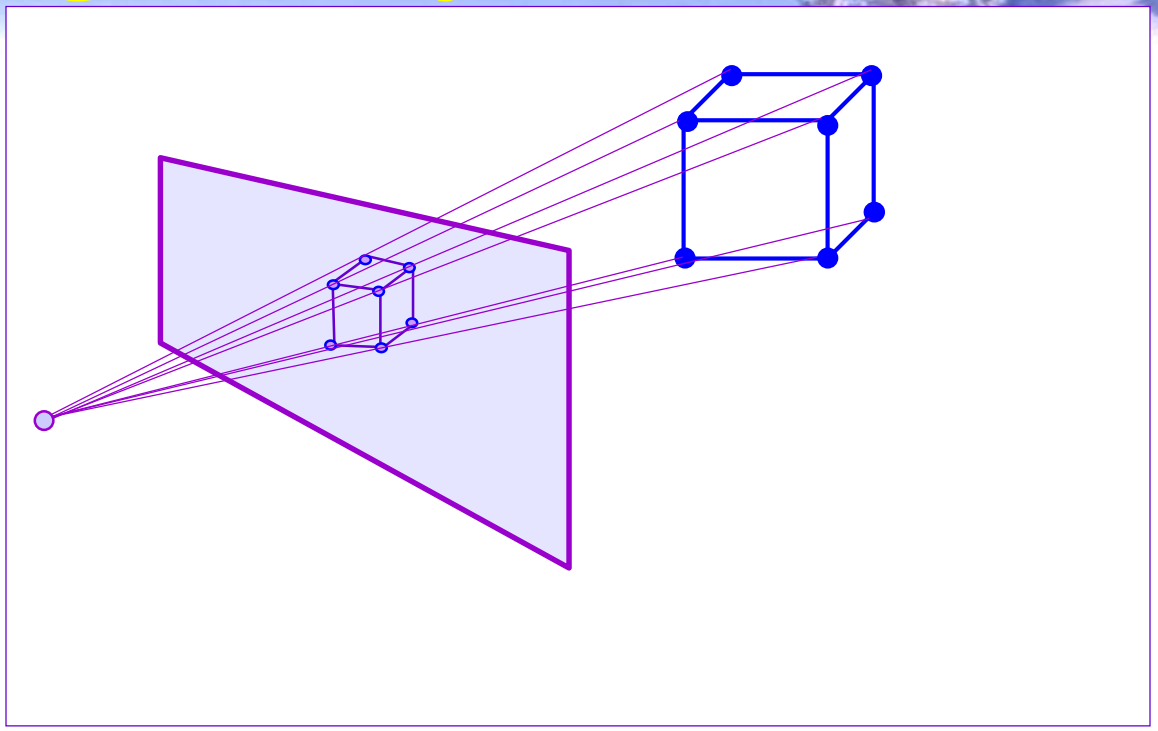
- When coordinate systems are not aligned
  - Projective: x image coordinates, X 3D coord, and P an arbitrary 3x4 matrix
  - x = PX
  - Euclidean
  - x = [R|T]X

Upcoming 2 weeks
Multi-view geometry - resection

- Projection equation

  $x_i = P_i X$

- Resection:
  - $x_i, X \longrightarrow P_i$



Solve with SVD just like Homography

Given image points and 3D points calculate camera projection matrix.

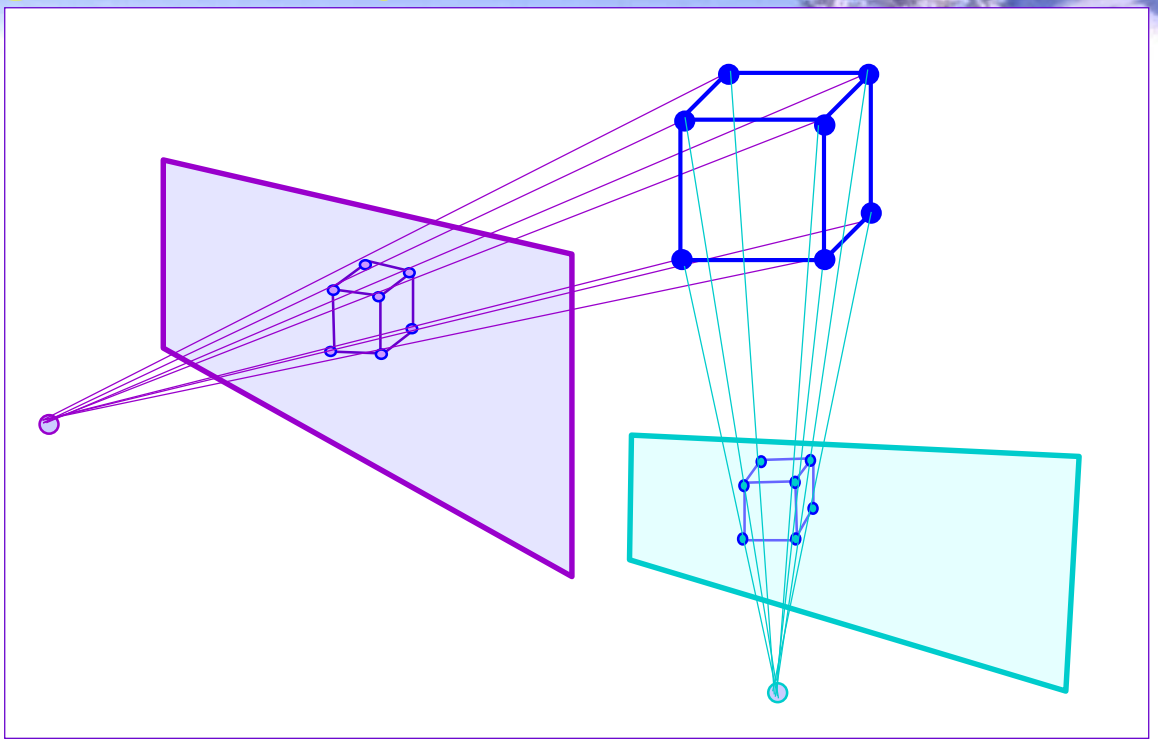
Upcoming 3 weeks
Multi-view geometry - intersection

- Projection equation

  $x_i = P_i X$

- Intersection:
  - $x_i, P_i \longrightarrow X$

SVD again!



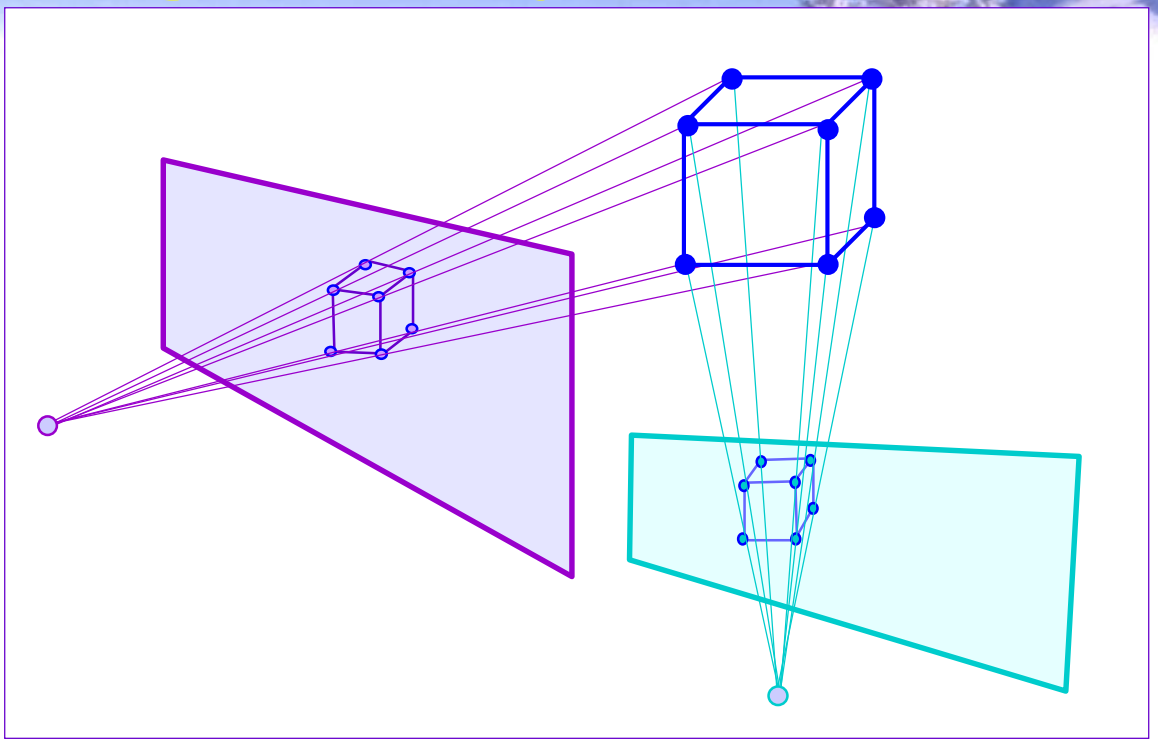
Given image points and camera projections in at least 2 views calculate the 3D points (structure)

Upcoming 4 weeks
Multi-view geometry - SFM

- Projection equation

  $x_i = P_i X$

- Structure from motion (SFM)
  - $x_i \longrightarrow P_i, X$



Given image points in at least 2 views calculate the 3D points (structure) and camera projection matrices (motion)

- Estimate projective structure
- Rectify the reconstruction to metric (autocalibration)

N-view geometry
Affine factorization
(HZ Ch 17, 18)

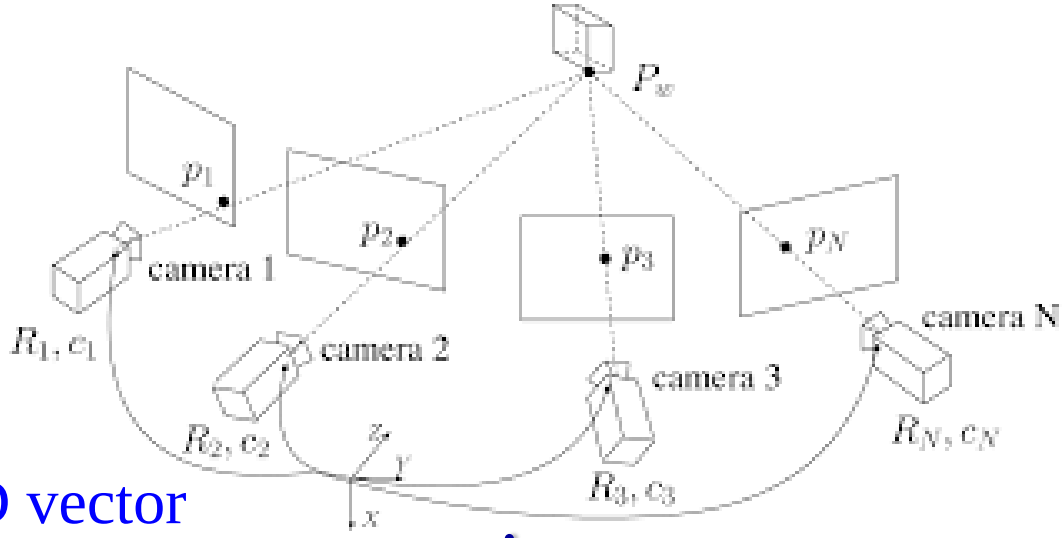
[Carlo Tomasi PhD thesis @CMU > Faculty offer at Stanford]

- Affine camera

$$P_\infty = [M \,|\, \mathbf{t}]$$

- Projection

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = M \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$



M 2x3 matrix; t 2D vector

- *n* points, *m* views: measurement matrix

$$\tilde{\mathbf{x}} = \mathbf{x} - \mathbf{t}$$

$$W = \begin{bmatrix} \tilde{\mathbf{x}}_1^1 & \cdots & \tilde{\mathbf{x}}_n^1 \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{x}}_1^m & \cdots & \tilde{\mathbf{x}}_n^m \end{bmatrix} = \begin{bmatrix} M^1 \\ \vdots \\ M^m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \cdots & \mathbf{X}_n \end{bmatrix}$$

W: Rank 3

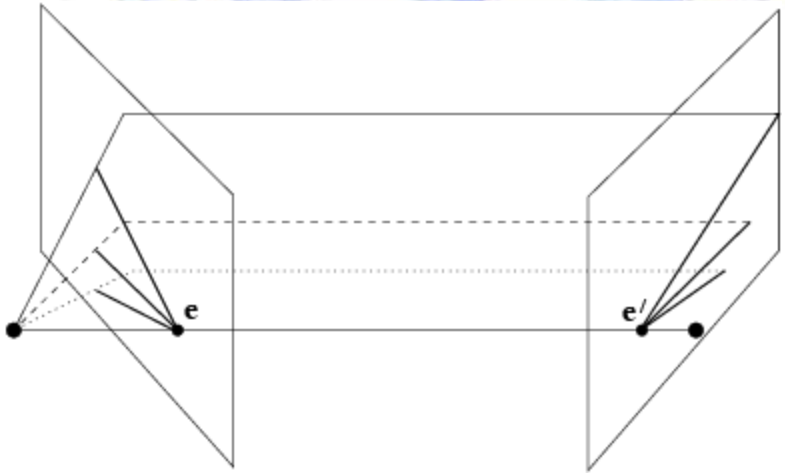
$$W = UDV^T$$

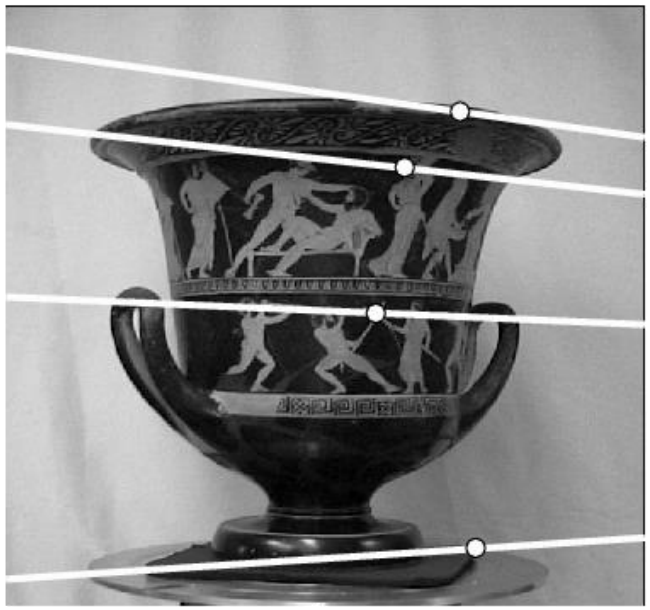
$$\hat{W} = \boxed{U_{2m\times 3}} \boxed{D_{3\times 3} V^T_{n\times 3}} = \hat{M}\hat{X}$$

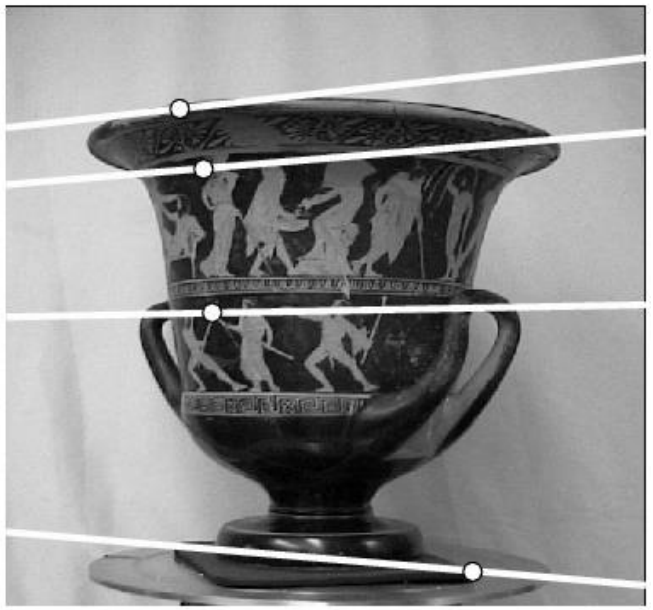
SVD!!

Assuming isotropic zero-mean Gaussian noise, factorization achieves ML affine reconstruction.

2-view geometry:
Epipolar lines and Fundamental matrix



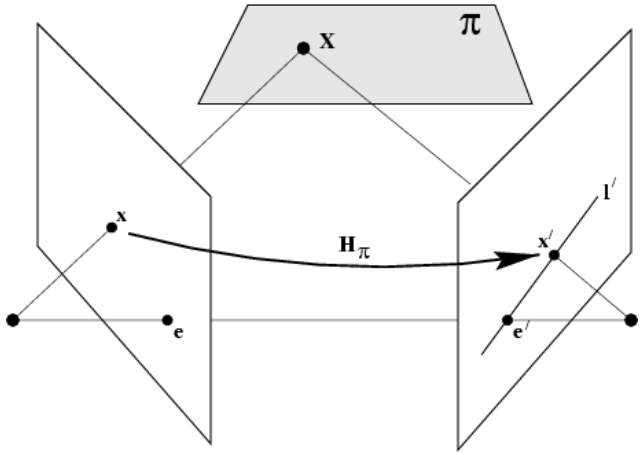




Fundamental matrix

[Faugeras '92, Hartley '92]

- Algebraic representation of epipolar geometry



Step 1: X on a plane π

$$\mathbf{x}' = H\mathbf{x}$$

Step 2: epipolar line l'

$$\mathbf{l}' = \mathbf{e}' \times \mathbf{x}' = [\mathbf{e}']_\times \mathbf{x}'$$
$$= [\mathbf{e}']_\times H\mathbf{x} = F\mathbf{x}$$

$$\mathbf{x}'^T F\mathbf{x} = 0$$

F

- 3x3, Rank 2, det(F)=0
- Linear sol. – 8 corr. Points (unique)
- Nonlinear sol. – 7 corr. points (3sol.)
- Very sensitive to noise & outliers

Epipolar lines: $\mathbf{l}' = F\mathbf{x} \quad \mathbf{l} = F^T\mathbf{x}'$

Epipoles: $F\mathbf{e} = 0 \quad F^T\mathbf{e}' = 0$

Projection matrices: $P = [I \mid \mathbf{0}]$

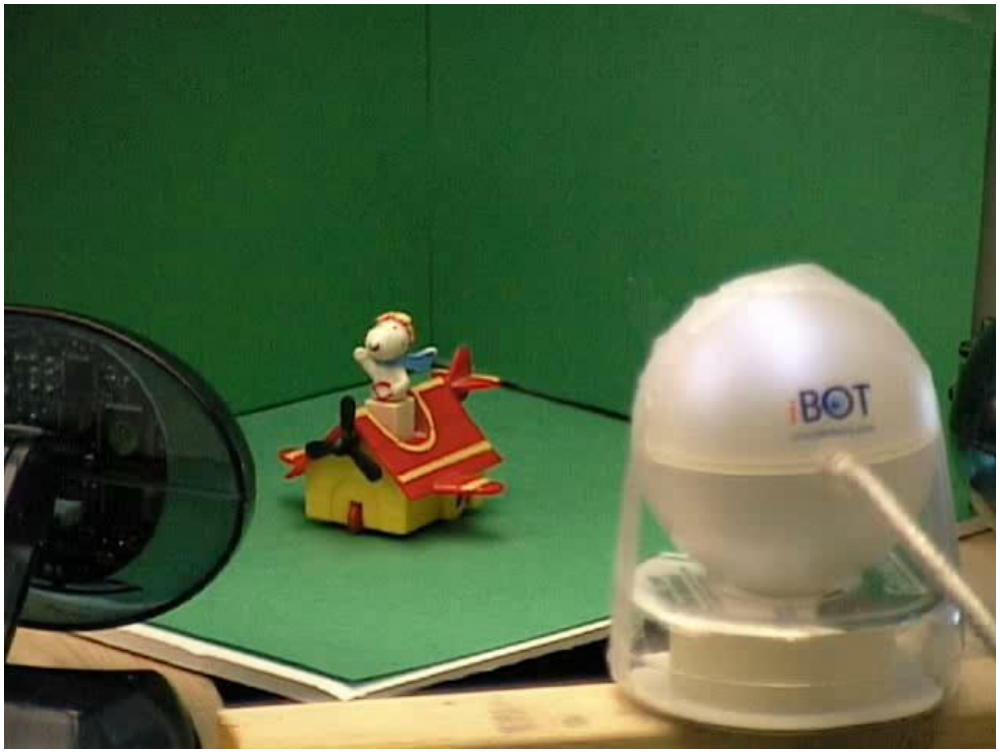
$$P' = \left[[\mathbf{e}']_\times F + \mathbf{e}'\mathbf{v}^T \mid \lambda \mathbf{e}' \right]$$

3D from video example
Capture objects

- Inexpensive
- Quick and convenient for the user
- Integrates with existing SW e.g. Blender, Maya

3D from video: Low budget

- Inexpensive
-
-

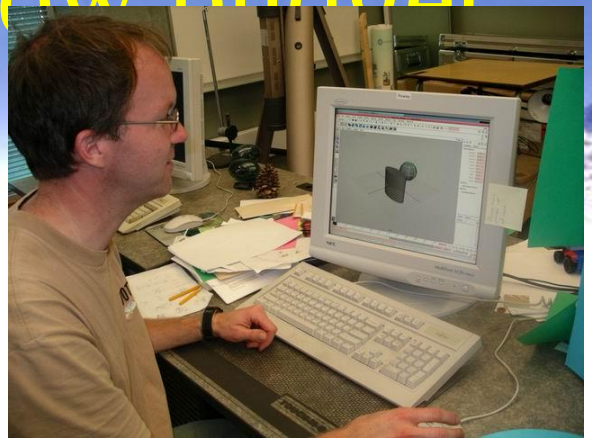


$100: Webcams, Digital Cams

$100,000 Laser scanners etc.

3D from video Low budget

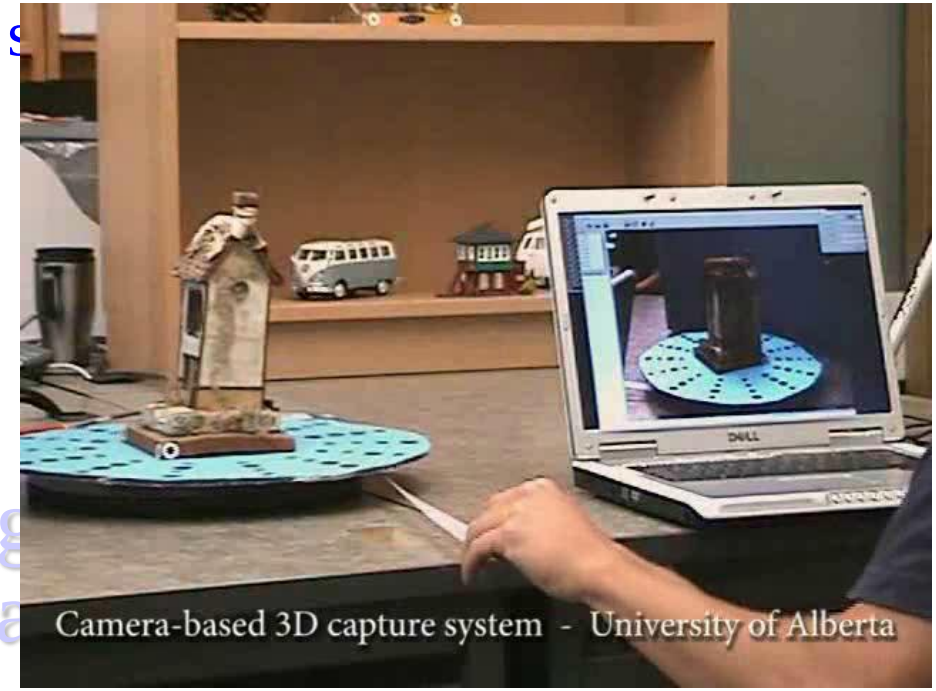
- Inexpensive



Modeling geom primitives into s

- Quick and convenient for the user

- Integrates with existing SW e.g. Blender, Maya



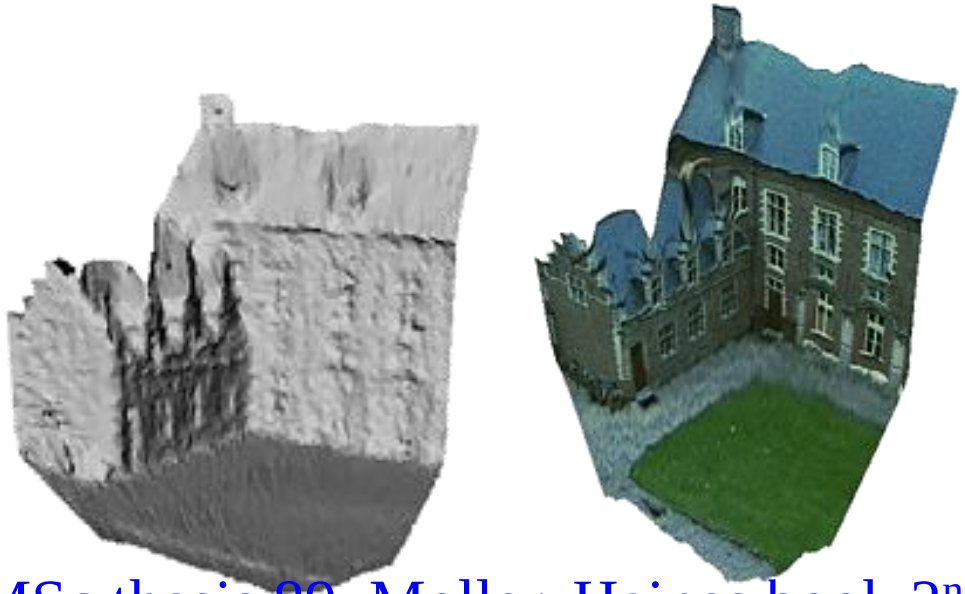
Capturing 3D from 2D video:

Low budget 3D from video

- Inexpensive
- Quick and convenient for the user
- Integrates with existing SW e.g. Blender, Maya

Texture

- Texture = Fine scale surface appearance.
- Often means endowing the surface of a geometric (polygonal) model with such fine scale properties.



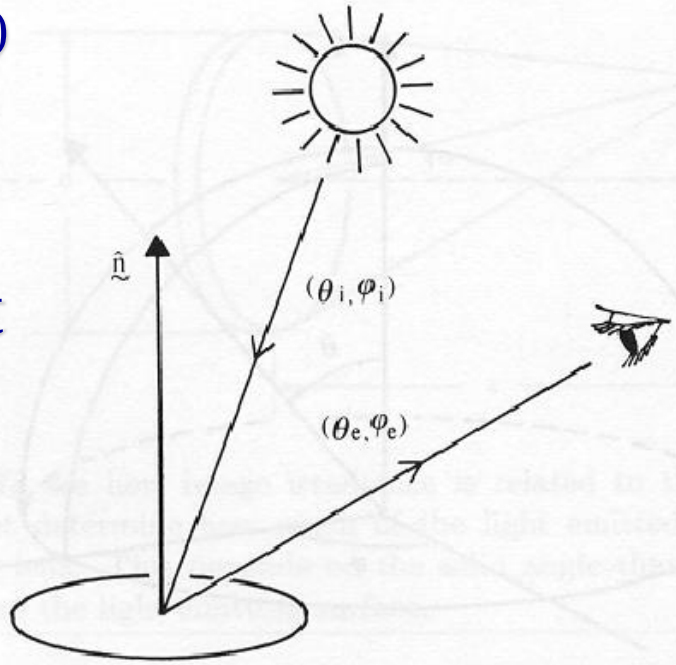
Std ref: Heckbert MSc thesis 89, Moller, Haines book 2nd ed. 2002.

What we see given a particular surface texture and light?

- BRDF

(Bi-Directional Reflectance Function)

- Describes how incoming light on a surface is reflected back in different directions.



Horn, 1986

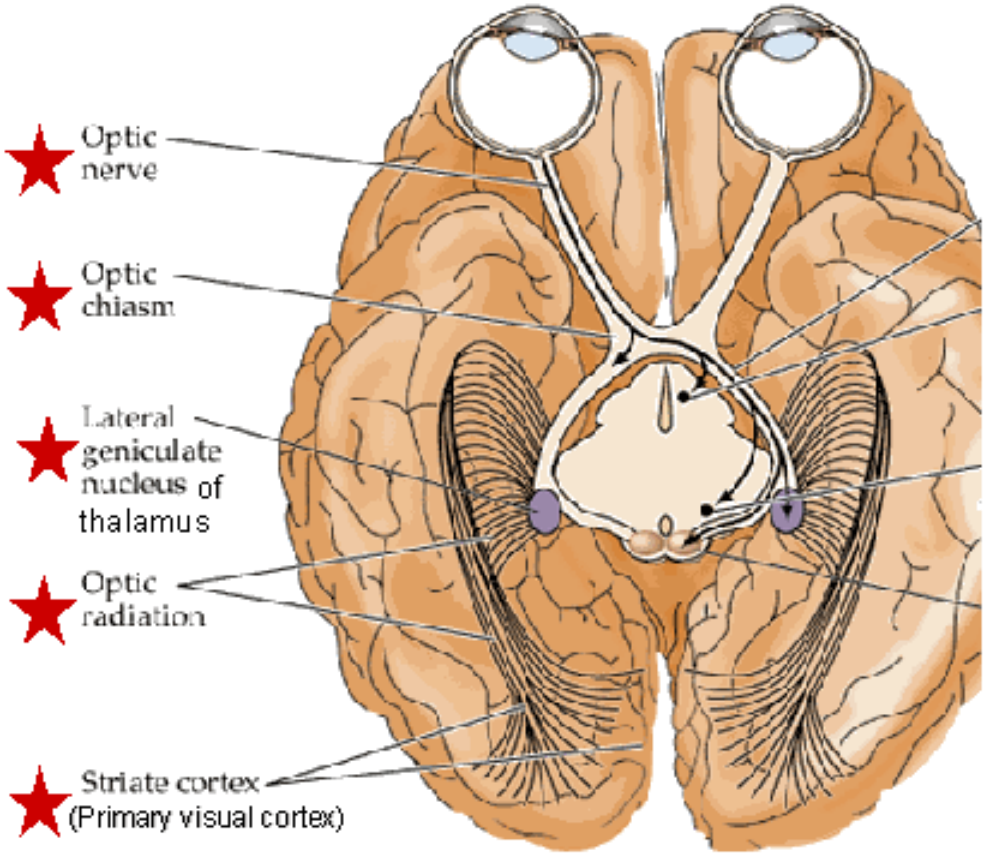
Figure 10-7. The bidirectional reflectance distribution function is the ratio of the radiance of the surface patch as viewed from the direction (θ_e, ϕ_e) to the irradiance resulting from illumination from the direction (θ_i, ϕ_i).

Light basis

$$\Delta \mathbf{T}_l = \mathbf{B}_\mathbf{l}\mathbf{y}_\mathbf{l}$$

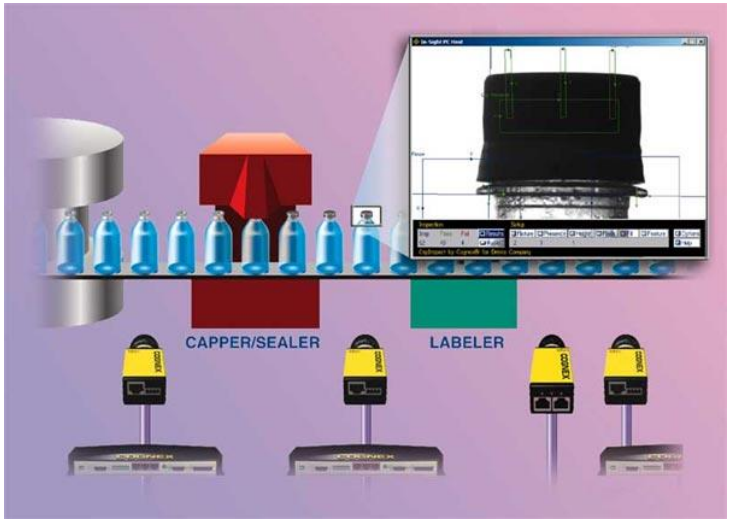
Demo E:\demos\light\light.exe

Processing elements and Pathways of Human Vision



- Eye transforms light into nerve impulses
- Optic chiasm splits left and right visual fields
- LGN: Exact function unknown. May have to do with stereo.
- V1 (Striate cortex) performs spatial filtering / coordinate transforms

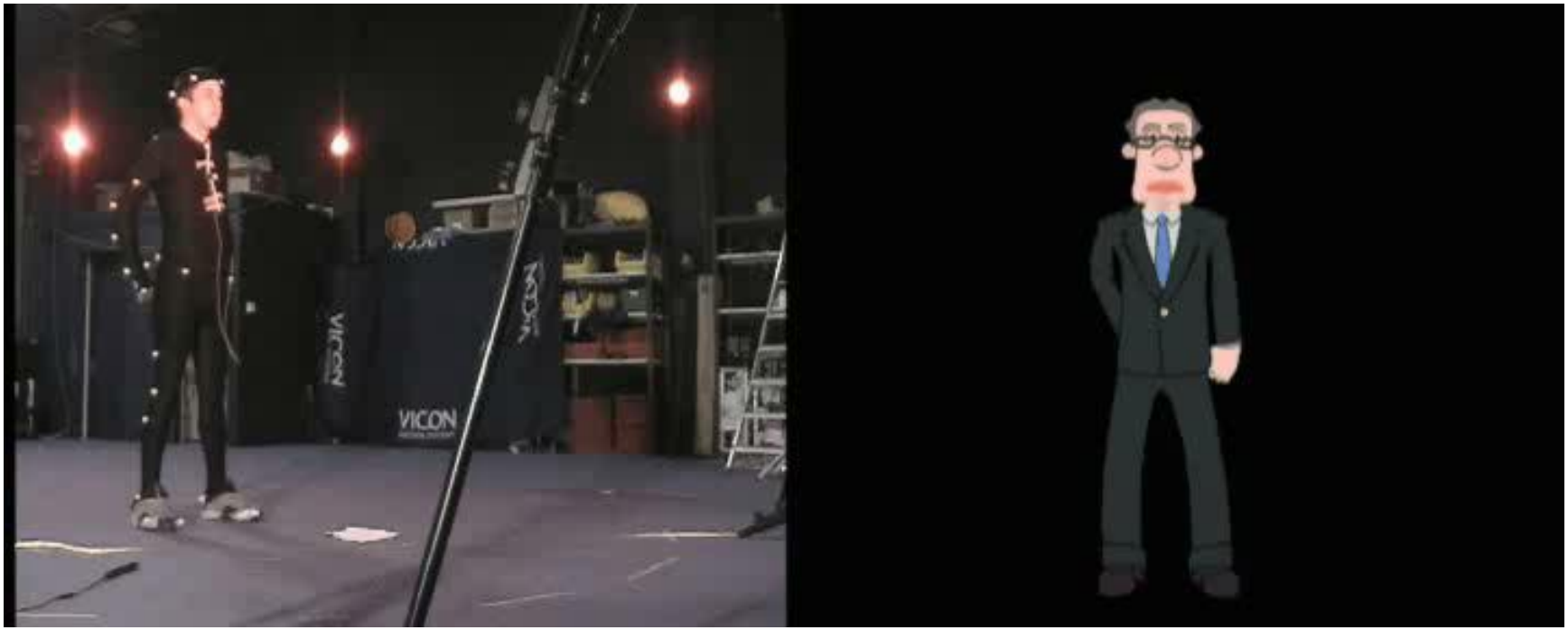
Industrial applications: Factory Inspection



Cognex's "CapInspect" system:

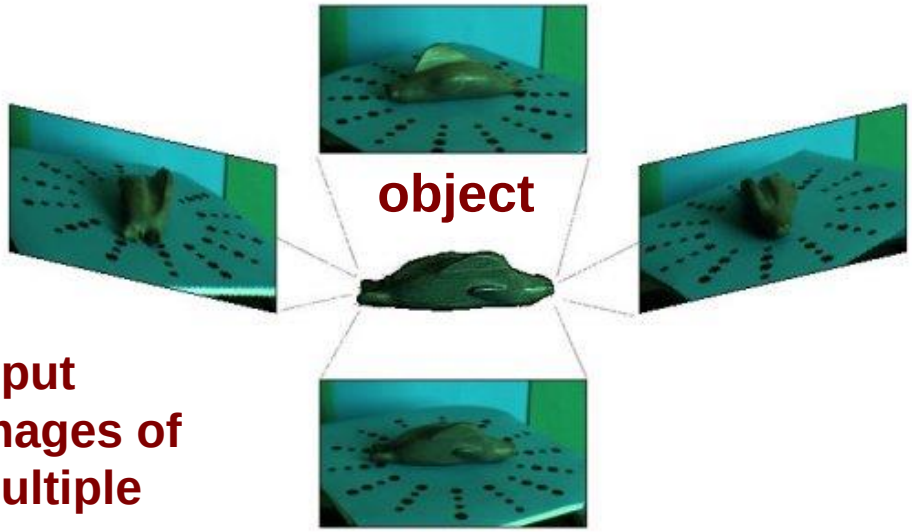
Low-level image analysis: Identify edges, regions
Mid-level: Distinguish "cap" from "no cap"
Estimation: What are orientation of cap, height of liquid?

Entertainment applications: Motion Capture / Animation



Vicon software:
12 cameras, 41 markers for body capture;
6 zoom cameras, 30 markers for face

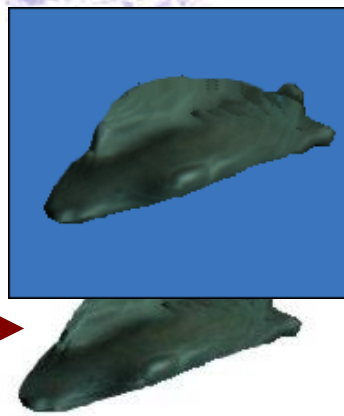
Applications educational Virtual Heritage



Input images of multiple views

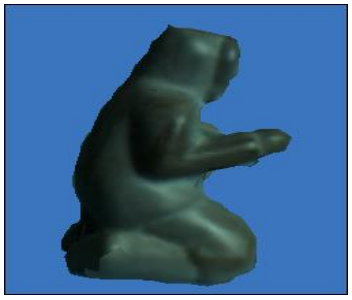
Geometric model from SFS

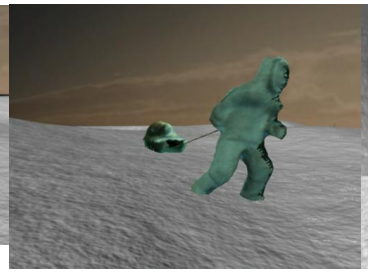
Dynamic texture



Rendered m

animations and Internet web study material.





Project and labs

- Optic flow and tracking: done
- Geometry and cameras: On now
- Upcoming labs:
  - L3: 3D modeling
  - (L4: Reflectance and appearance)
- Project Applications of Computer Vision:
  - Group of 2 or individual
  - Worth 2-3 labs. (20% ugrad, 30% grad)
  - P1: Review and proposal. Revised-> P1a, P1b
  - P2: Implementation, demo and final report: End of classes.

Goals of project

- Learn more in depth and hands-on about one of the course topics
  - Read web descriptions of system and research papers
  - Practically experiment with an implemented system to learn how to use it and what type of images/video it works well at.
  - Find our what are weaknesses and how those could be addressed.
- Study and implement a small addition or modification to a part of this system.

Value of hands-on experiments on implemented systems

- Computer vision methods are fragile. (similar to much other research)
- Generally don't work robustly and reliably on all inputs, including normal real-world video/images
- Some implemented systems represent many man years of effort. Working with these can learn much of how they work w/o taking years!

Systems presented in course
(and where Instructor and TA can help)

- **MTF, XVision:** Tracking system. (Other individual trackers are available in e.g. OpenCV and other web sites)
- **Capgui:** Shape from Silhouette Geometry and Multitexturing
- **ARC3D:** Corresponding points based SFM (Structure from Motion). Download geometry and cameras
- **Photosynth:** similar SFM. View geometry and cameras on web. AFAIK: format of data/geometry not published.
- **Capture_UI:** Simpler factorization used in lab 3
- **PTAM, mono SLAM, ORB SLAM**: Real time incremental geometry
- **CARV**: Physically correct triangulation of 3D points

Some possible project topics who they may fit

- **Dynamic vision/Tracking:** Interest in dynamics, robotics, filtering can help
- **Geometry related:** Suitable for someone with interest in computer vision and graphics and/or computational geometry
- **Appearance/texture related:** Interest/knowledge in data representation, subspace/manifold estimation helps.

Multi-texturing

Multi texturing: capgui blending 20 basis textures. Generated by PCA over all the input textures.

Possible improvement:

- Partition view angles into subsets, then PCA on each subset.
- What is the best subset of view angles? Quadrants? Octants? Scene/object dependent?
- Figure out how to handle overlap and the boundaries.

Study references from graphics literature, e.g. "A Theory of Locally Low Dimensional Light Transport" by Dhruv Mahajan, Ira Kemelmacher Shlizerman, Ravi Ramamoorthi, and Peter Belhumeur

Multi-texturing (cont)

- In some cases (e.g. geometry very inaccurate) other (e.g. non-linear) subspace methods may perform better (e.g. isomap, kernel pca, etc). Try to find the properties of the texture manifold and matc h that to a suitable method.
- Write a texturing plugin using some method (e.g. original or one of the above) for Blender so captured objects can be integrated and rendered in blender scenes.
- Texture decomposition into low/high frequency.

Single texturing

- Legacy SW that can only handle a single texture,
- How to distill the in some measure "best" color w.r.t. all the input data for a particular object point?
- Heuristic methods for this
  - Blending images for texturing 3D Models, Adam Baumberg BMVC 2002)
- Globally optimal:
  - Spatio-temporal Image-based Texture Atlases for Dynamic 3-D Models Janko and Jean-Philippe Pons 3DIM2009,
  - Seamless image-based texture atlases using multi-band blending, Allène, C. and Pons, J-P. and Keriven, R. ICPR 2008,
  - Seamless Mosaicing of Image-Based Texture Maps Victor Lempitsky and Denis Ivanov CVPR.2007)

Study these methods (we have some example code that can get you started) and see how they compare.

See if you can integrate one into the CapGui

Tracking

1. Study a different tracking modality, e.g.
   - Hyperplanes for SSD tracking (Juri and Dhome, PAMI 2004)
   - Meanshift (Dorin Comaniciu, and Peter Meer)

   Integrate this tracking into XVison and compare performance with existing XVision trackers.

- Implement compositional tracking similar to the IFA
  - Incremental Focus of Attention system by Toyama and Hager

Dynamics in P^2

- How do Newton's laws look in the image plane?
- What invariants are preserved? Are there others we can compute?
- How do we build a motion model for what we observe in tracking?
- How do we control motion of a robot from what we see in video (visual servoing)?

Quadri-linear Factorization

- Under the usual equality x=y we can compute a low rank approximation (aka PCA) using the regular SVD: $\mathbf{M}=U\Sigma V^T$
- Under projective equality $\lambda x=y$ we need to numerically account for scale factors λ
- How do we compute a projective "SVD" $\Lambda\mathbf{M}=U\Sigma V^T$?

How to go about it

- For example in a geometry project:
  - First study existing methods. The HZ book has the mathematical fundamentalsThe Szeliski book describes the state of the art and has paper refences.
  - Try one or more of the existing systems to get a feel for how current methods work. Systems you can try include our capgui (shown in the labs), Arc3D, Photosynth.
  - Try to find what type of models can be created, what are limitations (e.g. how do the images have to be taken).
- The reading and trying is for the review part to go with the proposal P1 to be handed in first.
- Then find something to research and implement yourself. This could be a subtopic such as geometric refinement or an application where you capture and use vision based models for some effects in e.g. an animation or computer game.

How to go about it (cont)

In a tracking project you can similarly

- first try our XVision system and or other trackers you can download,
- then propose some extension or additional tracking modality to implement.

Application project
Possible topics

- Mathematical/geometric: A capture system for man-made structure:
  - Capture structure using some a-priori knowledge: orthogonality, parallelism, planes. This makes capture easier and gives Euclidean structure w/o cumbersome "self calibration"
- Systems: Live video or 3D conferencing
  - Track, compress video (dynamic) textures
  - Encode, transmit and render

- Games: Capture 3D characters, scenes, script a plot and put into game engine
- Haptics/robotics: Capture something visually and allow haptic exploration of this. (ie as aid to blind)
- Survelliance: Track people etc (ie for security system)

Application project

- Discuss ideas for interesting projects
- Identify some books, web pages or papers to serve as core information sources.
- Identify a handful of early classic papers to seed your literature search. I will do my best to help here also.
- Tips on carrying out your projects
  - Balance between reading and doing

Schedule

- Now (reading week) good time to think about what you would be interested in doing
  - Find available systems (ours and on Internet) and try to use them practically.
- March

  project proposal.
  - Include reference list and a start at literature review, ie. Read some papers and write a few pages summary
  - Include experience from trying
- Throughout course in class: Keep up to date on your project progress.
- End of semester: Project reports.

Resources

- SW: I/we will install and try to help support:
  - Real time video input (Currently linux, could be Windows)
  - Basic tracking, MTF, Xvision, OpenCV, ROS
  - Geometry: CapGui, SLAM, CARV, Hand-eye, Robotics code
- HW: Access to machines in Robotics/Vision lab
  - Cameras: Linux IEEE 1394 cameras in lab. Could also use digital still camera, camcorder.
  - Visualization: Ok in lab (HW acc graphics), Better in research lab (new ATI and nvidia, SGI HW, projectors and CAVE)
  - Vision and haptics: Have phantoms.
  - Vision for motion control: Robot arms: Could be set up.
  - Anything else? Some resources for buying available

Martin's tips

- Plan incremental progress and checkpoints.
  - Makes it easier to identify promising directions as well as difficulties and redefine plans as needed.
- Find balance between reading and doing
  - It is difficult to fully grasp methods by only reading
  - Some experiments are incomplete, results wrong
- Practical trying out can add a lot of insight.
  - Learn how to quickly prototype in e.g. matlab

Literature search

- **Goal:** Find the handful to dozen most relevant and recent papers in a subarea.
- **Method:**
  1. Seed with a few relevant papers.
  2. Do citation search backwards and forwards.
  3. Find common “buzz words”. Do title and abstract text search.
  4. Do internet search. e.g. “Cora” from justresearch
  5. Check most recent proceedings manually. (They won’t be indexed yet)

Report:

1. Review
   - Summarize the main contributions and comparing the results in the papers.
2. Your contribution and experiments.
   - Methods
   - Results
3. Discussion
   - Where does it fit into the bigger picture
   - Future work