Lecture 17 - W9L1

Regularization (measures contd.)

* Batch Normalization :

Batch outputs for current layer $\{z^{(i)}\}_{i=1}^{q} \longrightarrow \{\hat{z}^{(i)}\}_{i=1}^{q}$ (# of examples)

$z^{(i)} \in \mathbb{R}^p$ (p = # units)

$$\hat{z}_j^{(i)} = \frac{z_j^{(i)} - \mu_j}{\sigma_j}$$

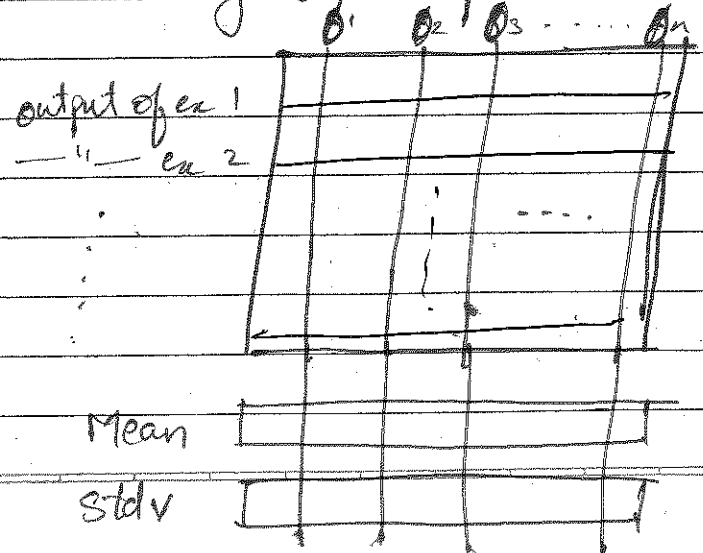
$$\mu_j = \frac{1}{q}\sum_{i=1}^{m} z_j^{(i)}$$

$$\sigma_j = \left(\frac{1}{q}\sum_{i=1}^{m}\left(z_j^{(i)} - \mu_j\right)^2\right)^{1/2}$$

$z_j^{(i)}$: output of j-th unit for ith batch example

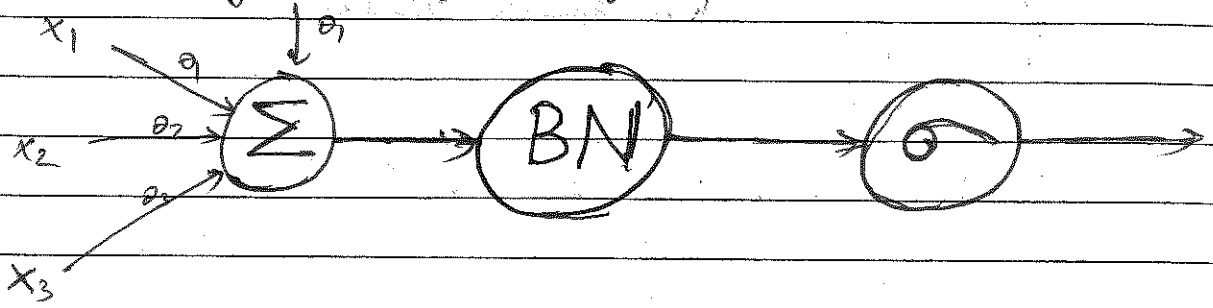
- Makes sure activations are not saturated
- Normalization values are computed for each batch in training
- Normalization is differentiable (suitable for backpropagation).

→ Layers output matrix :

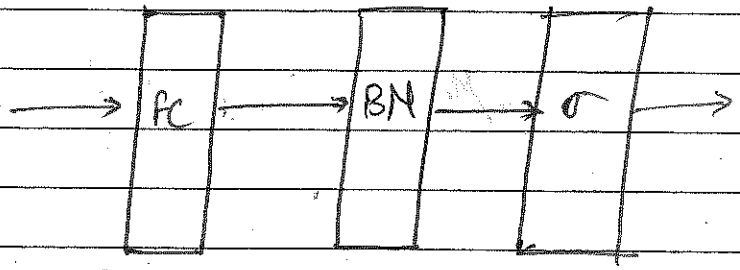


m = D.mean (axis=0)
D = D - m
s = D.std (axis=0)
D = D/s

→ Usually normalize before activation:



→ layers:



FC = fully connected or convolution layer without activation

σ = Normalize activation.

- Sometimes applied after activation

- B.N is not needed after every layer
- In layers with weight sharing (eg. convolution) batch norm^n for different spatial locations need to be the same.
  → use one mean and stdv per output channel.
- Some saturation is needed to terminate learning
  → may need scale and shift after normalization

→ Scale and shift parameters:

$$\{z^{(i)}\}_{i=1}^{q} \longrightarrow \{\hat{z}^{(i)}\}_{i=1}^{q} \longrightarrow \{\tilde{z}^{(i)}\}_{i=1}^{q}$$

$$\tilde{z}_j^{(i)} = \gamma_j \hat{z}_j^{(i)} + \beta_j$$

compute gradients wrt γ_j, β_j

- γ_j and β_j are learned. The network can learn to con B.N if there is no need for it, eg:

$$\gamma_j = \sigma_j \Rightarrow \tilde{z}_j^{(i)} = \sigma_j \hat{z}_j^{(i)} + M_j = \sigma_j \frac{z_j^{(i)} - M_j}{\sigma_j} + M_j = z_j$$
$$\beta_j = M_j$$

- During training because batches are random BN calls randomness into the training and so reduces overfitting

- During prediction use the average normalization values computed during training (i.e. average from all training batches).

✳ Ensemble Classifiers:

- train multiple independent models
- use majority vote or average during testing
- reduces overfitting
- to obtain multiple models:
  - change data
  - change parameters
  - Record multiple snapshots of the model during training.

Page No.
Date

*Summary:

- Start without regularization → see that can overfit
- add Batch normalization
- add dropout if needed
- add decay if needed
- augment data if needed
- create ensemble if needed.

} if observing overfitting

Hyperparameter Optimization

* Principle:
  - select a set of parameters
  - train on training data and test on validation data using a few epochs.
  - Modify parameters in search of better validation error
  - If loss is increasing (eg. Xz) stop.

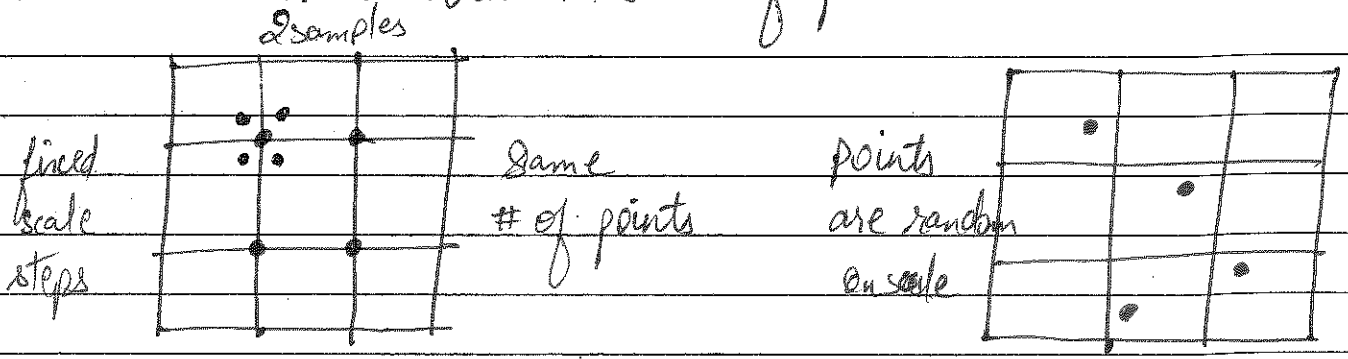
* Methods:
  - coordinate search
  - grid search
  - Random search
  - Bayesian model-based optimization
  - Evolutionary

→ Search procedure: (coordinate search)

- Hold all parameters fixed and change one
- Change parameter on a logarithmic scale to cover both small and large values (eg x2 or x10 val)
- Once identifying best parameter search around it (coarse to fine search) using more epochs.
- If best parameters are found at end of range extend range.
- Repeat several passes on all parameters.

→ Grid and random search:

- search combinations of parameters



grid search

random search generally pref.

In real we want to quasi-random sampling (low discrepancy sequence)
→ random coverage

⁂

Page No.
Date

* Optimize parameters in this order:

- Learning rate (value, decay, update procedure)
- regularization (batch norm, dropout, ...)
- Network architecture and network size
- Other:
  - optimizer
  - optimizer parameters
  - Activation
  - Initialization

Page No.
Date

GPU Frameworks

* training is time consuming → use GPUs

* Frameworks :

By Google { Tensorflow
Keras

By FB { PyTorch
Caffe

Introduction to Keras

Page No.
Date

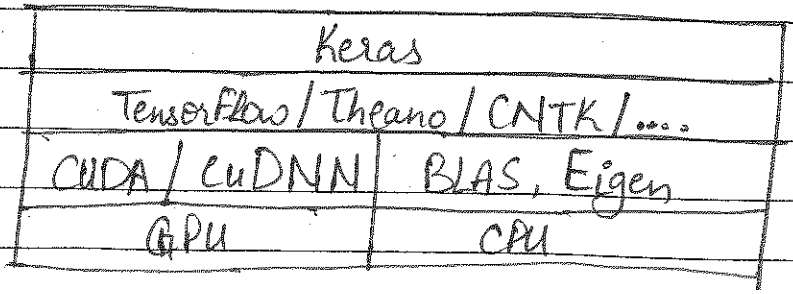
Use google - colab

→ Keras is a Deep-learning framework for Python

→ Key features:
- allows the same code to run seamlessly on CPU or GPU
- User - friendly API
- built-in support for convolutional networks
- built-in support for recurrent networks

→ Supported architectures:
- multi-input
- multi-output
- layer sharing
- model sharing
- arbitrary network architectures

Keras Architecture
- Model-level library, providing high-level building blocks
- low-level operations such as tensor manipulation and differentiation rely on backend engines.
- Run seamlessly on cpu or GPU (through tensor flow which wraps access to cuDNN and BLASS)



Page No.
Date

Keras Workflow

tensors → nD (data structure)

- outline
  - → define training data : input tensors and target tensors
  - → Define a model : network of layers that maps inputs to targets
  - → Configure the learning process (loss function, optimizer, and metrics monitor)
  - → train using fit() and search hyper-parameters

- Define a model
  - Model types
    - → sequential class - linear stacks of layers (most common)
    - → functional API - supports arbitrary architectures (eg. DAG). Specify function like layers to process the data.
  - The remaining steps do not depend on model architecture

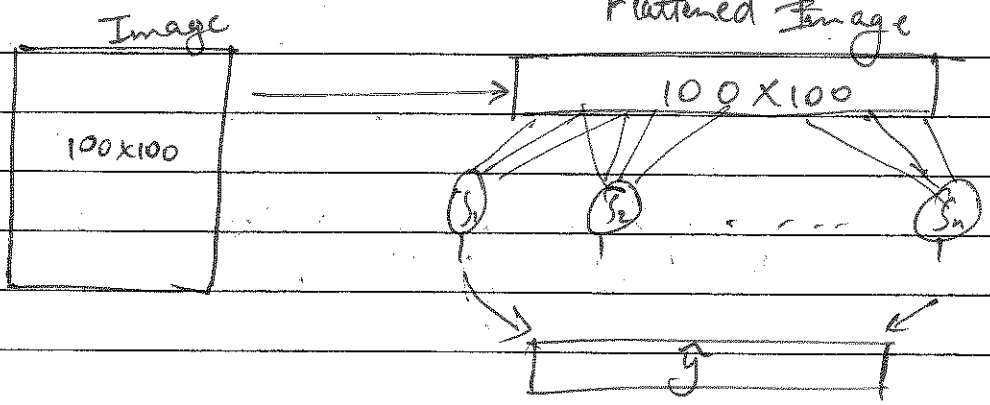
- Compile the network
  - Configure the learning process
    - optimizer
    - loss function(s)
    - metrics to monitor during training

- Learning
  - fit using Numpy arrays of input data and the corresponding target data

Lecture 18

Convolutional Neural Ne

Flattening an Image :



$$\hat{y} = \begin{bmatrix} w_1^T \\ \vdots \\ w_n^T \end{bmatrix} x$$

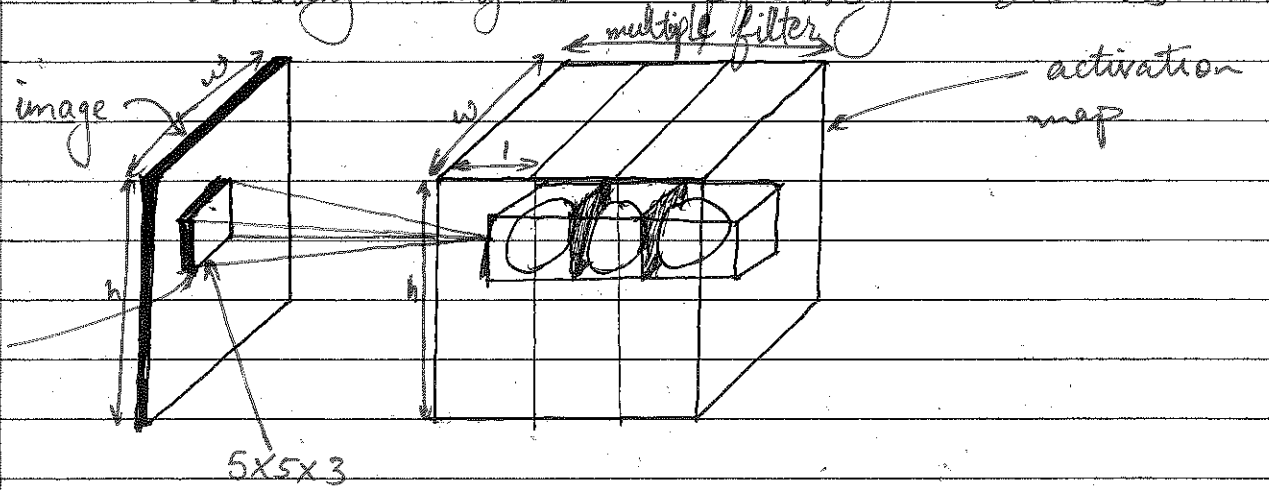
This is an issue in real world. Bigger input means more complex network.

- Problems with flattening :
  1. A lot of coefficients for each unit
  2. loss of spatial content (lost neighborhood)
  3. shift invariance (same object at different locations.)

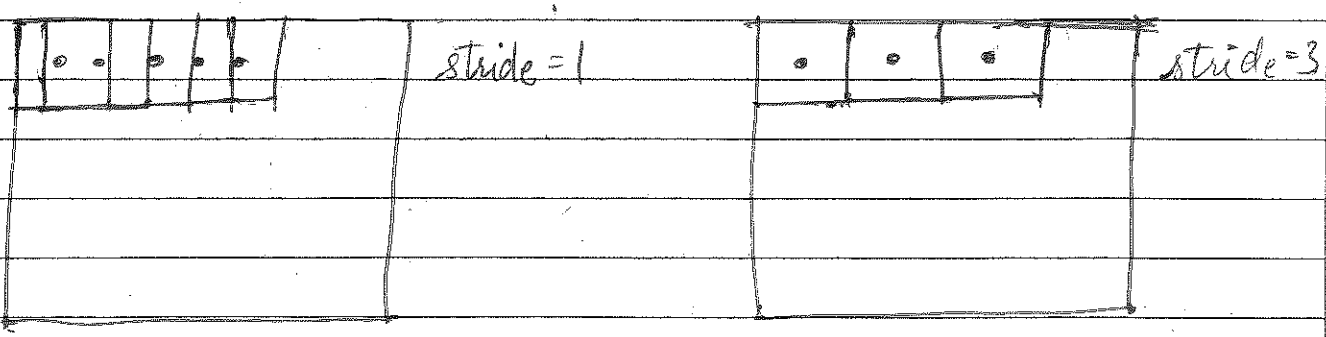
Solution is to use convolutions &.

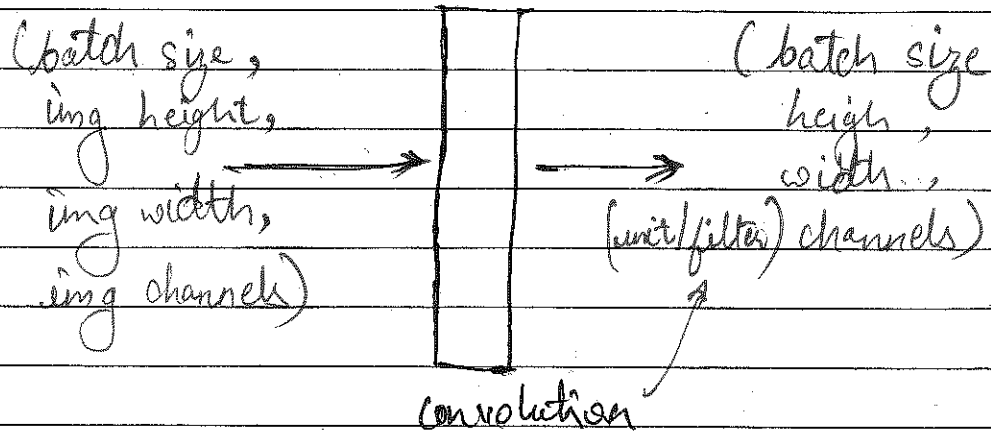
Convolution Layers

- convolve the image with small filters eg. 3x3 or 5x5
- share weights of filter between locations (shift invariance)
- sharing weights effectively increases data.



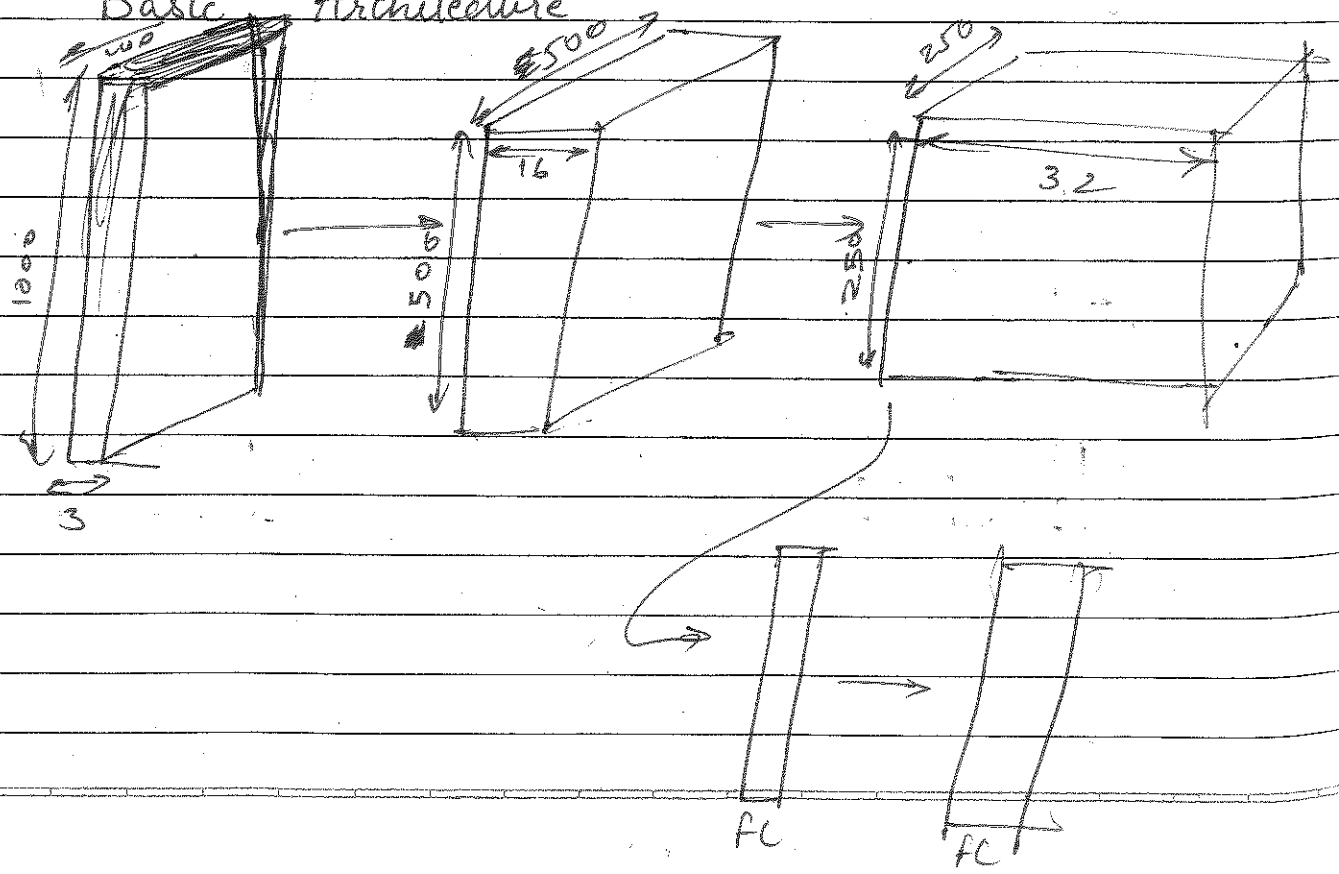
- extract image patches (windows)
- vectorize image window and filter and perform dot product (plus bias).
- Filter extends full depth of image
- Multiple convolution filters per location (eg oriented edges)
- Use stride to move filter → activation map may be smaller.





- Multiple layers: as spatial dimensions, decreases, depth increase to compensate for reduced coefficients (keep the same number of coefficients).
- Non-linear activation and sampling between layers
- Final FC layers (classification after feature extraction)

Basic Architecture

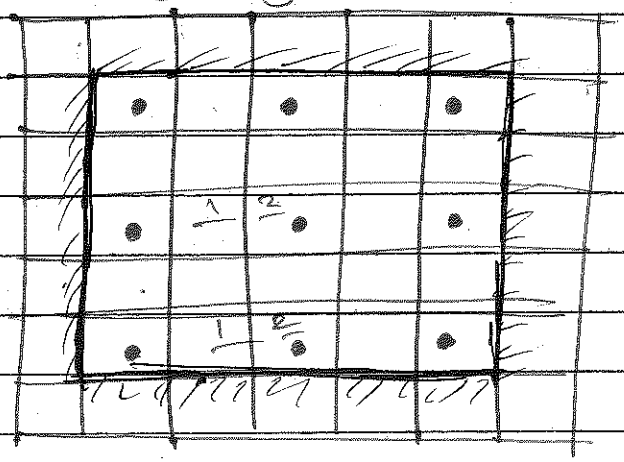


Page No.
Date

* Layer dimensions:
  - zero padding is needed to prevent unwanted shrinking, especially with deep networks.
  - width and height of the output of a layer depend on stride. With a stride of 1 layer overlap of receptive fields and large objects.

* 1×1 convolution:
  1×1 convolution is used to reduce dimensions (i.e. lower # of channels).

* Dilated convolution (Atrous convolution):
  - Increase receptive field without increasing the number of parameters
  - we consider value at ~~dot~~ red dot represents the neighboring values.



2 dilated

Instead of 1D convolution ordinary:

$$(I * K)(t) = \sum_{\tau} I(t-\tau) K(\tau)$$

Use:

$$(I *_l K)(t) = \sum_\tau I(t - l\tau) K(\tau)$$

($*_l$: l = dilation factor; $l\tau$: take steps of size l in image)

- take steps of size l in image when performing the convolution.

* Pooling

pooling = down sampling spatial dimensions (depth unchanged)

- strategies:
  - → Max pooling (usually 2×2) – partition non-overlapping regions & choose max in each region.
  - ⇒ Alternatives:
    - average pooling
    - L2 norm pooling
    - RoI pooling (output size is fixed and input variable)

- Pooling is done by scanning with a filter with stride – often stride is selected without filter overlap

- Convolution stride can also be used to downsample but this is a random
  - normally pooling instead of stride in convolutions
  - retains more info

- pooling has no weights, while convolutions do have; so no learning.
- Depth dimension is normally not pooled.
- Pooling supports multiple scale analysis (pyramid)
- Pooling helps in reducing the number of parameters coefficients.
- The amount of pooling is choice (hyper-parameter)

(Implementation in lecture)

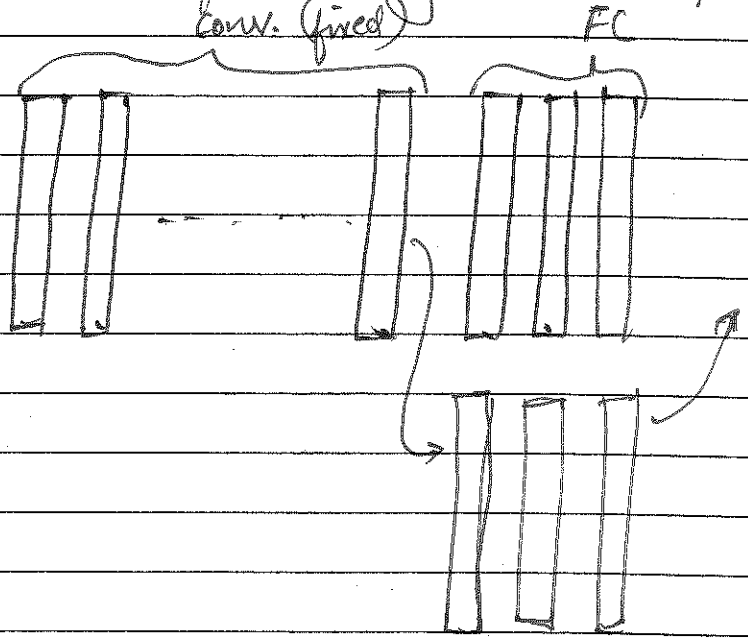
Lecture 19 - W10L1

(CNN contd.)

* To advis the problem of limited data:

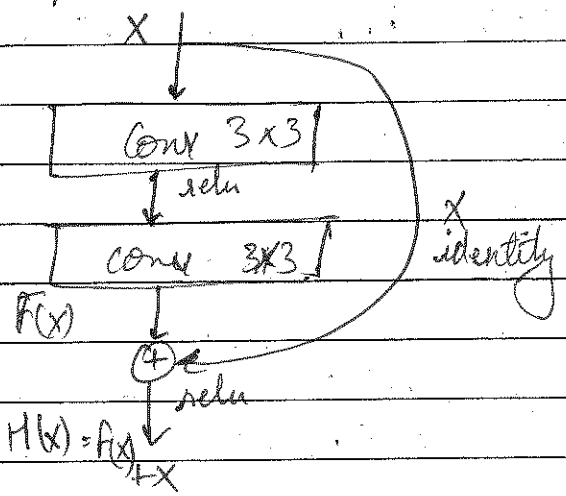
1) data augmentation: add examples with perturbations (eg rotation, flip, contrast change).

2) Transfer learning: use pre-trained convolution base



* Networks that are pretrained & can be used:
1) VGG (16/19)
2) Google Net / Inception (22 layers + fewer parameters)
   - To reduce lot of parameters due to inception, they use 1x1 parameter.
   - Convergence can be difficult
   - Use auxiliary classifiers to help with vanishing gradients (not needed with batch norm).
3) Res Net (uses residual block):

- Concept:



- Easier to learn $F(x)$ residual compared with $H(x)$ (learn deviation from identity instead of function)

- Skip connections help with vanishing gradients.

4) Mobile Nets:

- for mobile and embedded devices.
- tradeoff b/w latency & accuracy
- Use depth-wise separable convolutions:

* separable convolutions - separate convolution into :-
  - depth wise (channel) convolutions, followed by
  - point wise 1x1 convolutions

* Simplification parameters

- width multiplier (fewer channels) [α]
- resolution multiplier (smaller resolution) [β]

[More in notes]

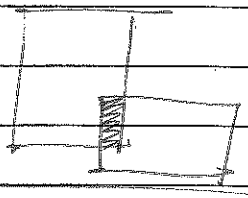
Page No.
Date

☆ M AND SEGMENTATION (using M.N.)

* [illegible]

→ IoU / Jaccard index (similarity)

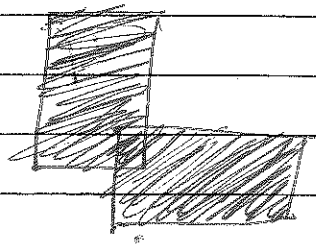
$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{(A \cap B)}{|A| + |B| - |A \cap B|}$$



mutually exclusive

$$IoU \in [0, 1]$$

some



→ Jaccard distance

$$d_J(A, B) = 1 - IoU(A, B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

Sub(*) Mean Average Precision (MAP)

- Commonly used to evaluate object detection
- whereas IoU only takes into account localization accuracy, MAP measures classification accuracy of detected object.

→ precision and recall:

pred \ true labels	P	N
P	TP	FP
N	FN	TN

$$\text{recall} = \frac{TP}{\text{total positive}} = \frac{TP}{TP+FN}$$

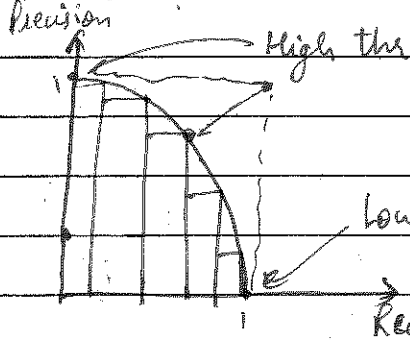
$$\text{Precision} = \frac{TP}{\text{Total P predictions}} = \frac{TP}{TP+FP}$$

→ tradeoff between precision and recall:

- higher confidence threshold – higher precision
  – lower recall
- Lower confidence threshold – lower precision
  – higher recall

sub(*) ROC Curve:

Record precision & recall at different confidence levels.



AUC = Area under curve

AUC ∈ [0, 1] larger is better

AP = average Precision (computes AUC)

r_i recall at threshold t_i

P_i precision at threshold t_i

$$AP = \sum_{\text{confidence threshold } t_i} (r_{i+1} - r_i) P_{i+1}$$

Lecture 20 - W10L2

(Detection evaluation Contd.)

* MAP implementation:

  - take AP for each class object then take the average for all classes.
  - Include (no object) background as an object.

* IoU threshold

- In the context of detection, to compute AP we first have to select a IoU threshold to determine these was a detection (then use AP to measure correctness).
- What IoU threshold should we select?
  - lower IoU threshold - more detections but with worse localization and possibly less accurate.
  - higher IoU threshold - fewer detection but with better localization and possibly more accurate.

→ COCO map:

- calculate map at different IoU threshold values and average them.

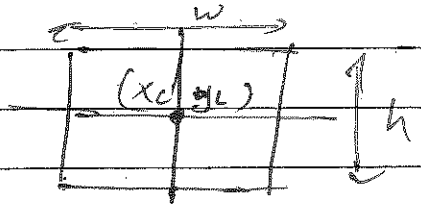
AP @ (0.5 : 0.05 : 0.95)

MAP = $\frac{MAP_{0.5} + \cdots\cdots + MAP_{0.95}}{10}$

* Object localization

- to localize an object we need to specify a boundry box (x_{min} , y_{min} , x_{max} , y_{max})
- because we process the image at different scales, absolute coordinates cannot be used → use relative locations → $\in [0, 1]$
- Alternatively use

$(x_c , y_c , w, h) \longleftrightarrow (x_{min}, y_{min}, x_{max}, y_{max})$



* Training :

- For each example mark bounding box and label class
- Include background (no object) boxes.
- One-hot encoded class labels.

$$y = [\underbrace{p_c}_{\text{1 if object box, 0 if not object box}}, \underbrace{x_c, y_c, w, h}_{\text{bounding box}}, \underbrace{c_1, \ldots, c_K}_{\text{one-hot encoded class}}]$$

* Classification:

$$\hat{y} = [\underbrace{p_c}_{\text{obj prob}}, \underbrace{x_c, y_c, w, h}_{\text{bounding box}}, \underbrace{c_1, \ldots, c_K}_{\text{class probabilities}}]$$

Page No.
Date

* Loss:

$$L = L_{reg} + L_{cls}$$

L_{reg}: regression, eg. cross entropy (has to do with box)

L_{cls}: classification (has to do with img inside box) eg L2.

$$L_i\left(y^{(i)}, \hat{y}^{(i)}\right) = \begin{cases} \left(y_1^{(i)} - \hat{y}_1^{(i)}\right)^2 & \text{if } y_1^{(i)} = 0 \\ \sum_j \left(y_j^{(i)} - \hat{y}_j^{(i)}\right)^2 & \text{otherwise} \end{cases}$$

use cross entropy for class labels

* Multi object detection:

* → Approach:
1. find candidate
2. classify candidate
3. possibly refine candidate

→ Finding candidate:
- Sliding window
- grid cells
- Region proposal

Page No.
Date

* Sliding window detection

→ slide window through image.
① crop sub window
② classify each sub-window

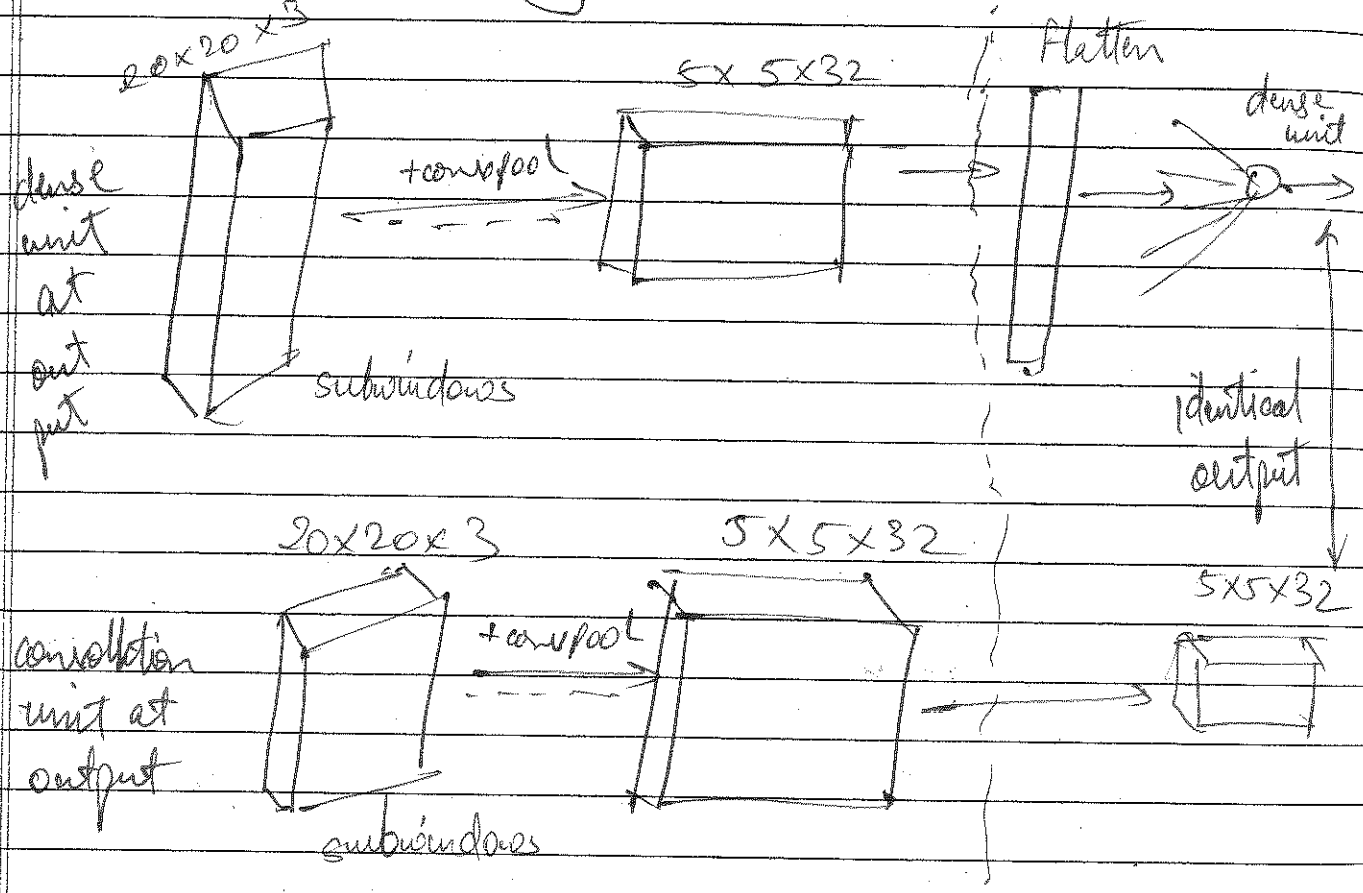
→ To deal with different object scales, crop different sub-windows with different sizes and aspect ratio at each location.

→ May be slow - many windows to classify.

→ problems:
- running dense layers for classifying each ~~sub-wind~~ subwindow is inefficient.
- in addition dense layers need a fixed input size and so the ~~input~~ size need also be fixed.

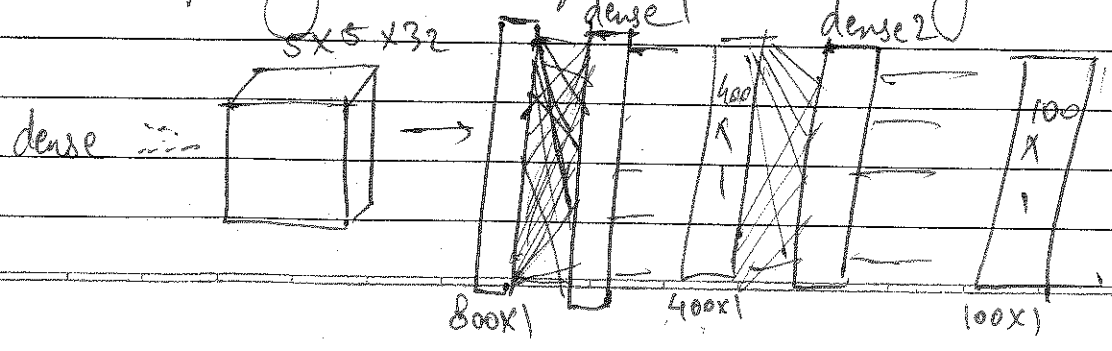
→ solution (fully convolutional implementation):
- A FCI is more efficient: no need to process each sub-window separately and convolution can be implemented effeciently.
- A FCI can work on variable image size.

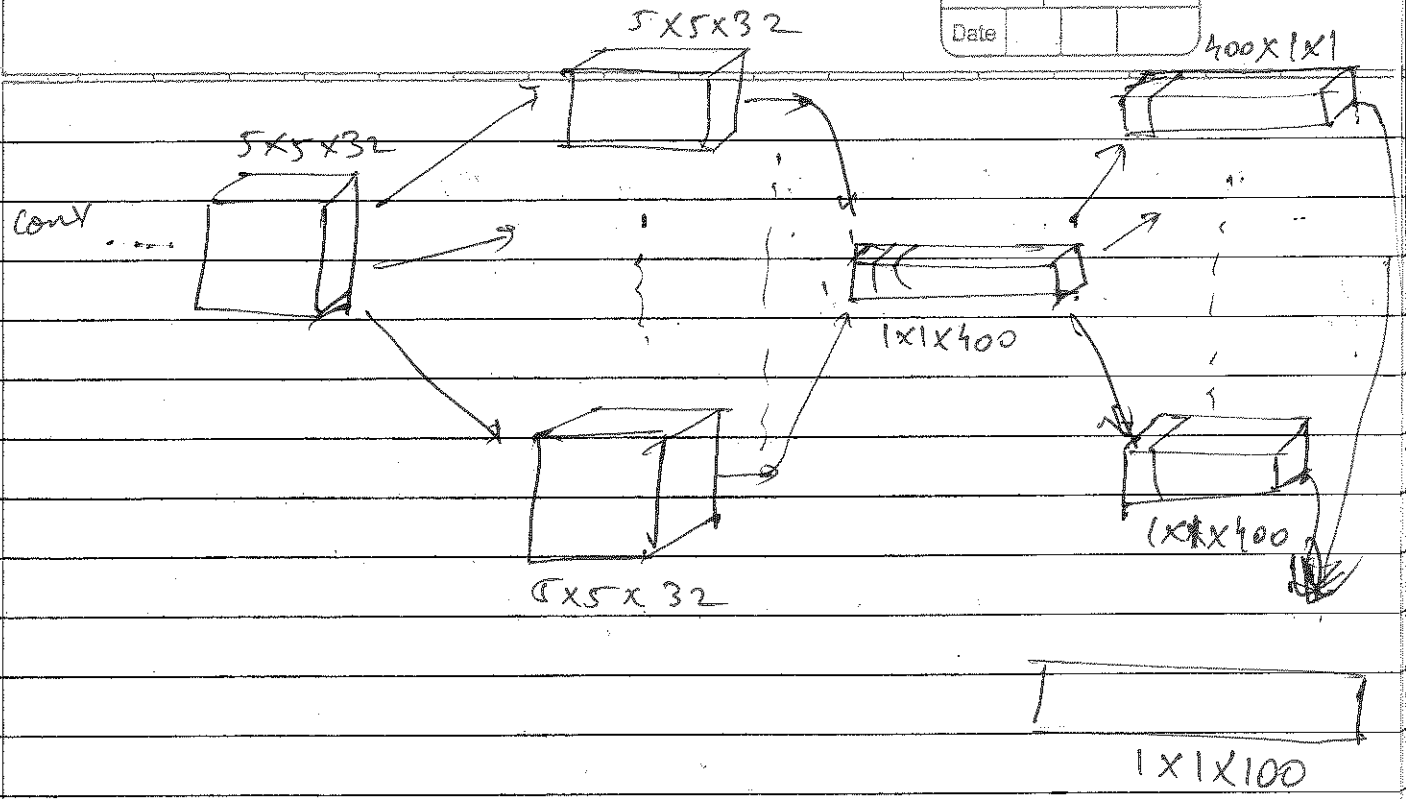
→ A single fully connected (dense) unit may be replaced by a single convolutional unit that has the dimension of the feature tensor (without flattening)



→ consider replacing 400 dense units
- output is 400x1 vector
- we can replace 400 units with 400 conv. filters
- their output will be 400 x 1 vector

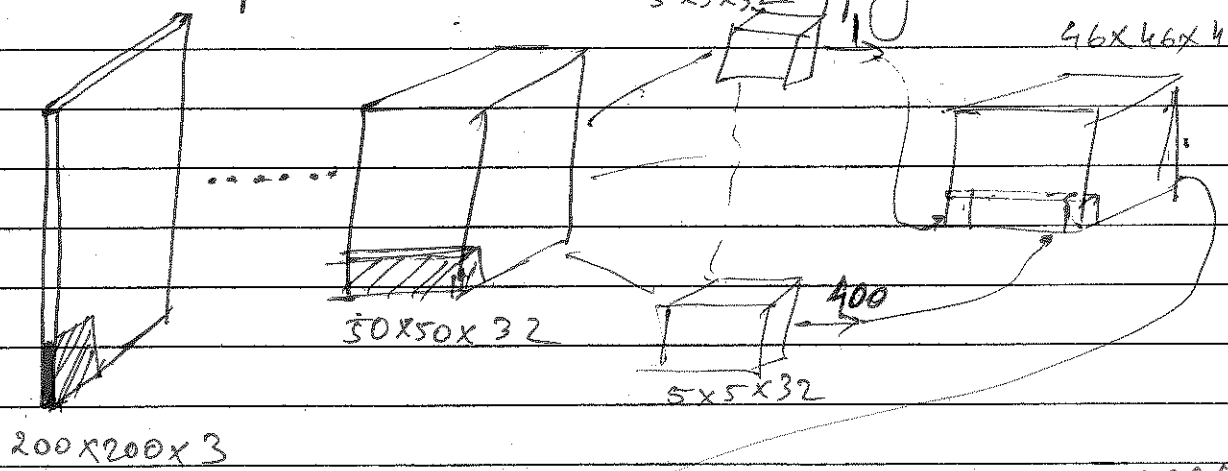
→ replacing a subsequent dense layer with 100 units



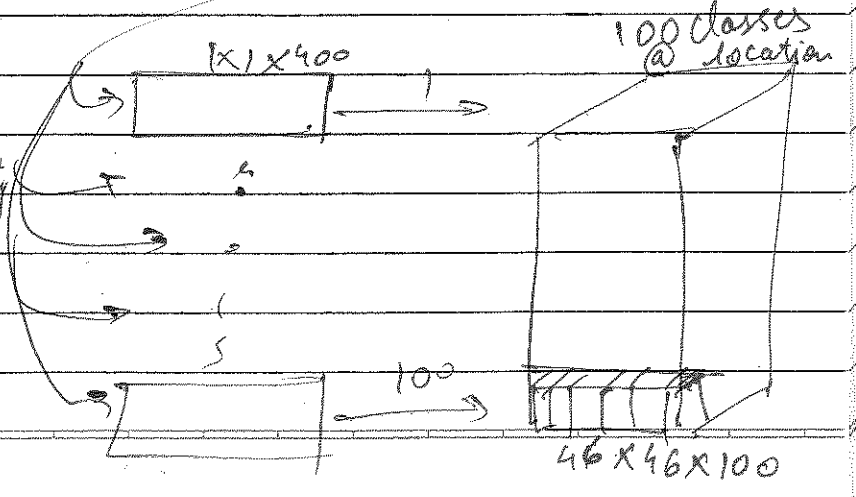


→ So far a single subwindow classification.

- for a single sub-window no reduction in computation
- The advantage of convolutions is when sharing computations between overlapping windows.



- each column at output provides probability of a class at a window.



Page No.
Date

* Summary: (sliding window detection)

- use conv + pooling to extract features and reduce spatial resolution.

- each location in reduced resolution layer corresponds to 'virtual' patch in original image. virtual patches overlap.

- use dense layer or 1x1 conv to classify or regress at each location in reduced resolution layer.

* Problem (SWD):

- doesnot produce accurate detection because:
  - It is based on classification in a pooled (lower resolution) layer.
  - We detect fixed size boundary boxes

* Solution (SWD):

To solve problem we can, in addition to classifying each location, regress a bounding box there.

Lecture 21 - W11L1

Page No.
Date

★ Grid Cell Detection:

- Divide the image into grids and attempt to scan an object in each grid cell
  - cells do not necessarily match objects and so use regression to find a bounding box for each detected object
- similar to object localization (single object) except that now at each grid cell detect and localize objects
- at each grid cell train and predict using:

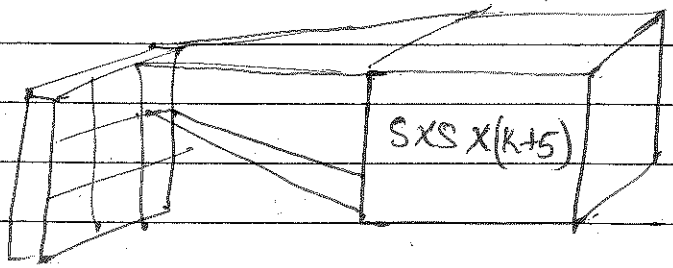
$$y = [\underbrace{p_c}_{\text{classification (object yes/no)}}, \underbrace{x_c, y_c, w, H}_{\text{regression (bounding box)}}, \underbrace{c_1, c_2, \ldots c_K}_{\text{classification (object class) } P(c_1)\ P(c_2)\ \ldots\ P(c_K)}]$$

$\hat{y}$ for prediction

- x_c, y_c, w, h are relative to each grid cell $x_c, y_c \in [0, 1]$ whereas w and h may be bigger or smaller than 1 for boxes larger than a cell.
- for $S \times S$ grid and K classes the output is a tensor of size:

$$S \times S \times (K+5)$$

$5 \rightarrow$ 4 bounding box & 1 class prob.



- at each grid cell with object a precise bounding box instead of a fixed window.
- only on object at each grid cell.

* Non-max supression (NMS):

Since we might get multiple object detections for adjacent grids when the object is large we use NMS.

- select detection in grid cell with highest score (probability Pc) and delete detections that have IoU $> \tau$ with selection.
- perform NMS for each class separately to allow object overlap.

Intersection over union

- problems (NMS):
  - top scoring box may not be the best fit
  - It may suppress nearby objects.
  - It does not suppress false positive

* Problems (with grid cells):

1) what should be grid size?
   - Too small will fail in classify big objects and vice versa.
2) what should be the aspect ratio of candidate cells?
   - different objects have different aspect ratio of boxes that should be used for their detection
3) How to detect more than one object in each

candidate cell?

Solution: Anchor boxes
- to solve the three problems, use multiple anchor boxes at each grid location.
- Anchor boxes have different sizes and different aspect ratios (select according to what we see in training data).

- Multiple anchor boxes (multiple objects at same grid):
  - To allow detecting two different (overlapping) objects at each grid location (eg objects with different aspect ratios) double the target vector y.

$$y = [p_c^1, x_c^1, y_c^1, w^1, h^1, c_1^1, \ldots c_k^1; p_c^2, x_c^2, y_c^2, w^2, h^2, c_1^2 \ldots c_k^2]$$

$p_c^1 = 0$ and $p_c^2 = 0$ if object does not exist

  - most grid cells will have zero or one anchor boxes.

- Anchor box assignment during training:
  - During training, the anchor box assigned to an object is the one having highest IoU with ground truth box.

* Two classes of methods for object detection:
- Single shot v.s. double shot methods:
  - Single shot detectors perform object detection in a single shot network that looks at the image once.
  - two shot detectors look at the image twice: Once

to propose object regions (region proposal) and a second to refine and classify regions.
- two shot methods have higher computational cost but may be more accurate (lower frame rate).

• Single shot methods:
- consider predefined anchor boxes at grid cells
- refine and classify region proposals

• two shot methods:
- consider pre-defined anchor boxes at grid cells
- classify them as object or not → region proposal
- refine and classify region proposals.

• single shot detectors (popular):
- You only look once (YOLO)
- Multibox
- Single shot multi shot detect (SSD)

→ Yolo algorithm:
- divide image to SxS grid cells. A grid cell is responsible for detecting objects with centers in it.
- Each grid cell predicts B bounding boxes and confidence scores. (confidence that the box contains the object and is accurate).

confidence = p(objects) + IoU(truth, prediction)
confidence = 0 if no object in cell.

[Loss in ppt]

→ Multibox:

- use prior for anchor boxes determined per class
- Inception-like detection (detect at different scales).

- Loss:
  - confidence loss - class categorical crossentropy
  - location loss - ~~dist~~ bounding box IoU

so, Multibox loss = confidence loss + α * locationloss

→ Single shot multishot detect (SSD):

- SSD = YOLO + multibox

- use feature maps from intermidiate convolution layers.
- Use different prior for different convolution layers.
- Apply ~~to~~ NMS.

- Use hard negatives (False positive by the models) to train model instead of all negatives.
- this helps balancing negatives & positive examples.

Page No.
Date

- Two shot detectors :

RCNN — region detection CNN
  └ region proposals without network. SVM classifier.
Fast-RCNN — CNN classifier
faster-RCNN — use region proposal network
R-FCN — fully convolutional (no depth layer)
Mask RCNN — add object segmentation (instance segmentation)

Lecture 22 - W11L2

(contd.)

→ RCNN :-

- Instead of processing candidates on a grid use superpixels (select search algo) to find candidates.
  - Extract 2000 proposals

- Advantages :
  - candidates have different aspect ratios and scales
  - fewer candidates

- save and process each candidate separately (slow).

→ Fast - RCNN :-

- Convolution implementation instead of processing one candidate box at a time.
- Faster (x 200) than RCNN but slower than YOLO.

- fast RCNN architecture : [refer notes]
  - ROI pooling - use max pooling to convert the region of interest to a fixed size (ex. 7x7) to the FC classifier.

region of interest

→ faster RCNN :

* Region proposal Network (RPN):
  - train CNN to produce region proposals
  - light weight CNN (relatively simple task)

- motivation :
  1) speed
  2) complete end to end & network & train.

- Algorithm (RPN):
  1) slide window and classify (object/noobject)
     use fast convolution implementation (sliding windows)
  2) use K anchor boxes at each location (different size and aspect ratio)
     - around 200K boxes.
  3) classify & regress K-boxes:
     cls - output 2K classification scores
     reg - output 4K regression boxes.

     Anchor boxes = W X H X K

  4) NMS - select candidate with highest confidence and delete candidates overlapping (IoU > T) with it.

• faster RCNN = fast RCNN + RPN
  - use RPN for region proposal
  - use ROI - pooling then classify using dense layers

- Faster RCNN loss:
  - RPN [classification (obj) + box regression] loss
  - classification [classification (class) + refine box regression] loss.

  bounding box losses use ℓ_2 loss & classifications use cross entropy.

- Faster RCNN algorithm:
  1) get pretrained CNN (eg VGG16 or ~~resnet~~ resnet 101)
  2) get feature map from last (deep) conv. layer.
  3) ~~train~~ apply RPN: obj / no obj and box regression.
  4) Apply RoI-pooling to proposals
  5) Use FL (dense) layers for class. classification and refined box regression.

→ RFCN (region based fully convolutional Network):

- Use convolutions, instead of costly dense layers that are applied for each region)
- events position sensitive-score maps where each score map is sensitive to another region

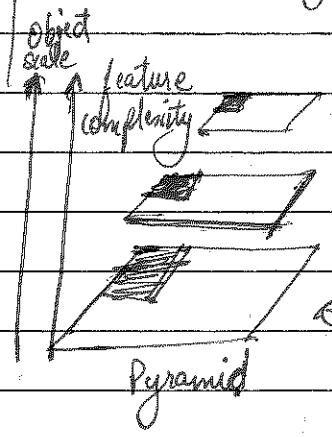
  KxK grid → K^2 score maps

  each score map has C+1 channels (C classes and 1 obj/no obj class)

- position sensitive RoI pooling: [refer diagram]
- RoI architecture [refer diagram]

→ mask RCNN: (also does instance segmentation)

* Feature pyramid network (FPN):
  - goal - Detect objects at different scales using higher level features.

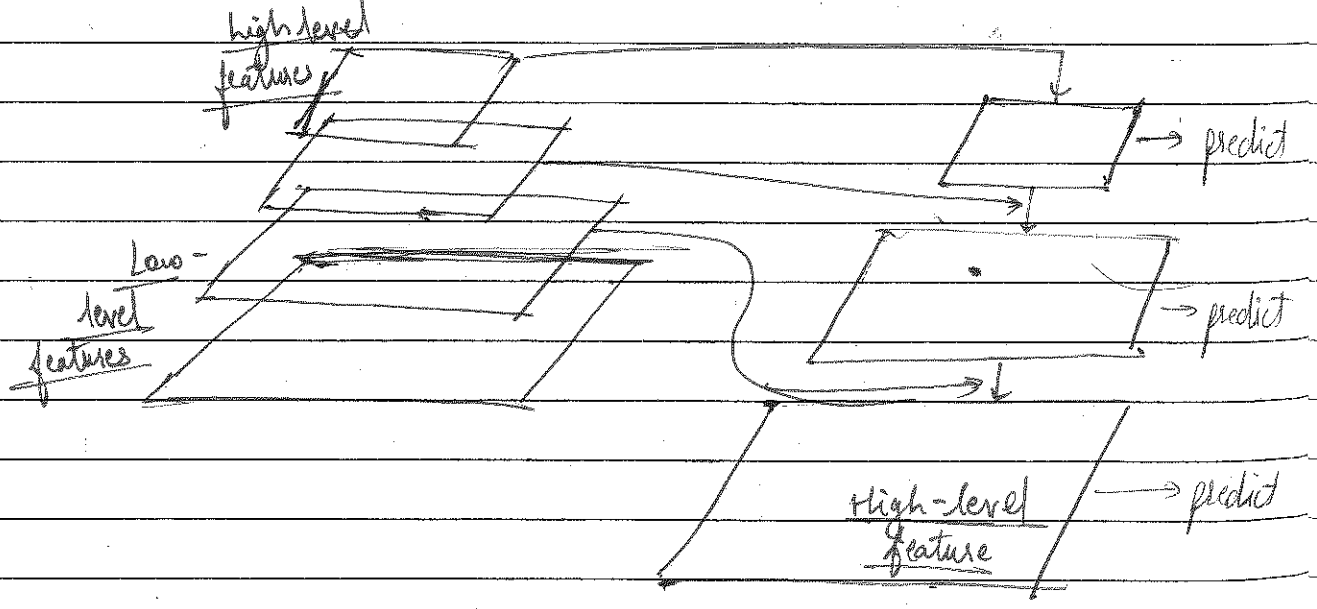


← detect large scale objects using high level features

← detect small scale objects using low level features.

But we want to detect both large and small scale objects (solving FPN.)

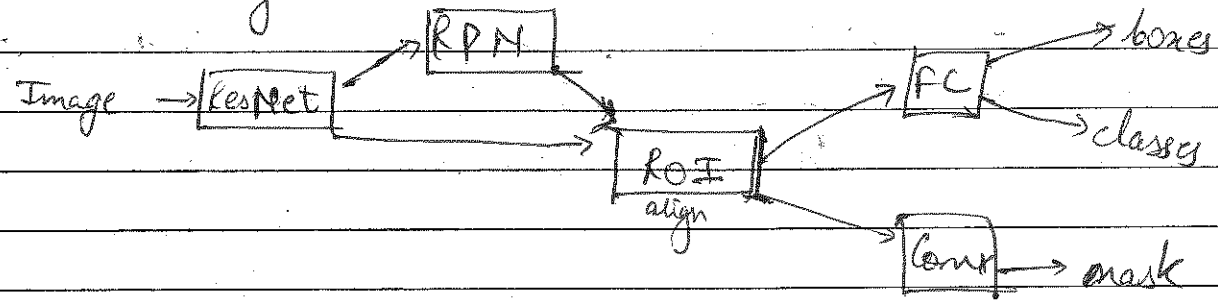
sol": lateral connections in FPN.



- maintain strong semantic features at each level.
- lateral connections combine features at corresponding levels.

x

- mask RCNN adds segmentation to faster RCNN instance segmentation

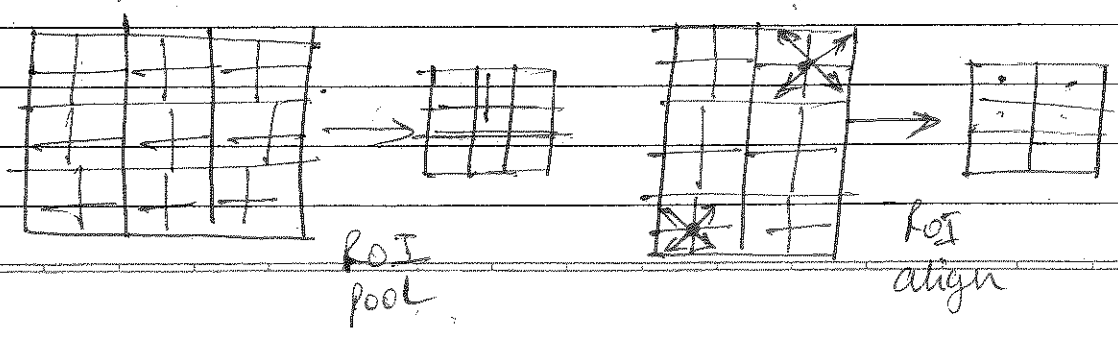


$$loss = L_{cls} + L_{reg} + L_{seg}$$

$$L_{cls} = L_{cls}^{RPN} + L_{cls}^{obj}$$

$$L_{reg} = L_{reg}^{RPN} + N_{reg}^{obj}$$

- predicted masks are scaled up to original resolution.
- masked-RCNN architecture [refer diagram]
- RoI align:
  - RoI pooling extracts small feature maps of fixed size.
  - RoI-pooling is sensitive to alignment - shifting to grid will change results.
  - RoI-align solves this by sampling with interpolation



Page No.
Date

☆ Semantic Segmentation:

- Segment (cluster) according to class (eg. road)
- density each pixel
- Evaluate using mean IoU (of masks)

* Methods
- FCN (Fully convolutional segmentation)
- U-Net: add skip connections between encoders and decoder branches.
- Deeplab: add spatial pyramid pooling (spp)

→ FCN

- Uses convolutions to extract features then classifies
- 32, 16, 8 architectures.

→ UNet
- Uses encoder-decoder architecture

  - Encoder → convolve & pool (depth doubles)
  - then feature extract
  - decoder then at end classifier
  - Uses lateral connections to decode without loss.

- Skip connections
  - skip connections between encoder & decoder
  - upsample output is concatenated with cropped input (cropped due to loss of pixels in conv)

- Fully convolutional:
  3x3 convolution : produces segmented output (proparity). The number of filters in final convolutional layer = number of classes.

→ DeepLab :—

* Spatial pyramid pooling (SPP):

goal: support objects at different scales

multiple pooling levels as input to the classification layer. [refer diagram]

- spatial pyramid pooling results in increased dimensions → use atrous (dilated) convolution
- This results in atrous spatial pyramid pooling (ASPP)

— x —

- Deeplab uses encoder - decoder architecture
- Use Atrous spatial pyramid pulling (ASPP) with skip connections. [refer diagram]
- convolution implemented as depth wise separable convolutions (as in mobilenets).

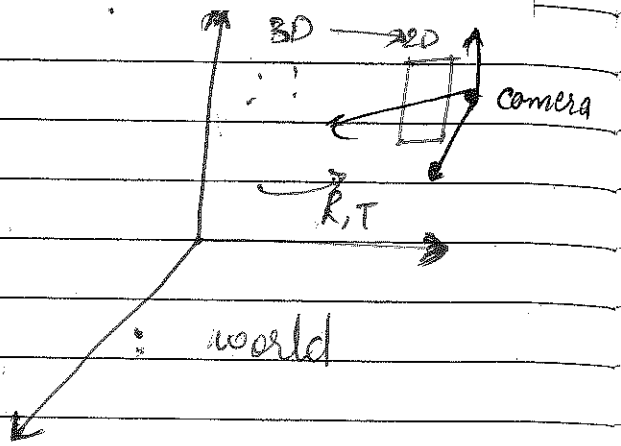
- Deeplab architecture [refer diagram]

☆ Camera Calibration

* Recall:

$$p_i = M \, P_i$$

- p_i: image point (2DH) 3×1
- M: projection matrix (3×4)
- P_i: world point (3DH) 4×1



Question?
- ① what is M?
- ② How to go 2D to 3D?

* Definiti of M (already learnt):

$$M_{3\times4} = K^*_{3\times3} \, [\, R^*_{3\times3} \mid T^*_{3\times1} \,]$$

- K^*: intrinsic parameters
- R^*, T^*: extrinsic parameters

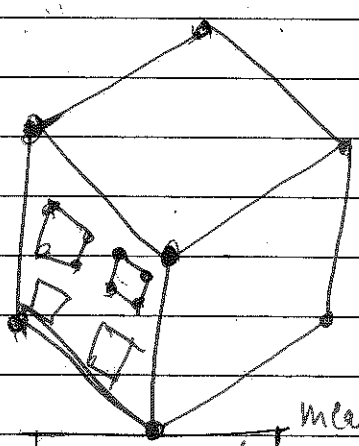
$$K^* = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

- $\alpha_u = f k_u$
- $\alpha_v = f k_v$
- $s = \alpha_u \, tg\,\theta$

* Calibration

$M \rightarrow K^*, R^*, T^*$

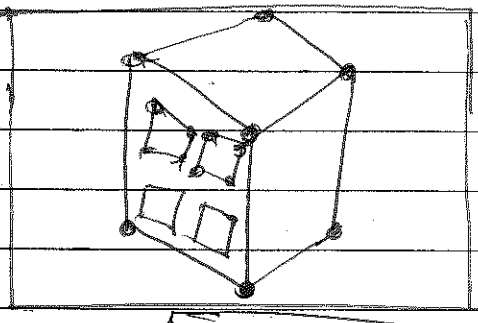
$\rightarrow \underbrace{R, T}_{R^*}, \underbrace{f, K_u, K_v, u_0, v_0, t_y\theta}_{K^*}$



3D object

measured

	x	y	z
3D world coord	3.2	1.7	1.8
	⋮	⋮	⋮



2D Image

	x	y
2D image coord	20.0	50.0
	⋮	⋮

Input for calibration alignment

given corresponding point $\underbrace{\{P_i\}_{i=1}^{m}}_{\text{pixels}} \longleftrightarrow \underbrace{\{P_i\}_{i=1}^{m}}_{\text{meters}}$ find parameters

Lecture 23 - 10121

(contd)

precision: order of magnitude larger than desired.

* Calibration approaches:

methods:
- planar (require 2D) (requires multiple views)
- non-planar (eg CUBE)

approaches:
- direct
- indirect (find M then parameters)

- During calibration we need to solve
  (A is not square)

$A^TAx = 0$

solution = zero eigenvalues of A^TA

Or

using SVD (singular value decomposition): $A = UDV^T$

solution = column of V belonging to zero singular value of A.

(right null space / SVD)

$$A_{m \times n} = U \; D \; V^T$$

U: m x m orthogonal; D: m x n diagonal; V^T: n x n orthogonal

properties :

1) columns of u are orthogonal
2) columns of v are orthogonal
3) D is diagonal with singular values on diagonal
4) columns of u are eigenvectors of AA^T
5) columns of v are eigenvectors of A^TA
6) a matrix is singular if $\sigma_0/\sigma_n < \tau$ (condition number)
7) for a symmetric matrix $u = v$ and singular values are squares of eigenvalues

So question is what is x in $A^TAx = 0 \Rightarrow x = ?$

① zero eigenvector of A^TA

② column of V belonging to zero singular value of A.
(right null space)

So given $\{P_i\}_{i=1}^{m} \longleftrightarrow \{p_i\}_{i=1}^{m}$ find camera parameters

steps :

① find projection matrix M
② find parameters from M

Estimating the Projection Matrix

2DH 3x4 3DH

$$p_i' = M P_i$$

Projection Matrix

Known

$$p_i' = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$ pixels

$$P_i = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$ world

so,

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} -m_1^T- \\ -m_2^T- \\ -m_3^T- \end{bmatrix} P_i$$

$$x_i = \frac{x_i'}{w_i'} \qquad y_i = \frac{y_i'}{w_i'}$$

Homogenized p_i' after projection.

$$p_i' = (x_i^T, y_i)$$

so,

$$\left.\begin{aligned} x_i' &= m_1^T P_i \\ y_i' &= m_2^T P_i \\ w_i' &= m_3^T P_i \end{aligned}\right\}$$

$$x_i = \frac{m_1^T P_i}{m_3^T P_i} \Rightarrow m_1^T P_i - x_i m_3^T P_i = 0$$

$$y_i = \frac{m_2^T P_i}{m_3^T P_i} \Rightarrow m_2^T P_i - y_i m_3^T P_i = 0$$

So,

$$\left.\begin{aligned} m_1^T P_i - x_i m_3^T P_i &= 0 \\ m_2^T P_i - y_i m_3^T P_i &= 0 \end{aligned}\right\}$$

for each point
2 equations with
12 unknown
⇒ need atleast 6 values.

Page No.
Date

for a single point → $[Ax=0]$ form

$$\begin{bmatrix} P_i^T & 0 & -x_i P_i^T \\ 0 & P_i^T & -y_i P_i^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

2×12 (known) $\quad 12\times 1$ (unknown) $\quad 2\times 1$

So for m points:

$$\begin{bmatrix} P_i^T & 0 & -x_i P_i^T \\ 0 & P_i^T & -y_i P_i^T \\ \vdots & \vdots & \vdots \\ P_m^T & 0 & -x_m P_m^T \\ 0 & P_m^T & -y_m P_m^T \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

1st point: $m_3^T P_i = 0$

2nd point: $m_3^T P_i = 0$

mth point

12×1

m = # corresponding points $\quad 2m \times 12 \quad\quad 2m\times 1$

As $Ax = 0$

$A^TAx = 0$ solve SVD to solve using just A.

$A = UDV^T$

Page No.
Date

solution = columns of V belonging to zero singular value.

$$\hat{x} = \begin{bmatrix} \hat{m_1} \\ \hat{m_2} \\ \hat{m_3} \end{bmatrix} \Rightarrow \hat{M} = \begin{bmatrix} — \ \hat{m_1}^T \ — \\ — \ \hat{m_2}^T \ — \\ — \ \hat{m_3}^T \ — \end{bmatrix}$$

the solution is not unique $A\hat{x} = 0$
we also have $A(s\hat{x}) = 0 \Rightarrow s\hat{x}$ is a solution

reason 2DH

$P_i \longrightarrow 3M P_i$

↑ homogeneous

we need to avoid this factor (s) because it changes everything R^*, T^* so on. Since $\hat{M}$ is to be decomposed.

So stepwise:

① $\{P_i\}_{i=1}^{m}$ (2DH) $\longleftrightarrow$ $\{P_i\}_{i=1}^{m}$ (3DH)

⇓

solved system of eq $\hat{M}$

② Take $\hat{M}$ and give parameters

$$\hat{M} \longrightarrow K, R, T$$
$$\uparrow$$
$$f, u_0, v_0, k_u, k_v$$

So now we do ② :-

we find s such that :

$$M \equiv K^*[R^* | T^*] = s\hat{M} \qquad s = ?$$

so we need to find parameters & the scale factor from M.

$$\underbrace{K^*[R^* | T^*]}_{M} = s\hat{M}$$

$$\Rightarrow [K^* R^* | K^* T^*] = s\hat{M}$$

$$K^*R^* = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} — r_1^T — \\ — r_2^T — \\ — r_3^T — \end{bmatrix} =$$

$$= \begin{bmatrix} \alpha_u r_1^T + s r_2^T + u_0 r_3^T \\ \alpha_v r_2^T + v_0 r_3^T \\ r_3^T \end{bmatrix}$$

$K^* \quad R^*$

$$\begin{bmatrix} \alpha_u r_1^T + s r_2^T + u_0 r_3^T \\ \alpha_v r_2^T + v_0 r_3^T \\ r_3^T \end{bmatrix}^T \begin{bmatrix} K^{*T} \end{bmatrix} = s \vec{M} = s \begin{bmatrix} -a_1^T- \\ -a_2^T- \\ -a_3^T- \end{bmatrix}_{3\times3} \begin{bmatrix} b \end{bmatrix}_{3\times1}$$

unknowns — unknown known

① $\alpha_u r_1^T + s r_2^T + u_0 r_3^T = s a_1^T$

② $\alpha_v r_2^T + v_0 r_3^T = s a_2^T$

③ $r_3^T = s a_3^T$

④ $K^* T^* = s b$

Knowns: a_1, a_2, a_3, b

unknowns: $r_1, r_2, r_3, T^*, (K^*)$ $s, \alpha_u, \alpha_v, u_0, v_0, s$

① * trick to solve this is to ~~remember~~ remember that rows of R^* are orthogonal.

To extract parameters using orthogonality of $r_1 r_2 r_3$

orthogonal property

$r_1 \cdot r_2 = 0 \quad r_2 \cdot r_3 = 0 \quad r_1 \cdot r_3 = 0$

$r_1 \times r_2 = r_3 \quad r_2 \times r_3 = r_1 \quad r_3 \times r_1 = r_2$

Find unknown scale:

$r_3^T = s a_3^T$

$\Rightarrow |r_3^T| = |s| \, |a_3^T| \Rightarrow |s| = \frac{1}{|a_3|}$ ← known

unknown

$|r_3^T| = 1$ since, R matrix is orthonormal. Vectors in R^* have unit length.

Page No.
Date

so now we know only magnitude but not scale.

(2) Find u_0:

$$s a_1^T \cdot s a_2^T = (\alpha_u r_1^T + s r_2^T + u_0 r_3^T) \cdot r_3^T$$
$$= 0 + 0 + u_0 \cdot 1 = u_0$$

$$\boxed{u_0 = |s|^2 \, a_1 \cdot a_3}$$ we know $|s|$ from (1)

similarly:

$$\boxed{v_0 = |s|^2 \, a_2 \cdot a_3}$$

(3) Finding α_v:

$$s a_2 \cdot s a_2 = (\alpha_v r_2^T + v_0 r_3^T) \cdot (\alpha_v r_2^T + v_0 r_3^T)$$

$$= \alpha_v^2 + v_0^2$$

$$\boxed{\alpha_v = \sqrt{s^2 a_2 \cdot a_2 - v_0^2}}$$

Page No.
Date

(4) Finding s:

$$\rho a_1 \times \rho a_3 = (\alpha_u r_1^T + s r_2^T + u_0 r_3^T) \times r_3^T$$

$$= -\alpha_u r_2^T + s r_1^T$$

$$\rho a_2 \times \rho a_3 = \alpha_v r_1^T$$

$$(\rho a_1 \times \rho a_2) \cdot (\rho a_2 \times \rho a_3) = (-\alpha_u r_2^T + s r_1^T) \cdot \alpha_v r_1^T$$

$$= s \alpha_v$$

$$\boxed{s = \frac{1}{\alpha_v} \rho^4 (a_1 \times a_3) \cdot (a_2 \times a_3)}$$

* Finding the unknown sign of ρ

$$\rho = \epsilon |\rho| \qquad \epsilon = \text{sign}(\rho) = ?$$

ϵ ? $+1/-1$

$$K^* T^* = \rho b = \epsilon |\rho| b$$

$$[K^* T^*]_z = \epsilon |\rho| b_z$$

$+ve$ (object in front of camera)

Page No.
Date

$\Rightarrow \epsilon = \text{sign}(b_z)$ we take sign of b_z such that $[K^* r_3^*]$ stay +ve.

* So now we found $u_0, v_0, s, \alpha_u, \alpha_v, \beta, \epsilon$ so now we get K^* intrinsic parameter.

Lecture 24 - W12 L2

Page No.
Date

We find R^*, T^* using equations we have.

R^* involves equation 1, 2, 3.

T^* involves equation 4.

So we start with eq (4).

* Finding T^* :

We know K^* from recovered intrinsic parameters

$$K^* \quad T^* = \underbrace{\epsilon\, [s]}\, b \leftarrow \text{known (part of } \hat{M})$$

K^*: known

T^*: unknown

$\epsilon\,[s]\, b$: known

so,

$$\boxed{T^* = (K^*)^{-1} \epsilon\, [s]\, b}$$

Page No.
Date

★ Summary:

(1) $|\beta| = \frac{1}{|a_3|}$

(2) $u_0 = |\beta|^2 a_1 \cdot a_3$

(3) $v_0 = |\beta|^2 a_2 \cdot a_3$

(4) $\alpha_v = \sqrt{|\beta|^2 a_2 \cdot a_2 - v_0^2}$

(5) $s = |\beta|^4 \alpha_v (a_1 \times a_3) \cdot (a_2 \times a_3)$

(6) $\alpha_u = \sqrt{|\beta|^2 a_1 \cdot a_1 - s^2 - u_0^2}$

(7) $K^* = \begin{bmatrix} \alpha_u & s & u_0 \\ & \alpha_v & v_0 \\ & & 1 \end{bmatrix}$

(8) $\epsilon = \text{sgn}(b_3)$

(9) $T^* = \epsilon |\beta| (K^*)^{-1} b$

(10) $r_3 = \epsilon |\beta| a_3$

(11) $r_1 = |\beta|^2 / \alpha_v a_2 \times a_3$

(12) $r_2 = r_3 \times r_1$

(13) $R^* = [r_1^T \; r_2^T \; r_3^T]^T$

Page No.
Date

* Degenerate Configurations

If we take only picture of 1 plane of ~~dat~~ calibrator, our $\vec{M}$ fails.

To recover M we need to solve: *

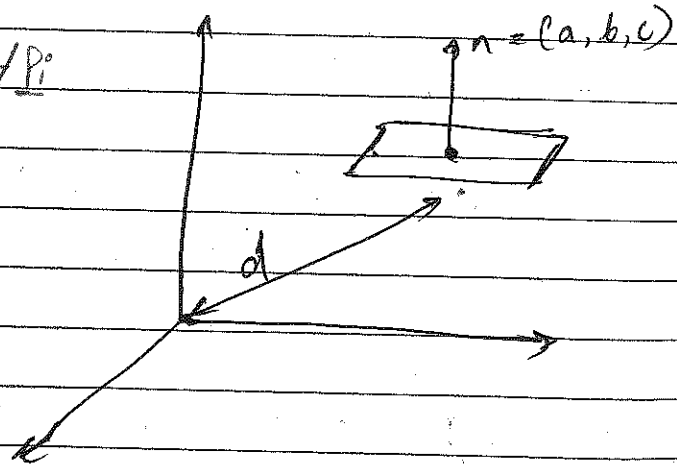
$$\begin{bmatrix} \underline{P_1}^T & 0 & -x_1 \underline{P_1}^T \\ 0 & \underline{P_1}^T & -y_1 \underline{P_1}^T \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

when $\underline{P_i}$ are all on the same plane π, we have

$$\Pi^T \underline{P_i} = 0 \qquad \forall \underline{P_i}$$

(a, b, c, d) — normal; d — distance

$(x, y, z, 1)$



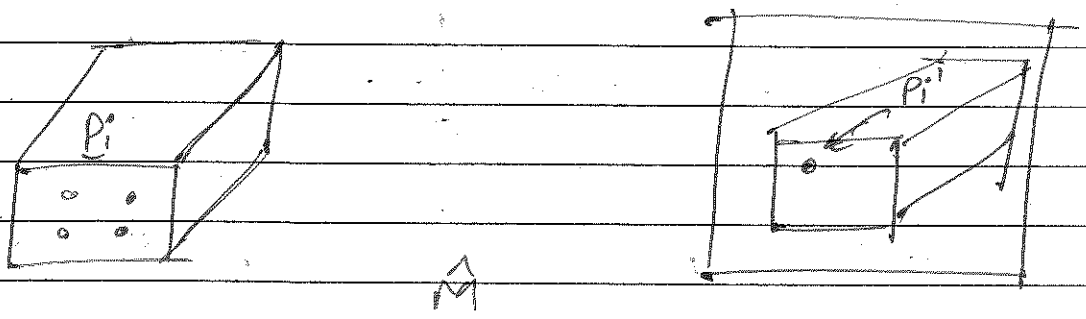
- When $\underline{P_i}$ are on same plane there are many solutions of the form:

$$\begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}_{(12\times1)} = \alpha \begin{bmatrix} \pi \\ 0 \\ 0 \end{bmatrix}_{(12\times1)} + \beta \begin{bmatrix} 0 \\ \pi \\ 0 \end{bmatrix}_{(12\times1)} + \gamma \begin{bmatrix} 0 \\ 0 \\ \pi \end{bmatrix}_{(12\times1)}$$

there are other degenerate configurations, but they are less likely to happen.

* Assessing the quality of fit:

We can measure quality by projecting world points onto image equivalent point with our solution $\hat{M}$ & then look at the distance between actual & found point.



given $\{P_i\}_{i=1}^{m} \longleftrightarrow \{\underline{P}_i\}_{i=1}^{m}$ and estimated $\hat{M} = \begin{bmatrix} -m_1^T- \\ -m_2^T- \\ -m_3^T- \end{bmatrix}$ access the quality of fit using:

$$E = \frac{1}{m} \sum_i \left(\left\| x_i - \frac{m_1^T P_i}{m_3^T P_i} \right\|^2 + \left\| y_i - \frac{m_2^T P_i}{m_3^T P_i} \right\|^2 \right)$$

2D (pixels)

i.e distance b/w known & predicted parameters.

if we take $\sqrt{E}$, we get error in pixels.

* Recovering R & T:

remember R^* & T^* are rotation of world wrt camera.

but we usually need R & T,

$$R^* = R^T \Rightarrow \boxed{R = (R^*)^T}$$

$$T^* = -R^T T \Rightarrow T = -R T^*$$

$$\Rightarrow \boxed{T = -(R^*)^T T^*}$$

Page No.
Date

☆ Planar calibration:-

* Approach:

→ $\hat{H}_1, \hat{H}_2, \hat{H}_3, \hat{H}_4 \ldots$

① estimate a 2D homography (projective map) between calibration plane and image (for several images)

② Estimate intrinsic parameters

③ Compute extrensic parameters for view of interest. ⟶ R, T

* 2D projective map:

$$P_i' = M P_i$$

P_i': 2DH (3×1)
M: 3×3
P_i: 2DH (3×1)

To get a 2D projective map we assume:

$$\{P_i\}_z = 0 \iff P_i = (X_i, Y_i, 0)$$

We make assumption that Z coordinates are 0.

$$\underset{2DH}{P_i'} = K^* \underbrace{[R^* \mid T^*]}_{(3\times4)} \underbrace{R_i}_{3DH}$$

$$= K^* \begin{bmatrix} r_1 & r_2 & r_3 & T^* \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{bmatrix}$$

$$= K^* (X_i r_1 + Y_i r_2 + T^*)$$

$$\underset{2DH}{P_i'} = K^* \underbrace{\begin{bmatrix} r_1 & r_2 & T^* \end{bmatrix}}_{3\times3} \underbrace{\begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}}_{2DH}$$

So,

Given : $\{P_i\}_{i=1}^{m} \longleftrightarrow \{\underline{P_i}\}_{i=1}^{m}$ $\qquad \underline{P_i} = \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$

Define : $\underline{P_i}^* = \begin{bmatrix} X \\ Y_i \\ 1 \end{bmatrix}$

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = K^* \begin{bmatrix} r_1 & r_2 & T^* \end{bmatrix} \underline{P_i}^*$$

$\{p_i\}_{i=1}^{m} \longleftrightarrow \{\underline{p}_i\}_{i=1}^{m} \longrightarrow \{p_i^*\}_{i=1}^{m}$

2DH 3DH 2DH

$p_i = \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$

$$\begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = H\underline{p}_i^* = \begin{bmatrix} -\ h_1^T\ - \\ -\ h_2^T\ - \\ -\ h_3^T\ - \end{bmatrix} \underline{p}_i^*$$

$$x_i = \frac{u_i}{w_i} = \frac{h_1^T \underline{p}_i^*}{h_3^T \underline{p}_i^*} \Rightarrow h_1^T \underline{p}_i^* - x_i h_3^T \underline{p}_i^* = 0$$

$$y_i = \frac{v_i}{w_i} = \frac{h_2^T \underline{p}_i^*}{h_3^T \underline{p}_i^*} \Rightarrow h_2^T \underline{p}_i^* - y_i h_3^T \underline{p}_i^* = 0$$

so, 2 equation for each point pair.

9 unknown : h_1, h_2, h_3 (8 independent)

$\Rightarrow$ Need a minimum of 4 point pairs.

$$\begin{bmatrix} \underline{p}_1^{*T} & 0^T & -x_1 \underline{p}_1^{*T} \\ 0 & \underline{p}_1^{*T} & -y_1 \underline{p}_1^{*T} \\ \vdots & \vdots & \vdots \\ \underline{p}_m^{*T} & 0^T & -x_m \underline{p}_m^{*T} \\ 0 & \underline{p}_m^{*T} & -y_m \underline{p}_m^{*T} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$2m \times 9$ $\quad 9 \times 1$ $\quad 2m \times 1$

$\Rightarrow AX = 0$ form

- Solve the homography equation ($Ax=0$) using SVD for several planar views (each with its own coordinate system)
$\Rightarrow \hat{H}_1, \hat{H}_2, \hat{H}_3 \ldots$

- each homography matrix is defined up to scale:

$$H \equiv K^* [r_1 \; r_2 \; T^*] = \alpha \hat{H}$$

(α: unknown scale)

like $M = s\hat{m}$

Let, $\hat{H} = [\hat{h}_1, \hat{h}_2, \hat{h}_3]$ (note columns of H instead of rows of M before)

$$H = K^* [r_1 \; r_2 \; T^*] = \alpha [\hat{h}_1 \; \hat{h}_2 \; \hat{h}_3]$$

$$\Rightarrow [r_1 \; r_2 \; T^*] = \alpha K^{*-1} [\hat{h}_1 \; \hat{h}_2 \; \hat{h}_3]$$

$$\Rightarrow \begin{aligned} r_1 &= \alpha K^{*-1} \hat{h}_1 \\ r_2 &= \alpha K^{*-1} \hat{h}_2 \\ T^* &= \alpha K^{*-1} \hat{h}_3 \end{aligned}$$

$$r_1 \cdot r_2 = 0 \Rightarrow (\alpha K^{*-1} \hat{h}_1)^T (\alpha K^{*-1} \hat{h}_2) = 0$$

$$\Rightarrow \hat{h}_1^T \underbrace{K^{*-T} K^{*-1}}_{S} \hat{h}_2 = 0$$

$$r_1 \cdot r_1 = r_2 \cdot r_2 = 1$$

$$\Rightarrow \hat{h}_1^T \underbrace{K^{*-T} K^{*-1}}_{S} \hat{h}_1 = \hat{h}_2^T \underbrace{K^{*-T} K^{*-1}}_{S} \hat{h}_2$$

$$\begin{cases} \hat{h}_1^T S \hat{h}_2 = 0 \\ \hat{h}_1^T S \hat{h}_1 = \hat{h}_2^T S \hat{h}_2 \end{cases} \qquad S \equiv K^{*-T} K^{*-1}$$

2 equations for each estimated homography $\hat{H}$

$$\begin{cases} \hat{h}_1^T S \hat{h}_2 = 0 \\ \hat{h}_1^T S \hat{h}_1 = \hat{h}_2^T S \hat{h}_2 \end{cases}$$

knowns : $\hat{h}_1$, $\hat{h}_2$

unknown : $S \rightarrow K^* \rightarrow \alpha_u, \alpha_v, u_0, v_0, s$ (5 unknowns)

$\rightarrow$ need atleast 3 planar views to estimate K^* ($2m = 6$)

* Algorithm:

1. write system of equations
2. solve for elements of S
3. extract intrinsic parameters from S ($\alpha_u, \alpha_v, u_0, v_0, s$)

[Refer notes for summary.]

$$H = [\hat{h}_1 \; \hat{h}_2 \; \hat{h}_3] = \begin{bmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{bmatrix}$$

$$v_{ij} = [h_{i1}h_{j1}, \; h_{i1}h_{j2} + h_{i2}h_{j1}, \; h_{i2}h_{j2}, \; h_{i3}h_{j1} + h_{i1}h_{j3}, \; h_{i3}h_{j2} + h_{i2}h_{j3}, \; h_{i3}h_{j3}]^T$$

Page No.
Date

$$\begin{bmatrix} V_{12}^T \\ V_{11}^T - V_{22}^T \\ \vdots \\ \{ \end{bmatrix} \begin{bmatrix} S_{11} \\ S_{12} \\ S_{22} \\ S_{13} \\ S_{23} \\ S_{33} \end{bmatrix} \begin{matrix} 1 \\ , \\ = \\ 3 \\ 4 \\ 5 \end{matrix} \begin{bmatrix} 0 \\ 0 \\ \} \\ \{ \\ \{ \\ \{ \end{bmatrix}$$

← solve using SVD

$c_1 = (S_{12} S_{13} - S_{11} S_{13})$

$c_2 = (S_{11} S_{22} - S_{12}^2)$

$v_0 = c_1 / c_2$

$\lambda = S_{33} - (S_{13}^2 + v_0 c_1) / S_{11}$

$\alpha_u = \sqrt{\lambda / S_{11}}$

$\alpha_v = \sqrt{\lambda S_{11} / c_2}$

$s = -S_{12} \alpha_u^2 \alpha_v / \lambda$

$u_0 = s v_0 / \alpha_u - S_{13} \alpha_u^2 / \lambda$

[refer notes]

★ Direct Methods

Given $\{P_i\}_{i=1}^{m} \longleftrightarrow \{\underline{p_i}\}_{i=1}^{m}$

the calibration parameters (K^*, R^*, T^*) can be used to project P_i :

$$q_i = K^* [R^* | T^*] \underline{P_i}$$

we start with guess and project q_i then we find ogleadean distance b/w prediction & action to get loss. Then we do gradient discent to get parameters.

$$\text{so, } L = \Sigma (q_i - p_i)^2$$

- objective function

$$E(K^*, R^*, T^*) = \sum_{i=1}^{m} \left\| \left(\frac{(q_i)_x}{(q_i)_z}, \frac{(q_i)_y}{(q_i)_z} \right) - \left((P_i)_x, (P_i)_y \right) \right\|^2$$

- minimize

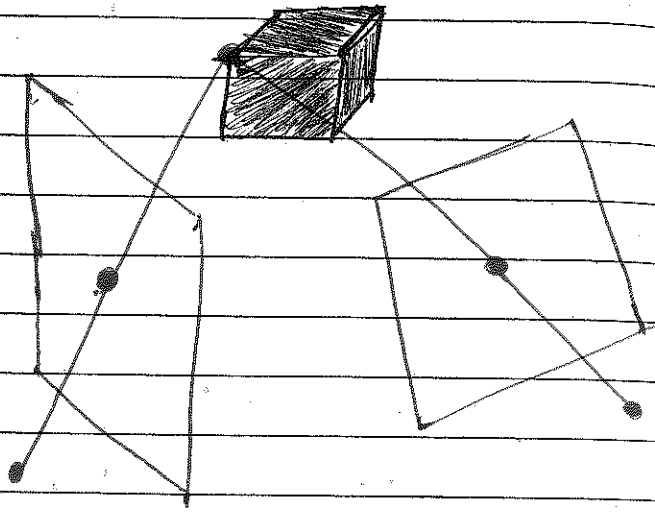
$$K^*, R^*, T^* = \underset{K,R,T}{\operatorname{argmin}} \; E(K,R,T)$$

do gradient discent

non-linear equation $\Longrightarrow$ Iterative solution (eg Tsai)

★ Multiple View Geometry

* 2 view geometry (stereo):



- use projection of the same 3D point in multiple views to recover structure (inverse problem)

- problems:
  - correspondence
  - Reconstruction

Correspondence:
which point correspond to which point in 2 images.

- approaches:-
  - sparse: feature-based (match points - can handle large
  - dense: correlation, SSD (produce more points)

- Cases when correspondence is not possible:
  - object points not visible in both views (occlusion)
  - ambiguous points (eg multiple condidates)
  - uniform regions (inside points cannot be distinguishe

Lecture 25 - W 13 L 1

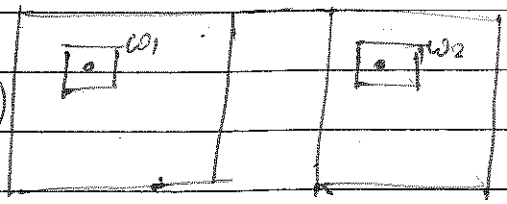
Page No.
Date

* Sparse correspondence

- find feature points (eg. corners)
- find local characterization (eg. SIFT, HOG)
- find corresponding points having similar features
- Greedy vs. optimal assignment
- Constraints for reducing # of candidates (eg. search around current location)

* Dense correspondence

- Instead of feature points capture all patches
- Instead of distance between feature vectors measure correlation / SSD
- Apply regularization to reduce errors and find correspondence in difficult areas (eg. uniform)



disparities)

Good correlation: $\Psi(w_1, w_2) = \sum_i w_1(x_i, y_i) w_2(x_i, y_i)$

Good SSD: $\Psi(w_1, w_2) = \sum_i (w_1(x_i, y_i) - w_2(x_i, y_i))^2$

- Removing dependency on absolute value:

[Fig] → values in windows could have similar values but some other window might have better corelation due to bigger value.

solution: normalize window values (subtract mean and divide by standard diviation)

Page No.
Date

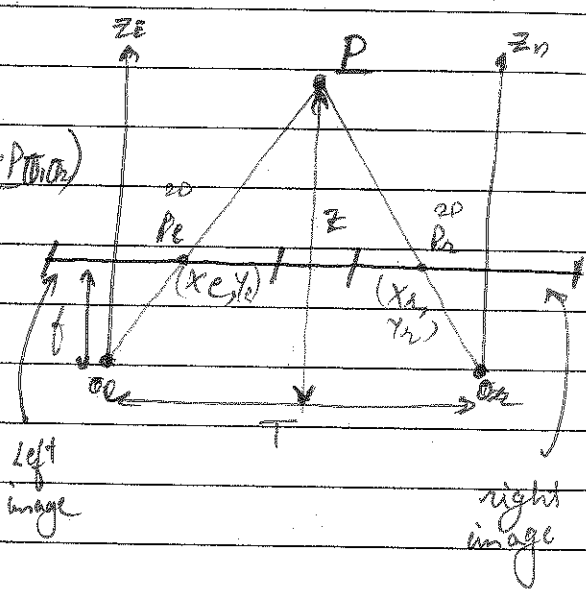
Zero mean normalize cross-correlation (ZNCC)

$$\psi(w_1, w_2) = \sum_i \frac{(w_1(x_i, y_i) - \mu w_1).(w_2(x_i, y_i) - \mu w_2)}{\sigma_1 \cdot \sigma_2}$$

Zeromean normalized SSD (ZNSSD)

$$\psi(w_1, w_2) = \sum_i \left(\frac{(w_1(x_i y_i) - \mu w_i)}{\sigma w_1} - \frac{(w_2(x_2 y_2) - \mu w_2)}{\sigma w_2} \right)^2$$

★ Axis-aligned stereo :



- Using similar triangles : ($PP_eP_r \sim PTO_r$)

$$\frac{I}{z} = \frac{T - (-x_e + x_r)}{z - f}$$

$$= \frac{T - (x_r - x_e)}{z - f} \quad (d = x_r - x_e)$$

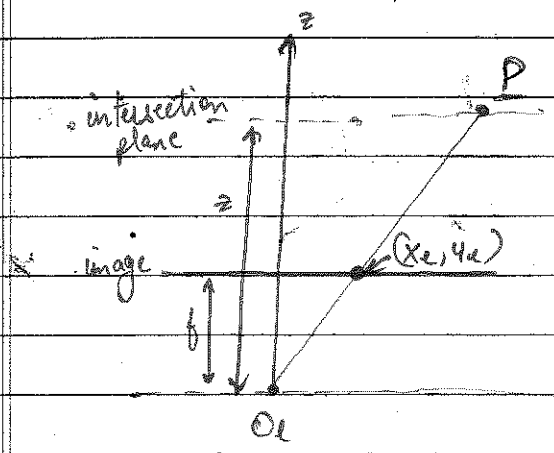
$$= \frac{T - d}{z - f} \leftarrow \text{disparity}$$

i.e. the bigger the disparity, smaller the depth.

$$\frac{T}{z} = \frac{T - d}{z - f} \Rightarrow \boxed{z = f \frac{T}{d}}$$

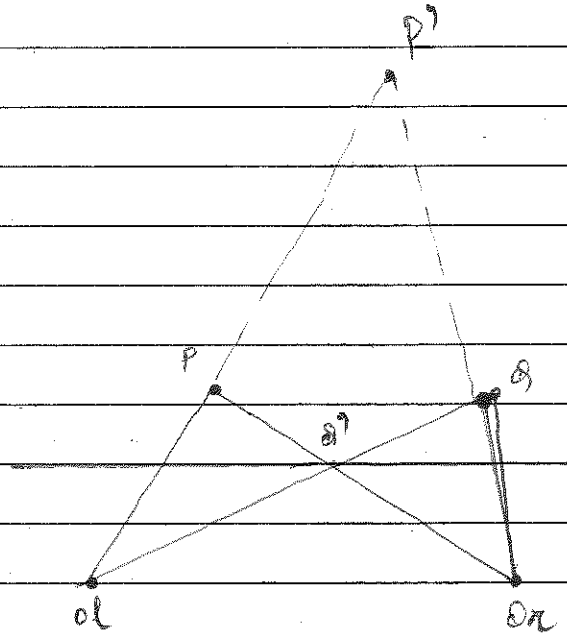
f, T is always same for all pairs.
d will vary.

- To reconstruct 3D point, send a ray from O_l through (x_e, y_e) and find intersection with a parallel plane at a distance of z (where z is computed as before)



Ambiguity problem:

correct: PQ
incorrect: P'Q'

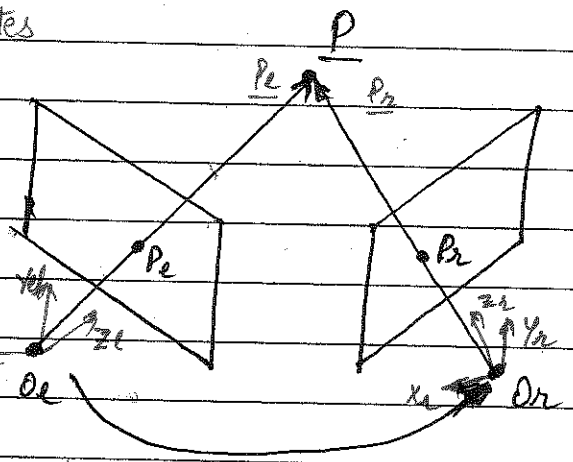


Page No.
Date

★ General stereo :

$\underline{P}$ = point in world coordinates
$\underline{P_e}$ = point in left coordinates
$\underline{P_r}$ = point in right coordinates

P_e = image point in left coordinates
P_r = image point in right coordinates



first rotate, then translate → R, T ← stereo parameters

– stereo calibration –

• getting relative rotation & translation (R, T)

calibrate ⇒ { R_e, T_e external : rotation / translation of left camera w.r.t world
R_r, T_r external : rotation / translation of right camera w.r.t world

$$M_{left \leftarrow right} = \underbrace{M_{left \leftarrow world}}_{R_e^T \; T_e^{-1}} \; \underbrace{M_{world \leftarrow right}}_{(R_r^T \; T_r^{-1})^{-1} = T_r R_r}$$

(right ← world)

$$= R_e^T \; T_e^{-1} \; T_r R_r = TR$$

so, $M_{left \leftarrow right} = R_e^T \; T_e^{-1} \; T_r \; R_r$

$$= \begin{bmatrix} R_e^T & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & -T_e \\ 0 & 1 \end{bmatrix} \begin{bmatrix} I & T_r \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_r & 0 \\ 0 & 1 \end{bmatrix}$$

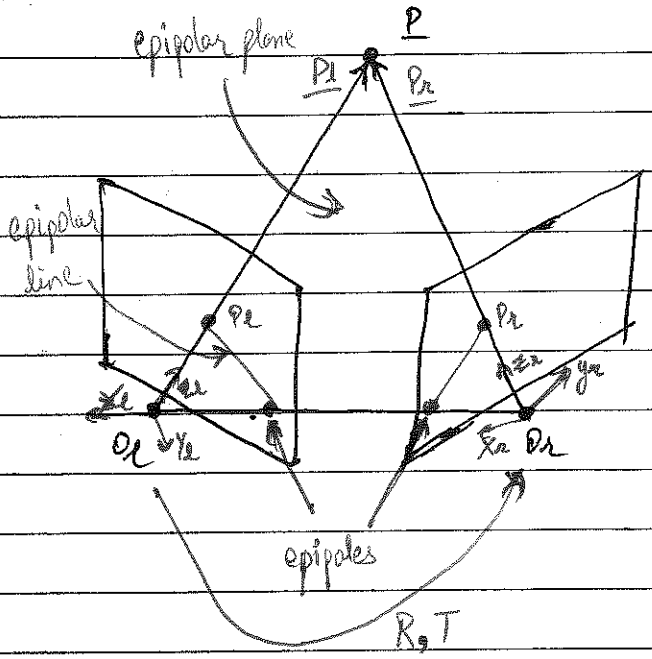
$$= \begin{bmatrix} R_e^T & -R_e^T T_e \\ 0 & 1 \end{bmatrix} \begin{bmatrix} R_r & T_r \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_e^T R_r & R_e^T (T_r - T_e) \\ 0 & \end{bmatrix}$$

(rotation, translation)

$\Rightarrow$ $\boxed{R = R_e^T R_r}$ $\boxed{T = R_e^T (T_r - T_e)}$

* Epipolar geometry

- the points O_e, O_r, P define epipolar plane.
- the intersection of the epipolar plane with the images define epipolar lines
- The intersection of the baseline with the images defines the epipoles.



- properties:
  - every epipolar line must pass through the epipole
  - epipoles may be insider or outside the image
  - epipoles may be at infinity (when the image plane become parallel).

* Epipolar constraint

- Given a point P_e in the left image P_e, O_e, O_r define an epipolar plane.
- The intersection of this epipolar plane with the right image defines an epipolar line on it.
- the point P_r corresponding to P_e on the left image must be on this epipolar line.

Page No.
Date

* Essential matrix :

- P_r and P_e are expressed in different coordinates systems. we would like to move to the left coordinate system.
- Given that the right view is rotated and translated by R, T w.r.t left, it is easy to move from left to right

$$\underline{P_r} = R^T(\underline{P_e} - T)$$

$$\Rightarrow \underline{P_e} = R\underline{P_r} + T \quad \leftarrow \text{convert } \underline{P_r} \text{ in left coordinates}$$

Lecture 26 - W13 L2

Page No. | Date

- moving from camera to image coordinates:

$$P_e = \frac{f_e}{z_e} P_e$$

$$P_r = \frac{f_r}{z_r} P_r$$

- Cross product as matrix multiplication:

matrix vector

$$A \times B = [A]_\times B \equiv \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} B$$

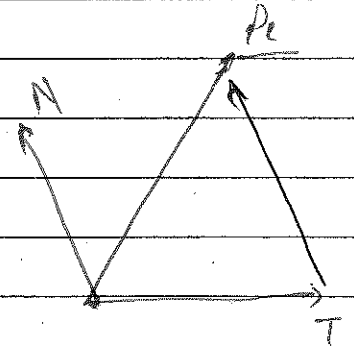
$$A = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

rank 2 skew-symmetric matrix ($A^T = -A$)

* Essential matrix:

- To specify an epipolar plane we need to find a vector normal to it.

$$N = T \times P_e$$



- The vector P_r is on the epipolar plane:

$$P_r \cdot N = 0$$

- Moving P_r to left coordinates

$$(R P_r + T) \cdot N = 0$$

$(R\underline{P_r} + T) \cdot N = 0 \Rightarrow (R\underline{P_r} + T) \cdot (T \times \underline{P_l}) = 0$

$= 0$

$\Rightarrow R\underline{P_r} \cdot (T \times \underline{P_l}) = 0$

$\Rightarrow (R\underline{P_r})^T [T]_\times \underline{P_l} = 0$

getting rid of dot & cross

$A \cdot B = A^T B$

$\Rightarrow \underline{P_r}^T R^T [T]_\times \underline{P_l} = 0$

$= E$ (3×3)

$\Rightarrow \boxed{\underline{P_r}^T E \underline{P_l} = 0}$ ← Epipolar constraint equation

Essential matrix

* summarize:

- The essential matrix

$$E \equiv R^T [T]_\times$$

rank 2 (because of $[T]_\times$)

- Epipolar constraint:

$$\underline{P_r}^T E \underline{P_l} = 0$$

- Epipolar constraint in image coordinates:

$$p_r^T E p_l = 0$$

Page No.
Date

* Epipolar Constraint

• right epipolar line : (given left image - P_e)

$$\boxed{l = E P_e}$$

the corresponding P_r should be on this line.

• Explanation :

$P_r^T E P_e = 0 \Rightarrow P_r^T l = 0 \Rightarrow P_r \cdot l = 0 \Rightarrow P_r$ is on the line l

line equation $p^T l = 0$; $l = (a, b, c)$ ~~distance from origin~~

• left epipolar line : (given right image - P_r)

$$l = E^T P_r$$

the corresponding point P_e should be on this line.

Explanation :

$$P_e^T E P_e = 0 \Rightarrow (P_r^T E P_e)^T = 0 \Rightarrow P_e^T E^T P_r = 0$$

$$\Rightarrow P_e^T l = 0$$

$$\Rightarrow l = E^T P_r$$

But these only take extrensic R & T into consideration, but we also want intrensic to be included and so we use fundamental matrix

Page No.
Date

* Fundamental matrix (related to essential matrix)

∴ so far the epipolar constraint was expressed in camera coordinates - we want to move to image coordinates

$$\overline{P_e} = K_e^* P_e \qquad K_e^* = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$\overline{P_e}$: image coordinates; P_e: camera coordinates

$u_0, v_0 \rightarrow$ optical center

$\alpha_u, \alpha_v \Rightarrow$ focal len in pixels

$$\Rightarrow P_e = K_e^{*-1} \overline{P_e}$$

- likewise:

$$P_r = K_r^{*-1} \overline{P_r}$$

so epipolar constraint:

$$P_r^T E P_e = 0$$

$$\Longleftrightarrow (K_r^{*-1} \overline{P_r})^T E (K_e^{*-1} \overline{P_e}) = 0$$

$$\Longrightarrow \overline{P_r}^T \underbrace{(K_r^{*-T} E K_e^{*-1})}_{\equiv F \text{ (fundamental matrix)}} \overline{P_e} = 0$$

- The fundamental matrix takes into account both internal and external parameters, whereas the essential matrix takes into account only external parameters.

F is rank 2 because of E

Page No.
Date

* Summary :

- In camera coordinates

$$P_r^T E P_e = 0$$

- In image coordinate

$$\bar{P}_r^T F \bar{P}_e = 0$$

- $\underset{3\times3}{F} = K_r^{*-T} E K_e^{*-1}$

- $\underset{3\times3}{F} = R^T [T]_\times$

both E & F are Rank 2

- Given P_e, right epipolar line is : $l = E\bar{P}_e$
- Given P_r, left epipolar line is : $l = E^T P_r$

* Weak Calibration:

- Full calibration : $K, T, K_e^*, K_r^* \Rightarrow E, F$
- weak calibration : E, F

Page No.
Date

- Problem statement:

given $\{P_i'\}_{i=1}^{m} \longleftrightarrow \{P_i\}_{i=1}^{m}$ $\quad m \geq 8$

(left points) (right points)

find F (3x3 matrix)

(8 unknown elements of F)

*Eight point algorithm:

- One equation for each point pair:

(1) $P_i^T F P_i' = 0$

8 unknowns $\Rightarrow$ need at least 8 point pairs

(2) $[x_i, y_i, 1] \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = 0$

(3) $x_i x_i' f_{11} + x_i y_i' f_{12} \ldots\ldots + f_{33} = 0$

(4)
$$\underset{m\times 9}{\begin{bmatrix} x_1x_1', x_1y_1', \ldots\ldots, 1 \\ \\ \\ x_mx_m', x_my_m', \ldots\ldots, 1 \end{bmatrix}} \underset{9\times 1}{\begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ \vdots \\ f_{33} \end{bmatrix}} = \underset{m\times 1}{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}$$

Page No.
Date

- this is in an $Ax=0 \Rightarrow$ solution is right zero eigen~~vector~~ vector of A

(SVD $\Rightarrow A = UDV^T$ then take last column of V)

- To enforce rank 2 for the estimated F:

$$SVD \rightarrow \underset{3\times3}{F} = UDV^T$$

$$\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

D' = D with smallest singular value set to 0 ($D'[3,3]=0$)

$F' \equiv UD'V^T \rightarrow$ rank 2 matrix

- Evaluate how good F is:

$$\{Pi\} \longleftrightarrow \{Pi'\} \quad \Sigma (Pi^T F Pi') \approx 0$$

good fit

sometimes it would fit, other times it wouldn't. Solution: Normalization.

* Normalization:

- for a stable solution in 8Palgorithm, normalize the point sets. m m m

$$\{Pi\}_1^m \longleftrightarrow \{Pi'\}_1^m \Rightarrow \{qi\}_1^m \longleftrightarrow \{qi'\}_1^m$$

Page No.
Date

$$q_i = \frac{p_i - M_p}{\sigma_p} \qquad q_i' = \frac{p_i' - M_{p'}}{\sigma_{p'}}$$

M = mean
σ = std dev

- But the solution will be good for q points, so we need to convert that to p points.
- In matrix form:

$$q_i = \underbrace{\begin{bmatrix} 1/\sigma_p & & \\ & 1/\sigma_p & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -M_{px} \\ 0 & 1 & -M_{py} \\ 0 & 0 & 1 \end{bmatrix}}_{\equiv T_p} p_i$$

$$q_i = T_p p_i$$

$$q_i' = T_{p'} p_i'$$

- using $\{q_i\}_1^m \leftrightarrow \{q_i'\}_1^m$ and 8P algo find F':

$$q_i^T F' q_i' = 0$$

$$(T_p p_i)^T F' T_{p'} p_i' = 0$$

$$p_i^T \underbrace{T_p^T F' T_{p'}}_{=F} p_i' =$$

$$\boxed{F = T_p^T F' T_{p'}}$$

Page No.
Date

* Summarize:

given $\{p_i\}_1^m \longleftrightarrow \{p_i'\}_1^m$

① normalize points: $q_i = M_p p_i$
$q_i' = M_{p'} p_i'$

② use 8 point algo to get F'

③ Convert F' to F: $F = M_{p'}^T F' M_p$

↑ weak calibration

Lecture 27 - W14L1

(contd)

$P_R^T F P_e = 0$

- given P_e we have epipolar line FP_e in the right image
- every point on this line satisfies $P^T F P_e = 0$
- specifically: $l_r^T F P_e = 0$
- since this is for all P_e it must be that:

$l_r^T F = 0$ ← $Ax = 0$ form

- $l_r^T F = 0 \Rightarrow \underset{3\times3}{F^T} \underset{3\times1}{l_r}$ → l_r is zero eigenvector of F^T

⇒ l_r is the left null space of F (SVD: $F = UDV^T$ → e_r is the last column of U)

- for the left epipole: $P_l^T F^T P_R = 0$

⇒ $e_l^T F^T P_R = 0 \ \forall P_R \Rightarrow e_l^T F^T = 0 \Rightarrow F e_l = 0$

$P_R^T F e_l = 0 \ \forall P_R \Rightarrow F e_l = 0$ → e_l is zero eigenvector of F

⇒ e_l is right null space of F
(SVD: $F = UDV^T$ → e_l is the last col^n of V)

Page No.
Date

tion

$\{p_i\}_{i=1}^{m} \longleftrightarrow \{p_i'\}_{i=1}^{m}$ find $\{P_i\}_{i=1}^{n}$

2D 3D

Cases:

1) know intrinsic + extrinsic → (Absolute reconstruction)

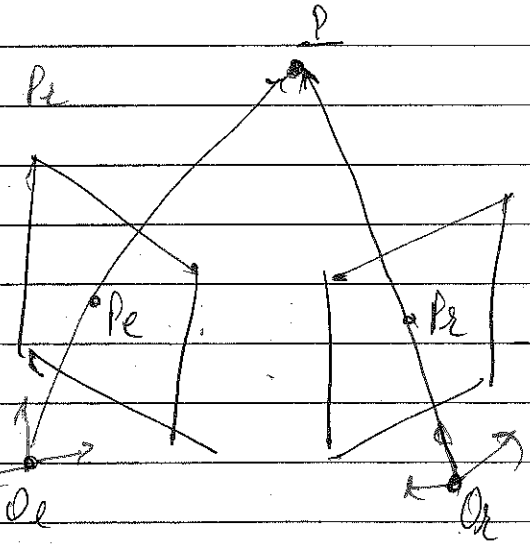
2) know intrinsic → (euclidean reconstruction (up to scale))

3) None are known → ~~Eq~~ Reconstruction up to unknown 3D projective map. (quite useless)

I) Absolute Reconstruction

* Triangulation Algorithm: (algorithm to find reconstruction)

- send ray from o_l through p_l & a ray from o_r through p_r
- Reconstruct P at rays intersection.



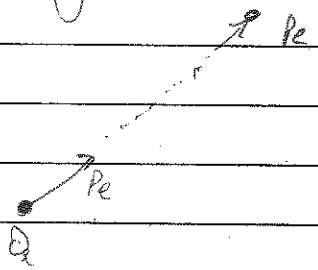
*** step 1:** move to camera coord:

$\bar{p}_l = K_l^* P_l$ (image coord) ⟹ $p_l = K_l^{*-1} \bar{p}_l$ (camera coord.)

$\bar{p}_r = K_r^* P_r$ ⟹ $p_r = K_r^{*-1} \bar{p}_r$

Page No.
Date

* step 2 : find intersections of rays:



left ray: a Pe

right ray: b Pr*

intersection: a Pe = b Pr*

- the ray b Pr* is right ray in left coordinates

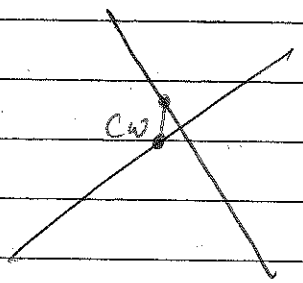
Pr* = M left ← right Pr = R Pe + T

* - To find ~~instersection~~ intersection solve:

a Pe = b Pr* → a Pe = b R Pe + T → solve for

- Problem:

due to inaccuracies rays do not intersect (no solution)



so we introduce cw

$$\begin{cases} a Pe + cw = b Pr^* & \leftarrow R Pr + T \quad (1) \\ w = Pe \times Pr^* = Pe \times R Pr & (2) \end{cases}$$

→ Putting (2) in (1):

$$\underbrace{a}\, Pe + \underbrace{C}(Pe \times R Pr) - \underbrace{b}\, R Pr = T$$

unknowns

$$\begin{bmatrix} 1 & 1 & 1 \\ \bar{P}_e & \bar{P}_e \times R\bar{P}_r & -R\bar{P}_r \\ | & | & | \end{bmatrix} \begin{bmatrix} a \\ c \\ b \end{bmatrix} = T \rightarrow \text{solve for } a,b,c$$

$$Ax = B \rightarrow x = A^{-1}B$$

*Summary:

1) Given $\bar{P}_e$, $\bar{P}_r$, R, T, k_e^*, k_i^* solve for a b c

2) Compute $\underline{P}$:

a,b

$$\underline{P} = \frac{1}{2}\left(a_0 \bar{P}_e + \underbrace{b_0 P R_r + T}_{b_0 P_r^*}\right)$$

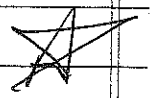
— or —

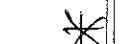
$$\underline{P} = a_0 \underline{P}_d + \frac{1}{2} c_0 \underline{w}$$

3) Convert $\underline{P}$ to world coordinates (from left coords)

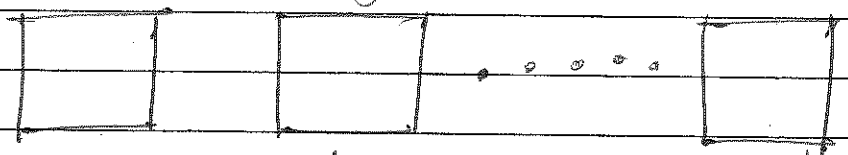
$$\underline{P}^{(w)} = R_e \underline{P} + T_e$$

Page No.
Date





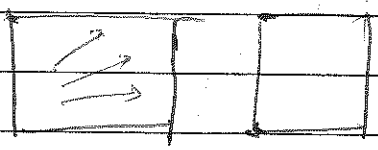
● → sequence of images:



if frame rate is 24 fps.

$$\Delta t = \frac{1}{24} [s]$$

→ Difference from stereo



large motion

→ types of motion:

① Single motion (static camera and moving object-or-static object and moving camera).

② Multiple motions (multiple objects & camera moving).

→ Problem:

1. Correspondence (easier in video)
2. Reconstruction (easier in stereo)
3. Motion segmentation

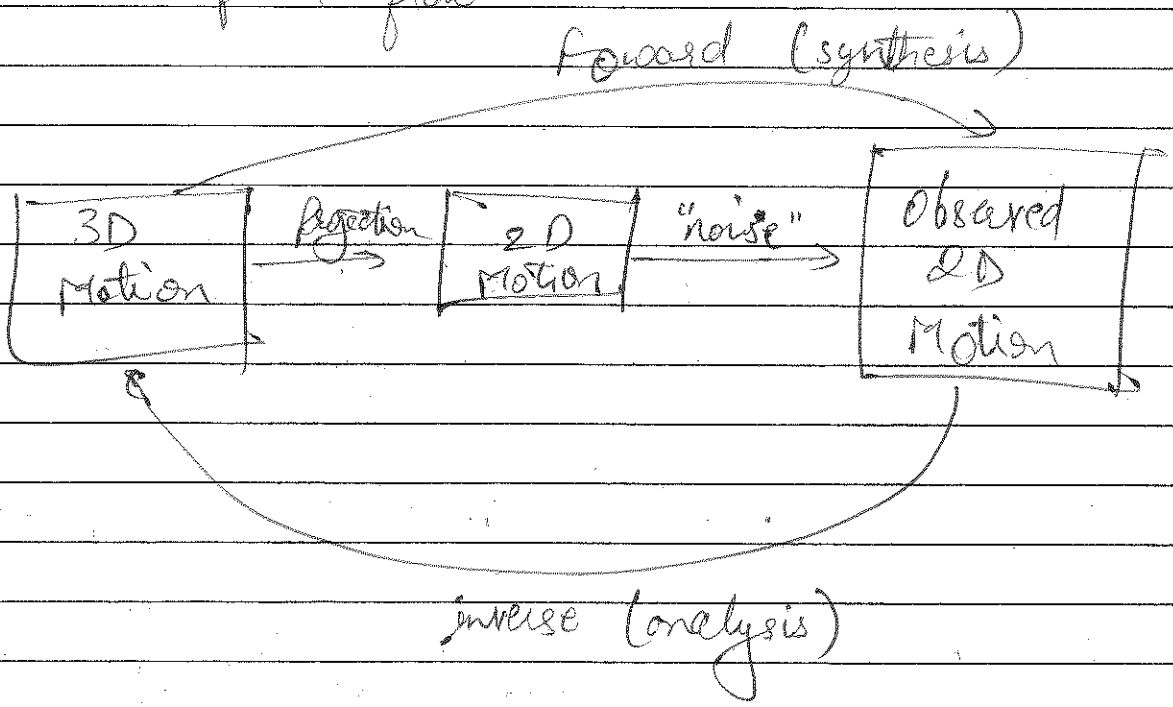
→ Motion of rigid objects

3D motion vectors (world)

2D projected motion (camera)

2D observed motion (image)

Optical flow



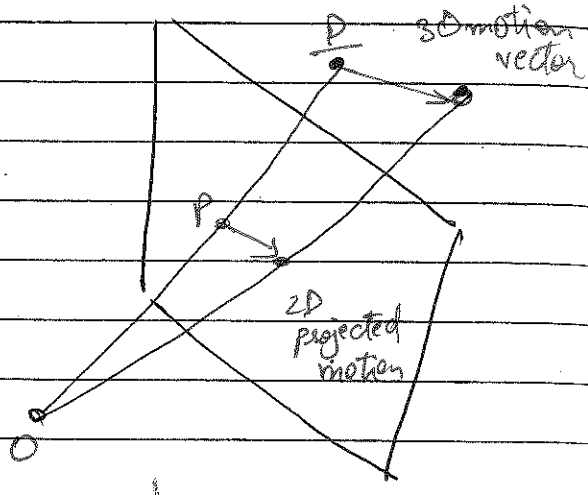
Page No.
Date

* Basic equations of projected motion

$$P = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad p = \begin{bmatrix} x \\ y \\ f \end{bmatrix}$$

$$p = f \frac{P}{Z}$$

Z coordinate of P



p, P, z all depend on t

* Motion in image:

$$\frac{dp}{dt} = \frac{d}{df}\left(f \frac{P}{Z}\right) = f\left(\frac{1}{Z}\frac{dP}{dt} - \frac{1}{Z^2}\frac{dz}{dt}P\right)$$

$$v = f\left(\frac{1}{Z}V - \frac{1}{Z^2}V_z P\right)$$

$$\boxed{v = \frac{f}{Z^2}\left(ZV - V_z P\right)}$$

Page No.
Date

* In component form:

$$\begin{cases} V_x = \dfrac{f}{Z^2}\left(Z V_x - V_z X\right) \\ V_y = \dfrac{f}{Z^2}\left(Z V_y - V_z Y\right) \\ V_z = \dfrac{f}{Z^2}\left(Z V_z - V_z Z\right) = 0 \end{cases}$$

Projected motion vectors do not have z component.

* Moving to image coordinates from world coords.

$$p = \frac{f P}{Z} \Rightarrow P = \frac{p Z}{f}$$

$$v = \frac{f}{Z^2}\left(Z V - V_z P\right)$$

$$= \frac{f}{Z^2}\left(Z V - V_z \frac{p}{f} Z\right)$$

$$v = \frac{f}{Z} V - \frac{p}{Z} V_z$$

Page No.
Date

* Decomposing projected motion

Plugging V_x, V_y, V_z, & projected motion equation.

$$\begin{cases} V_x = V_x^{(T)} + V_x^{(\omega)} \\ V_y = V_y^{(T)} + V_y^{(\omega)} \\ V_z = 0 \end{cases}$$

Projected motion

$$\begin{cases} V_x^{(T)} = \dfrac{z_z\, x - z_x\, f}{z} \\ V_y^{(T)} = \dfrac{z_z\, y - z_y f}{z} \end{cases}$$

(T) translational component

$$\begin{cases} V_x^{(\omega)} = -\omega_y f + \omega_z y + \dfrac{\omega_x xy}{f} - \dfrac{\omega_y x^2}{f} \\ V_y^{(\omega)} = \omega_x f - \omega_z x - \dfrac{\omega_y xy}{f} + \dfrac{\omega_x y^2}{f} \end{cases}^{(\omega)}$$

rotational motion comp.

Page No.
Date

* Special cases

(1) Pure translational motion

$$\Omega = 0 \Rightarrow \begin{cases} v_x = v_x^{(T)} \\ v_y = v_y^{(T)} \end{cases}$$

2 Subcases:

(a) $T_z \neq 0$ (translation in z)

(b) $T_z = 0$ (no translation z)

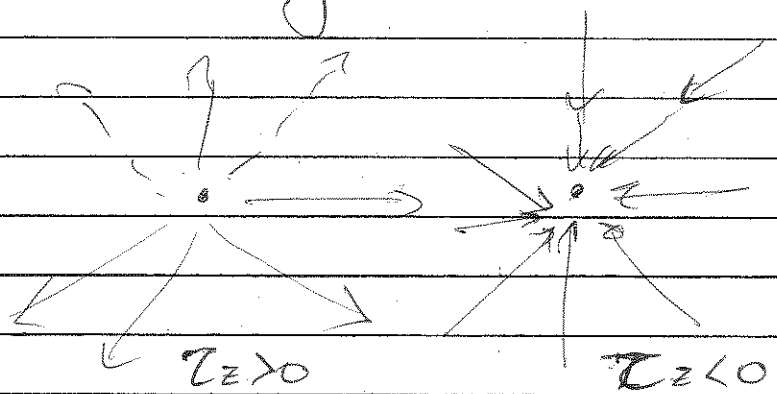
Subcase 1°: $T_z \neq 0$ (and pure translation)

$$\begin{cases} v_x = \dfrac{T_z x - T_x f}{z} = \dfrac{T_z}{z}\left(x - \dfrac{T_x}{T_z} f\right) \\ v_y = \dfrac{T_z y - T_y f}{z} = \dfrac{T_z}{z}\left(y - \dfrac{T_y}{T_z} f\right) \end{cases}$$

x_0, y_0

$$\begin{cases} v_x = \dfrac{T_z}{z}(x - x_0) \\ v_y = \dfrac{T_z}{z}(y - y_0) \end{cases}$$

→ the x_0, y_0 are instantaneous epipole



2D

$$(x_0, y_0) \equiv \left(\frac{T_x}{T_z} f, \frac{T_y}{T_z} f \right)$$

In homogeneous coords: 2DH

$$(T_x f, T_y f, T_z)$$

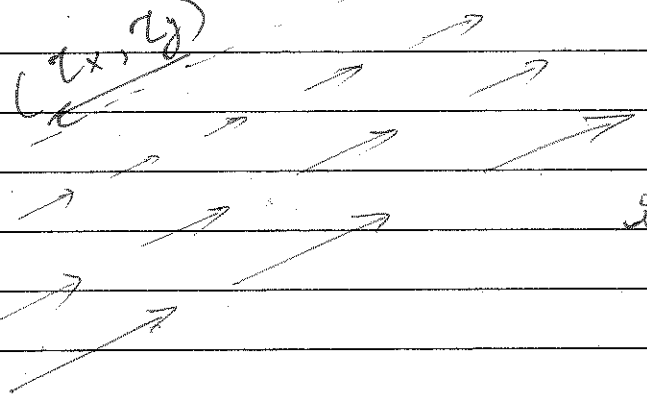
Lecture 28 - 01/5/21

Page No.
Date

Subcase 2: ($\tau_0 = 0$) (and pure translation)

$$\begin{cases} v_x = \dfrac{\tau_z x - \tau_x f}{z} = -\dfrac{\tau_x}{z} \cdot f \\ v_y = \dfrac{\tau_z y - \tau_y f}{z} = -\dfrac{\tau_y}{z} \cdot f \end{cases}$$

linear motion field



Instantaneous epipole in homogeneous coordinates is a point at infinity

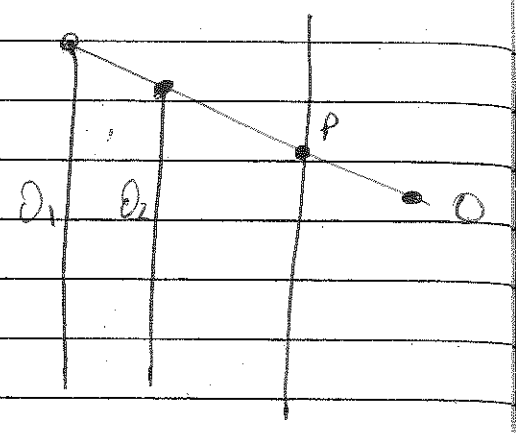
$(\tau_x f, \tau_y f, \tau_z)$

$\tau_z = 0 \Rightarrow$ direction

Page No.
Date

* Motion in instantaneous objects

Here $V^{(\omega)} = \bar{V}^{(\omega)}$



So,

$V_x = \bar{V}^{(T)}$

$\bar{V}_y = \bar{V}^{(T)}$

$V_x = V^{(T)}$

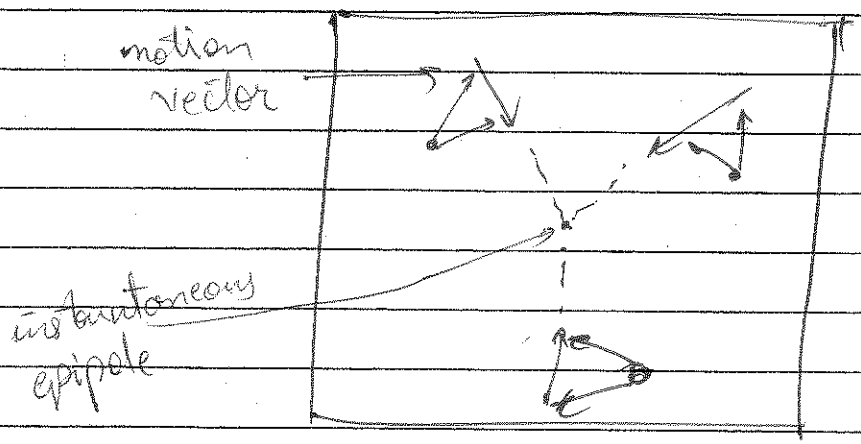
$V_y = V^{(T)}$

So, $V_x = (x - x_0)\frac{T_z}{z}$ $\quad \bar{V}_x = (\bar{x} - \bar{x}_0)\frac{T_z}{z}$

$V_y = (y - y_0)\frac{T_z}{z}$ $\quad \bar{V}_y = (\bar{y} - \bar{y}_0)\frac{T_z}{z}$

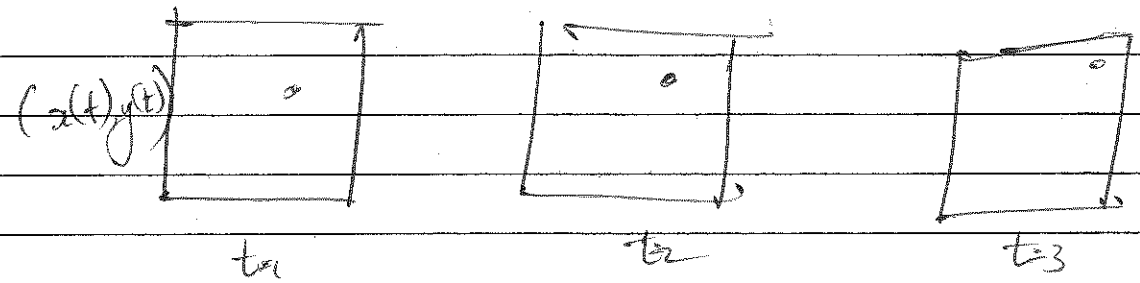
$$\Delta V_x = (x - x_0)\frac{T_z}{z} - (x - x_0)\frac{T_z}{\bar{z}} = (x - x_0)\, T_z \left(\frac{1}{z} - \frac{1}{\bar{z}}\right)$$

$$\Delta V_y = (y - y_0)\, T_z \left(\frac{1}{z} - \frac{1}{\bar{z}}\right)$$



* Optical flow Estimation

- optical flow = motion observed in images
- assume image brightness per object point is constant.



$$I(x(t), y(t), t) = C$$ — IBCE

IBCE - image bright constancy equation.

$$\frac{d}{dt}(IBCE) = 0$$

$$\frac{\partial I}{\partial x} \cdot \frac{\partial x}{\partial t} + \frac{\partial I}{\partial y} \cdot \frac{\partial y}{\partial t} + \frac{\partial I}{\partial t} = 0$$

$$\boxed{\nabla I \cdot v = -I_t}$$ ← optical flow constraint equation

∇I: spatial gradient; v: velocity vector; I_t: time derivative

$$\nabla I \cdot V = -I_t$$

$V = ?$

Ofc: $\frac{\nabla I}{\|\nabla I\|} \cdot V = -\frac{I_t}{\|\nabla I\|}$

- Problem?
  different motion vectors create same point.

- It is only possible to extract the component of the motion vector parallel to the image gradient

$$V_\perp = \frac{-I_t}{\|\nabla I\|}$$

Page No.
Date

* Second Order Of estimation

Assume constant spatial gradient (in addition to constant intensity):

$$\begin{cases} I_{BCE} = C_1 \\ \nabla (I_{BCE}) = C_2 \end{cases}$$

$$\frac{d}{dt}\left(\nabla I\,(x(t), y(t), t)\right) = 0$$

$$\frac{d}{dt}\begin{bmatrix} \frac{d}{dx} I_{BCE} \\ \frac{d}{dy} I_{BCE} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\frac{d}{dt}\begin{bmatrix} I_x \\ I_y \end{bmatrix} = \begin{bmatrix} I_{xx}\,x_t + I_{xy}\,y_t + I_{xt} \\ I_{yx}\,x_t + I_{yy}\,y_t + I_{yt} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} I_{xx} & I_{xy} \\ I_{yx} & I_{yy} \end{bmatrix}}_{A} \underbrace{v}_{V} = \underbrace{\begin{bmatrix} -I_{xt} \\ -I_{yt} \end{bmatrix}}_{B}$$

solve for v (noisy estimate)

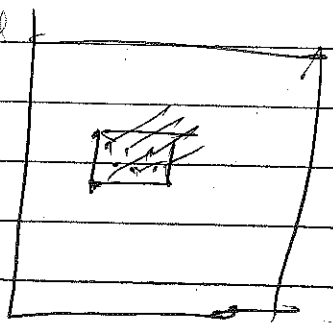
$$\boxed{V = A^{-1}B}$$

Page No.
Date

* Block based of estimate

assume motion is small patch is constant

$$E(v) = \sum_{(x,y) \in \text{patch}} \underbrace{\left(\nabla I(x,y)^T v + I_t(x,y) \right)^2}_{DFCE = 0}$$



$$v^* = \underset{v}{\operatorname{argmin}}\ E(v)$$

$$\nabla E(v) = 0$$

solving,

$$E(v) = \sum \left(I_x \cdot x_t + I_y \cdot y_t + I_t \right)^2$$

$$\frac{dE}{dx_t} = 0 = 2 \sum \left(I_x \cdot x_t + I_y \cdot y_t + I_t \right) I_x$$

$$\frac{dE}{dy_t} = 0 = 2 \sum \left(I_x x_t + I_y \cdot y_t + I_t \right) I_y$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

Known — unknown of vector — Known

Page No.
Date

* Summary for BBOF

$$\underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_t \\ y_t \end{bmatrix}}_{x} = \underbrace{\begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}}_{B}$$

$$x = A^{-1}B$$

- solve for optical flow vector in patch (x_t, y_t)
- solution exist when the structure tensor matrix is non singular (at corner).

* Weighted BBOF

- given preference to satisfying OFCE at the center of the patch.

$$E(x_t, y_t) = \sum w(x,y) \ (OFCE)^2$$

$$w(x,y) = \frac{1}{\|(x,y) - (x_c, y_c)\| + 1}$$

— or —

$$w(x,y) = \exp\left(-\|(x,y) - (x_c, y_c)\|^2 / \sigma^2\right)$$

Page No.
Date

$$\begin{bmatrix} \sum w I_x^2 & \sum w I_x I_y \\ \sum w I_x I_y & \sum w I_y^2 \end{bmatrix} V = \begin{bmatrix} -\sum w I_x I_t \\ -\sum w I_y I_t \end{bmatrix}$$

Known — unknown — Known

Known