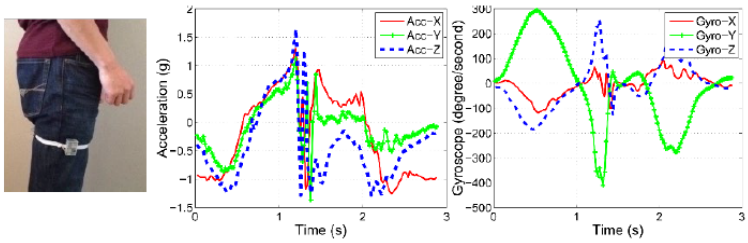
Spatial Reasoning from Sensor Data

- Sensor brief introduction

	BEACONS	GPS	WI-FI	NFC
Recommended for	In/near-store and micro-location use-cases	Macro-location and out of store use-cases	In-store use-cases	Close proximity, secure interaction
Some potential uses	In-aisle notifications and offers, in-store navigation, hands-free payment	Near-store notifications and offers, pre-arrival customer 'check-in'	In-aisle notifications and offers, in-store navigation, hands-free payment	Payments, product tagging
Ease of set up and maintenance	Medium	Medium-high	Medium	Medium
Range	Medium	Long	Medium-low	Close
Accuracy	Medium	Medium-low	Medium	High
Ease of use for consumer	Medium	Medium	Medium-high	Medium-high
Energy efficiency on consumer device	Medium-high	Medium-low	Medium-high	High

  - Proprioceptive sensors(internal)

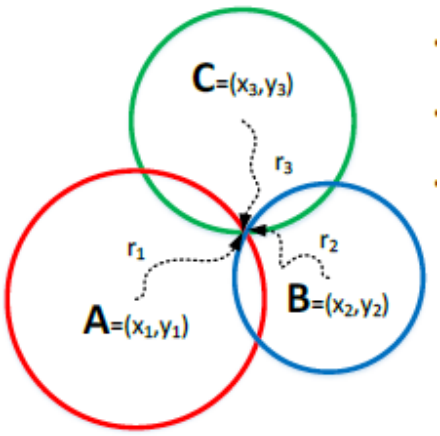


    - Measure values internally to the system(robot),
    - e.g.motor speed,wheel load,heading of the robot,battery status
  - Exteroceptive sensors(external)
    - Information from the robots environment
    - Distances to objects,intensity of the ambient light,unique features.
  - Passive sensors
    - Measure energy coming from the environment
  - Active sensors
    - Emit their proper energy and measure the reaction,better performance,but some influence on environment
  - What positioning accuracy do I really need and why?
    - How often do I need to determine an object's location?
    - How big is the a real need to cover?

- Measurement
  - Trilateration
    - Time of Arrival(TOA)
      - Distance=propagation time*speed of sound(around 335 meter/second)
      - measurement

        With three measurements, we have

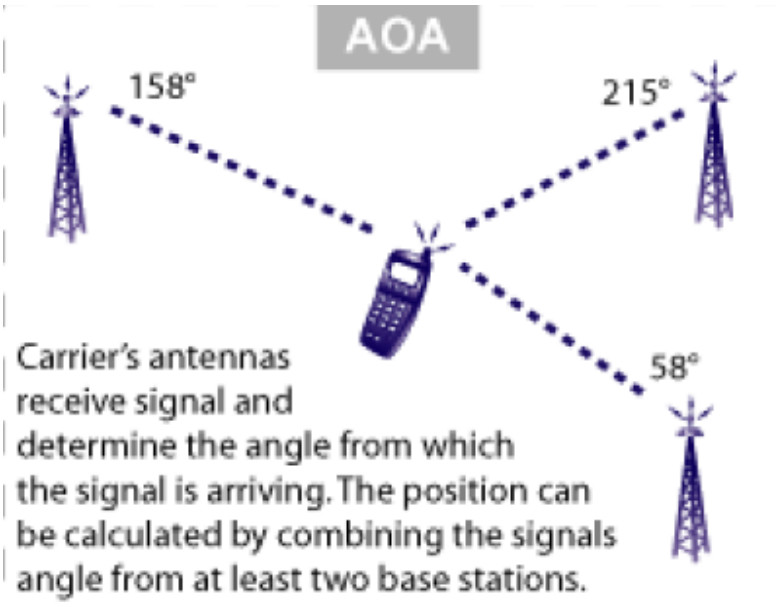
        $$(x-x_1)^2+(y-y_1)^2=r_1^2$$
        $$(x-x_2)^2+(y-y_2)^2=r_2^2$$
        $$(x-x_3)^2+(y-y_3)^2=r_3^2$$

    - Time Difference of Arrival(TDOA)

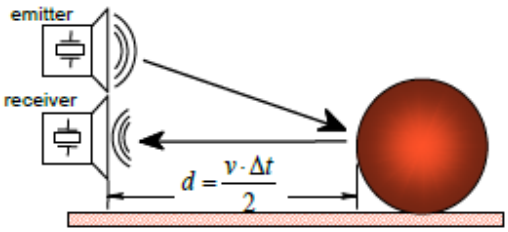
      The travel time of a signal from a reference station to the current position is given by the distance divided by the signal propagation speed v, we can obtain the time difference of arrival Δt_{AB} (between source A and B) and Δt_{BC} (between source B and C) as:

      $$\Delta t_{AB}=\frac{1}{v}\sqrt{(x-x_1)^2+(y-y_1)^2}-\frac{1}{v}\sqrt{(x-x_2)^2+(y-y_2)^2}$$
      $$\Delta t_{BC}=\frac{1}{v}\sqrt{(x-x_2)^2+(y-y_2)^2}-\frac{1}{v}\sqrt{(x-x_3)^2+(y-y_3)^2}$$
  - Anugulation

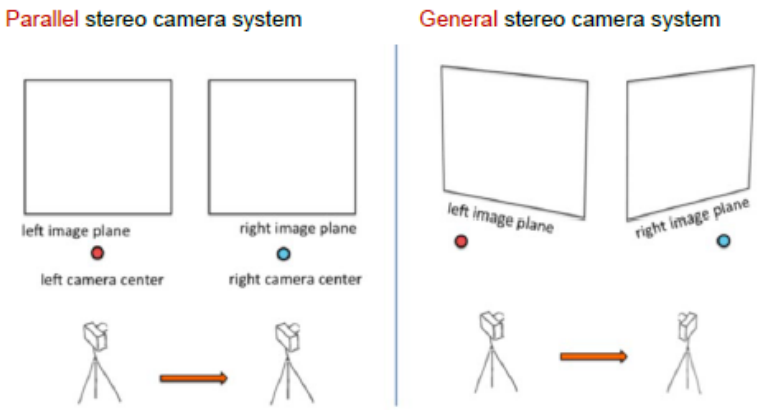
    

    - Angle of arrival(AOA)

    - Base station measures angle to mobile terminal
    - Rotate antenna to the highest RSS value
    - Derive angle from RSS values of individual antennas in an antenna array
- Ultrasonic range sensor



    - Main characteristics
        - Precision influenced by angle to object
        - Useful in ranges from several cm to several meters
        - Typically relatively inexpensive
    - Applications
        - Distance measurement
- Distance estimation(between physical point to camera)
    - Ambiguity:Any point on the ray(the line from the camera centre O to the point P)can be projected on the same the image point.
    - Solution:A second camera can resolve the ambiguity,enabling measurement of depth via triangulation
    - stereo camera systems



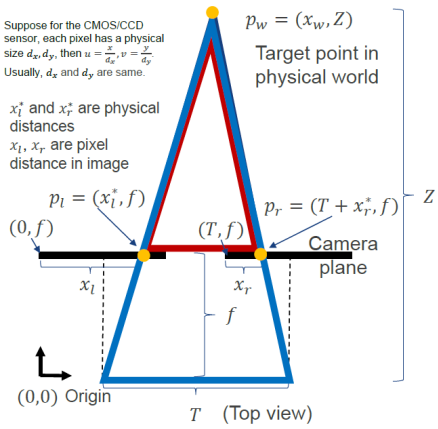
Given two calibrated parallel cameras, i.e. the right camera is some distance to the right of the left camera. According to triangle similarity theorem between blue triangle and red triangle, we have

$$\frac{T}{Z} = \frac{T + x_r^* - x_l^*}{Z - f}$$

$$x_l = \frac{x_l^*}{d_x},\ x_r = \frac{x_r^*}{d_x}$$

$$z = \frac{fT}{d_x(x_l - x_r)}$$

	Descriptions	Unit
Z	Distance between point p to camera	Physical distance, meter
T	Baseline distance between two cameras	Physical distance, meter
f	Focal length of the camera	Physical distance, meter
x_l, x_r	Locations of point p_l, p_r in images	Pixels
d_x, d_y	Physical size of a pixel in camera sensor CMOS/CCD	Physical distance per pixel

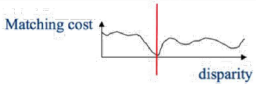


Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

- known: T, f, xl, xr ,dx,dy -> Z
- matching cost: SSD

$$SSD(\text{patch}_l, \text{patch}_r) = \sum_x \sum_y \left(I_{\text{patch}_l}(x,y) - I_{\text{patch}_r}(x,y) \right)^2$$

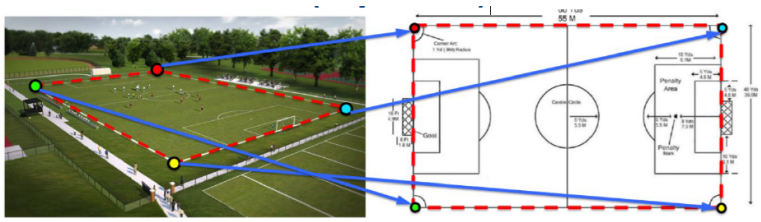
left image



Reference: http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

-

- Distance estimation (between two pixel positions in the image)
  - Transform the CCTV view to a bird-view (top-view)



  - various transformations

Transformation	Preserves
Translation	Orientation
Euclidean (rigid)	Lengths
Affine	Parallelism
Homography (projective)	Straight lines

- Homography matrix

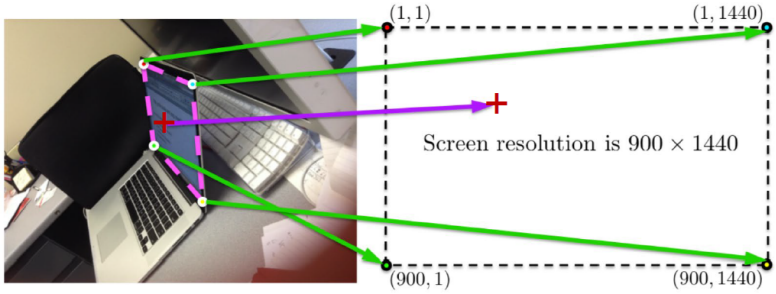
$$\begin{pmatrix} x1 \\ y1 \\ 1 \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \begin{pmatrix} x2 \\ y2 \\ 1 \end{pmatrix}$$

- rotation

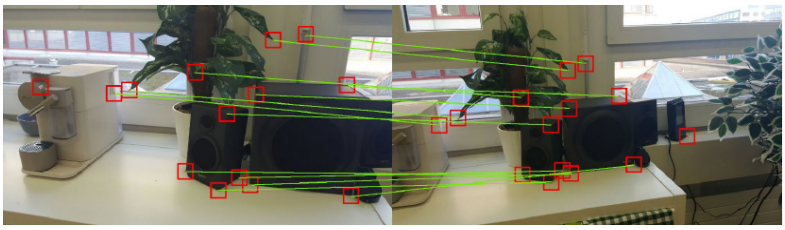
$$\mathbf{p}' = \mathbf{R} \cdot \mathbf{p}$$

$$\begin{bmatrix} \cos\theta\, x - \sin\theta\, y \\ \cos\theta\, y + \sin\theta\, x \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{R}} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

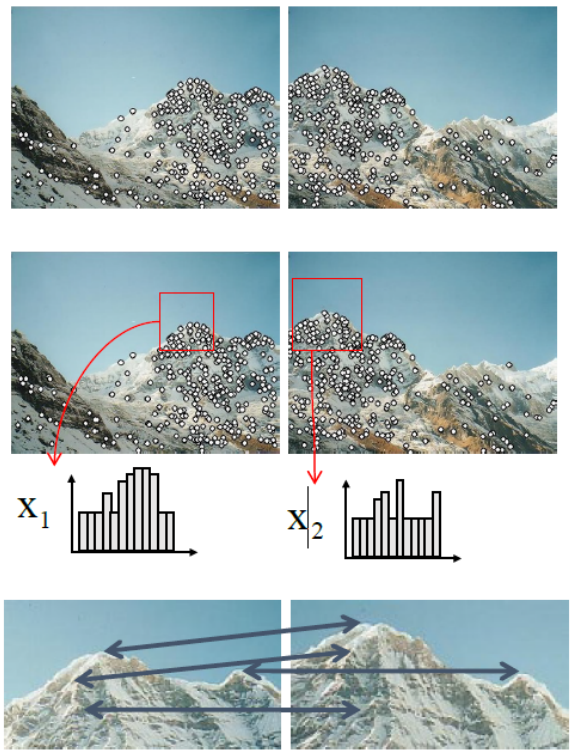
- Perspective correction



- Keypoint matching in multiple-view images

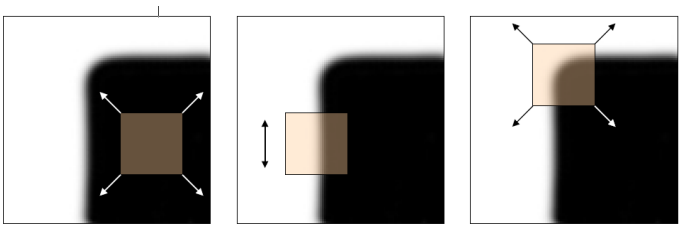


- Given a pair of matched keypoints(generated from the same physical point)in multiple-view images,we can estimate their disparity,then further estimate the distance between the physical point to the camera.
- Estimate the transformation(e.g.,Homography matrix)between multiple-view images,which can help to estimate distance between two pixel positions in the image.
- Method



- 1)Detection:Find a set of distinctive keypoints.
- 2)Description: Extract feature descriptor around each interest point as vector, such as $\mathbf{X}1$,$\mathbf{X}2$.
- 3)Matching: Compute distance between feature vectors to find correspondence based on user-defined threshold T, i.e., $d(\mathbf{X}1,\mathbf{X}2)<T$

- Harris corner detector



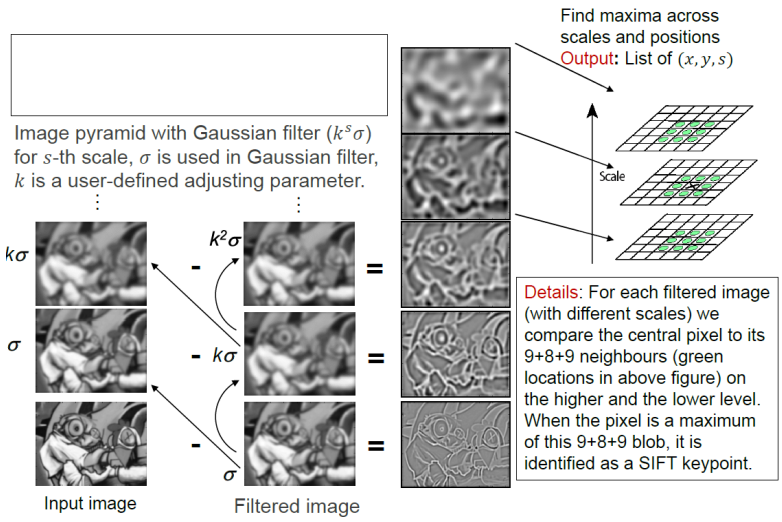
Flat: No change in all directions | Edge: No change along the edge direction | Corner: Significant change in all directions

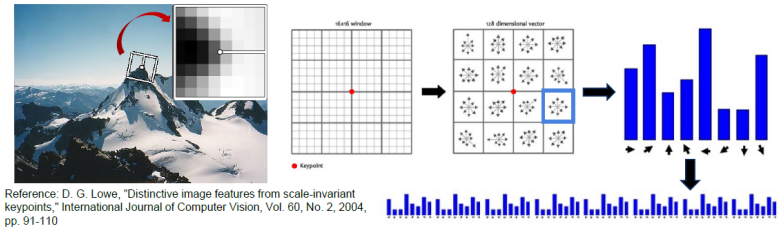
Objective: Find patches (a window $\Omega(x,y)$ centered at the location (x,y)) that generate a large variation when it is moved around with a shift value (u,v).

$$E_{u,v} = \sum_{\Omega(x,y)} w(x,y)(I(x+u,y+v) - I(x,y))^2$$

- $E_{u,v}$ is the difference between the original patch centered at $I(x,y)$ and that covered by the shifted window centered at $I(x+u,y+v)$.
- (u,v) are the window's displacements in the x,y directions, respectively.
- $w(x,y)$ is the mask function at position (x,y), e.g., uniform function or Gaussian function.
- We look for a patch with a large variation among $E_{0,0}$, $E_{1,0}$, $E_{0,1}$, $E_{1,1}$, etc.

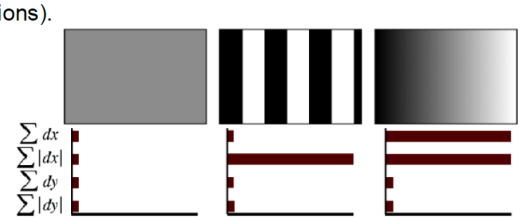
- SIFT(scale-invariant feature transform): Feature extraction





Reference: D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, Vol. 60, No. 2, 2004, pp. 91-110

Each sub-region (i.e., square used in SIFT) has a four-dimensional descriptor vector for its underlying intensity structure $\mathbf{V} = (\sum dx, \sum|dx|, \sum dy, \sum|dy|)$. This results in a descriptor vector for all 16 sub-regions of length $64 = 16 \times 4$. (In SIFT, each descriptor has $128 = 16 \times 8$ dimensions).



- Detect For each keypoint(at specific location$x$,y and scale$s$),warp the region(in the feature map at that specific scale,not the original image)around it to canonical orientation and resize the region to 16×16 pixels.Create histogram of local gradient directions.Assign canonical orientation at peak of the histogram.Rotate the patch so that the dominant orientation points rightward.This makes the patches rotation invariant.
- Divide the region into 4×4 squares(totally16).Each square has 4×4 pixels.For each square,compute gradients for each pixels,then compute gradient direction histogram over 8 directions(bins).Concatenate the histograms computed from 16 squares to obtain a 128(16×8=128)dimensional feature.

- Feature matching

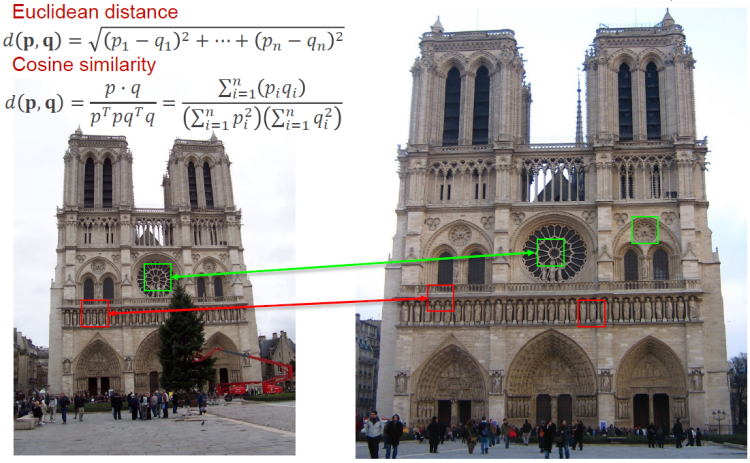
Given features $\mathbf{p}$ and $\mathbf{q}$ that are illustrated as squares in left/right images, respectively.

- Euclidean distance

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2}$$

- Cosine similarity

$$d(\mathbf{p},\mathbf{q}) = \frac{p \cdot q}{p^T p q^T q} = \frac{\sum_{i=1}^{n}(p_i q_i)}{\left(\sum_{i=1}^{n} p_i^2\right)\left(\sum_{i=1}^{n} q_i^2\right)}$$

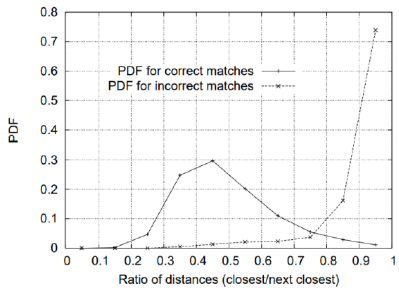


Compare distance of *the closest (NN1)* and the *second-closest (NN2)* feature vector neighbor.

$$\text{If } NN1 \approx NN2 \quad Ratio\ \frac{NN1}{NN2} \approx 1 \quad \text{Ambiguity matches}$$

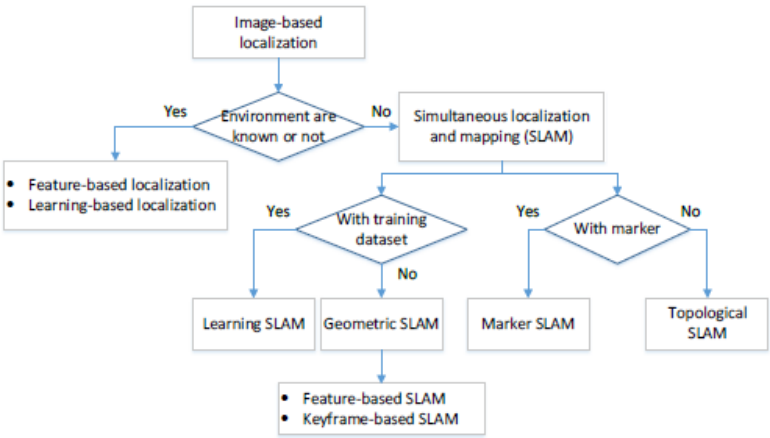
$$\text{If } NN1 \ll NN2 \quad Ratio\ \frac{NN1}{NN2} \to 0 \quad \text{Good match}$$

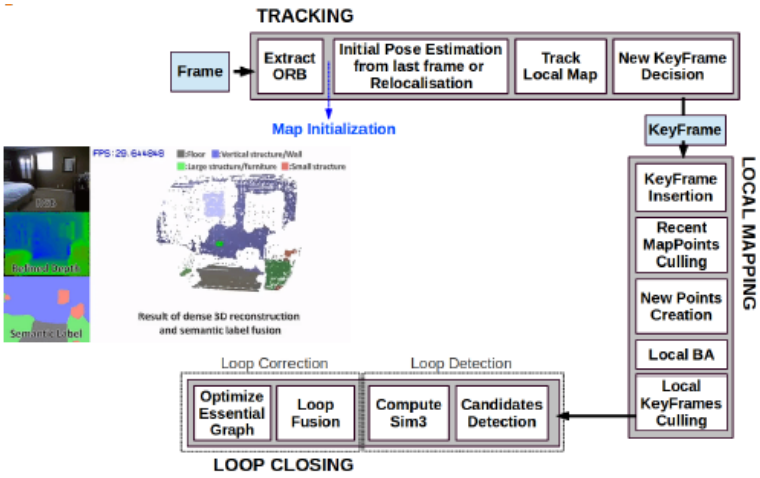
- Sort matches in the order of this ratio, then choose a threshold to select the reliable matches.

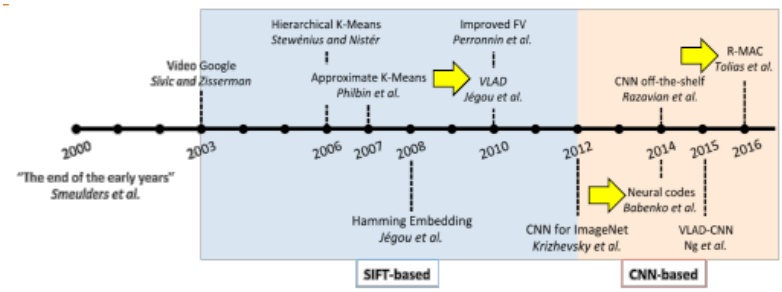


The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the *probability density function* (PDF) of this ratio for correct matches, while the dotted line is for matches that were incorrect.

- Image-based location and place recognition



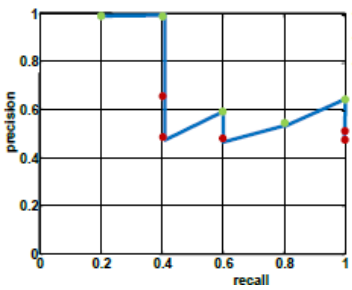




- Image retrieval:Have I seen this image before?Which images in my database look similar to it?
- Challenge
  - Lighting changes: Different time of day
  - Changes in camera viewpoint
  - Occlusions and ambiguous objects: People, cars, trees.
- Performance metric
  - precision rate > recall rate
    - Precision = #relevant / #returned
    - Recall = #relevant / #total relevant
  - Mean average precision (mAP) = mean of average precision over all queries

Ranked list of returned results with True/False labels (in previous slide example).

K	1	2	3	4	5	6	7	8	9	10
Label	T	T	F	F	T	F	T	T	F	F
TP	1	2	2	2	3	3	4	5	5	5
P	1	1	2/3	2/4	3/5	3/6	4/7	5/8	5/9	5/10
GTP	Supposed to be 5 for this query. It depends on dataset.									



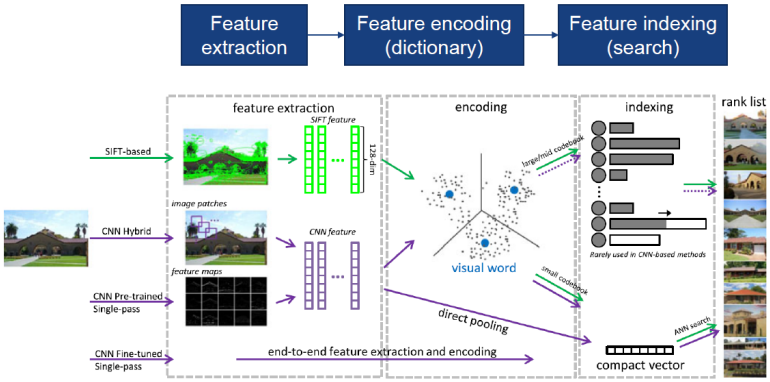
- K: current rank
- TP: true positives
- P: precision $= {}^{TP}/_{K}$
- GTP: total number of ground truth positives in the dataset

Summation of precision values of correct results / GTP

$$\frac{\left(1+1+\frac{3}{5}+\frac{4}{7}+\frac{5}{8}\right)}{5}$$

- Average precision = average precision (for a single query)
- Mean average precision (mAP) = mean of average precision over all queries

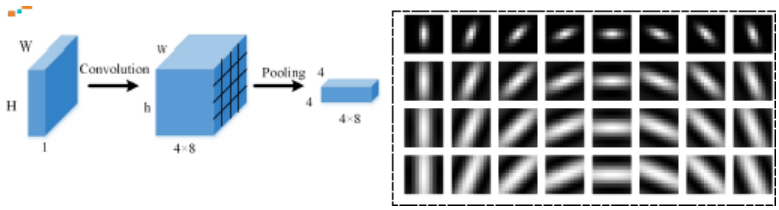
- Place recognition pipeline



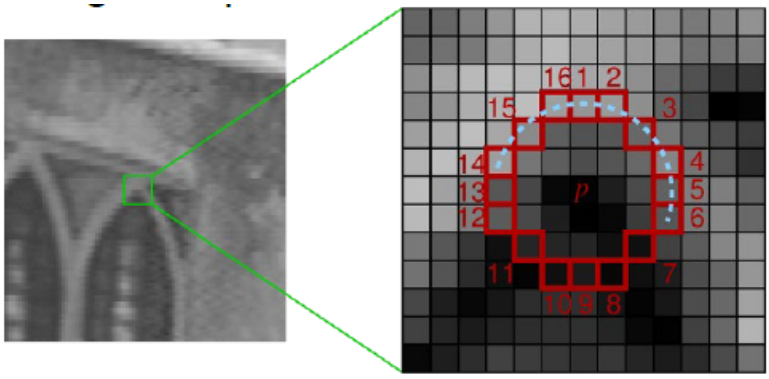
- Feature extraction

Features
Global feature: GIST
Point feature: SIFT, SURF
Point feature: ORB
Patch (blob) feature HoG, LBP
Learned feature: CNN-based

  - GIST

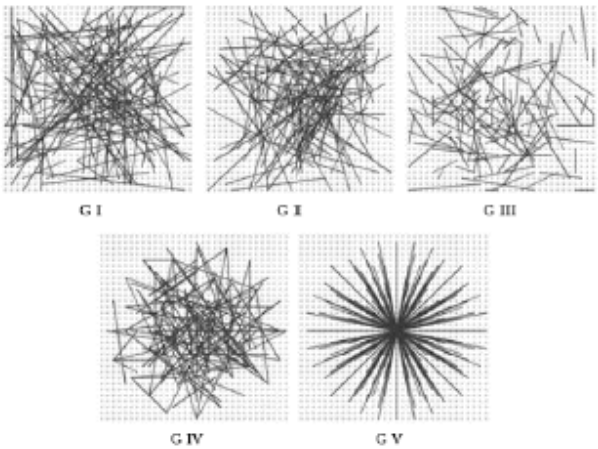


    - Given an input image,a GIST descriptor is computed by convolving the image with 32 Gabor filters(at 4 scales,8 orientations),producing 32 feature maps of the same size of the input image.
    - Divide each feature map into 16 cells(by a 4×4 grid),and then average the feature values with in each cell.
    - Concatenate the 16 averaged values of all 32 feature maps,resulting in a 16×32=512 GIST descriptor.
    - Intuitively,GIST summarizes the gradient information(scales and orientations)for different parts of an image
    - dimensions= 8 orientations * 4 scales * 16 bins
  - Point feature: ORB (Oriented FAST and rotated BRIEF)
    - FAST (Features from accelerated segment test)



- Objective: Determine a pixel pp (intensity value Ip) in the image as an interest point or not based on its neighboring pixels (say a circle of 16 pixels).
- Determine the pixel p is a corner, if there exists a set of nn continuous pixels in the circle (of 16 pixels) which are all brighter than $Ip + t$, or all darker than $Ip - t$, with an appropriate threshold value tt.
- Faster version: First compare the intensity of pixels 1, 5, 9 and 13 of the circle with Ip. At least three of these four pixels should satisfy the threshold criterion so that the interest point will exist.
  - If at least three of the four-pixel values $I1$, $I5$, $I9$, $I13$ are not above or below $Ip + t$, then p is not an interest point (corner). In this case reject the pixel pp as a possible interest point.
  - Else: check all 16 pixels and check if 12 contiguous pixels fall in the criterion.

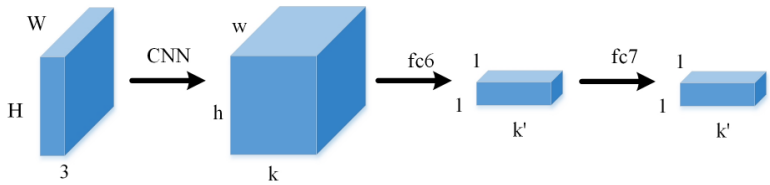
- Rotation calibration: It computes the intensity weighted centroid of the patch with located corner at center. The direction of the vector from this key point to centroid gives the orientation.
- Brief (Binary robust independent elementary feature)



  - Sample a pair of pixels a and b, according to sampling geometry patterns (five figures at below).
  - A vector of binary code: 1, if $a > b$, else 0.
  - Dimension of this feature: Number of pairs

- CNN

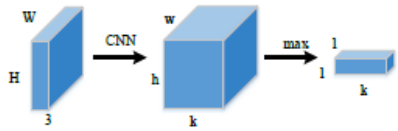
- Neural code



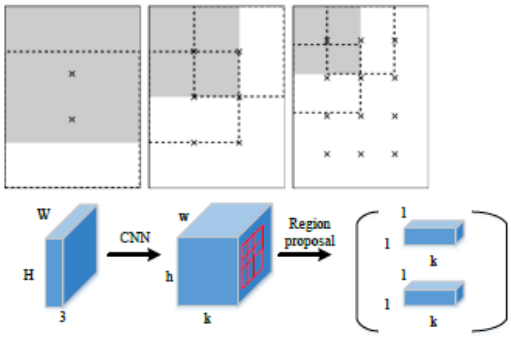
  - Use of feature activation from the top layers of CNN network as high level descriptor
  - 3-channel RGB input, 227 × 227
  - AlexNet last pooling layer, global descriptor of dimension $w \times h \times k = 6 \times 6 \times 256 = 9216$
  - Alternatively, fully connected layers fc6, fc7, global descriptors of dimension $k' = 4096$
  - Maximum activations of convolutions (MAC)

Given a set of 2D convolutional feature channel responses $X = \{X_i\}, i = 1, 2, \cdots k$, spatial max-pooling over all location is given as $f = [f_{\Omega,1}, \cdots, f_{\Omega,k}]$, where $f_{\Omega,i} = \max_{p \in \Omega} X_i(p)$, Ω is the set of valid spatial locations, $X_i(p)$ is the response at the particular position p, k is the number of feature channels

Global feature vector
(max-pooling per activation map)



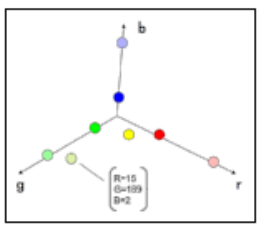
- Sampling region: Sample regions extracted at 3 different scales. We show the top-left region of each scale (gray colored region) and its neighbouring regions towards each direction (dashed borders). The cross indicates the region centre.
- Regional feature vector: Fixed multi-scale overlapping spatial region pooling.



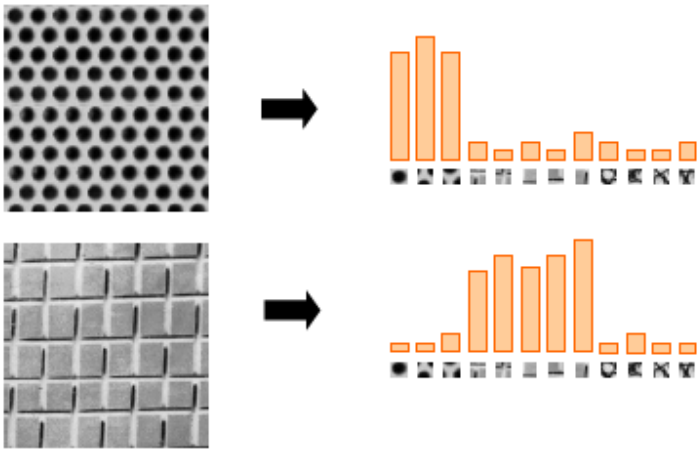
- Feature encoding(dictionary)
  - Histogram

C	$=$	$\{$	0	1	2	3	4	$\}$		
3	$\rightarrow$	(	0	0	0	1	0	)		
2	$\rightarrow$	(	0	0	1	0	0	)		
0	$\rightarrow$	(	1	0	0	0	0	)		
3	$\rightarrow$	(	0	0	0	1	0	)		
2	$\rightarrow$	(	0	0	1	0	0	)		
2	$\rightarrow$	(	0	0	1	0	0	)	$+$	
h	$=$	(	1	0	3	2	0	)	$/$	6

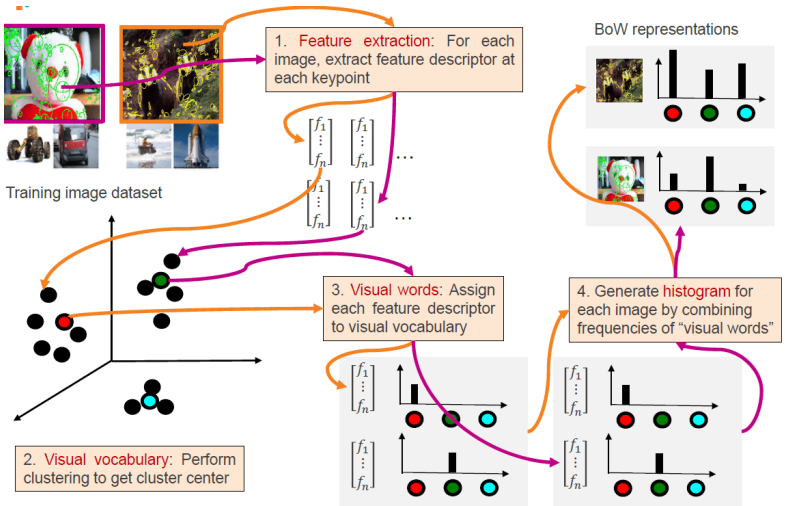
An example on color space



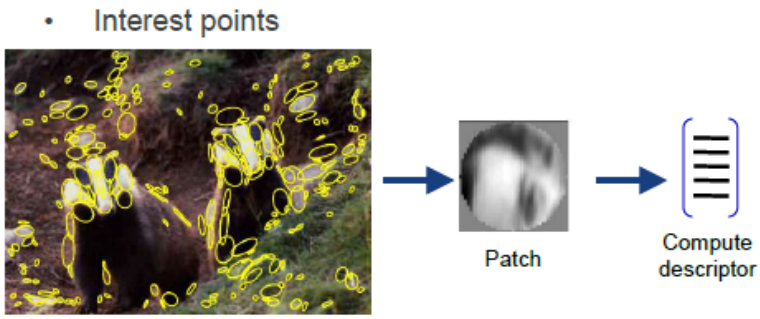
- Consider a histogram h over integers $C = \{0,1,2,3,4\}$, computed from the following samples.
- Each sample is encoded (hard assigned into one vector, all such vectors are pooled (averaged) into one vector.
- C is a codebook or vocabulary.
- Texture is characterized by the repetition of basic elements or textons. For stochastic textures, it is the identity of the textons, not their spatial arrangement.



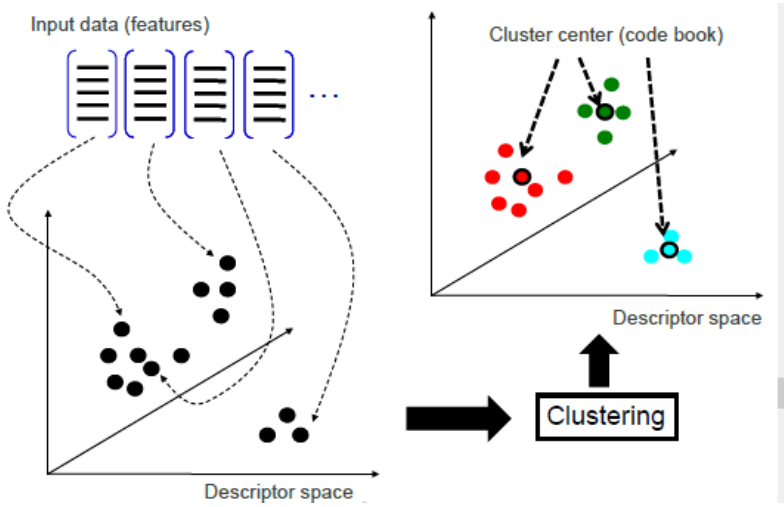
- Bag-of-words models(BOW)



- Orderless document representation:frequencies of words from a dictionary.
- Feature extraction

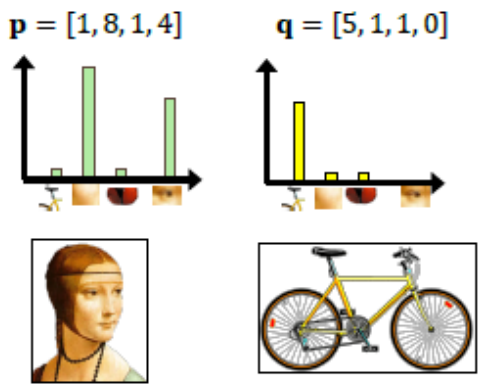


- Learn visual vocabulary



- Clustering is a common method for learning a visual vocabulary or codebook
- Unsupervised learning process
- Each cluster center produced by k-means becomes a codevector
- Codebook can be learned on separate training set
- The codebook is used for quantizing features
- A vector quantizer takes a feature vector and maps it to the index of the nearest codevector in a codebook
- Codebook = visual vocabulary
- Codevector = visual word

- Similarity evaluation



Histogram Intersection

$$\mathbf{H}_1 = (10, 0, 0, 0, 100, 10, 30, 0, 0)$$
$$\mathbf{H}_2 = (0, 40, 0, 0, 0, 6, 0, 110, 0)$$
$$S = \sum_{i=1}^{N} \min(H_1(i), H_2(i)) = 6$$

Euclidean distance

$$\mathbf{H}_1 = (10, 0, 0)$$
$$\mathbf{H}_2 = (0, 40, 0)$$
$$S = \sqrt{\sum_{i=1}^{N} (H_1(i) - H_2(i))^2} = 41.23$$

Manhatten distance

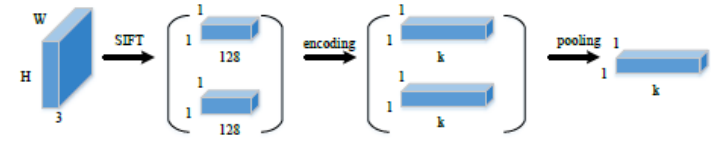
$$\mathbf{H}_1 = (10, 0, 0)$$
$$\mathbf{H}_2 = (0, 40, 0)$$
$$S = \sum_{i=1}^{N} |H_1(i) - H_2(i)| = 50$$

- Evaluate similarity of two images based on their BoW representations

- Example

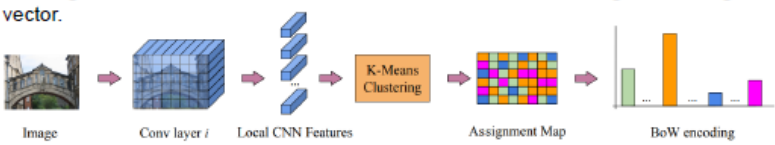
Example: SIFT

Given a gray–scale image input. $N \times 128$ descriptors (N is number of key points, 128 is the SIFT dimensions). Clustering/encoding (hard assignment) on k visual words). Note that N and k are user defined.



Example: CNN

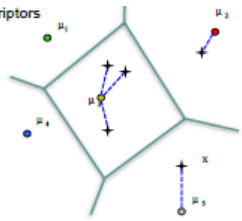
Use bag of words encode the local convolutional features of an image into a single vector.



- VLAD: Vector of Locally Aggregated Descriptors(ADD SPATIAL FEATURE)

Given a codebook $X = \{x_t, t = 1, \cdots, T\}$, $\{\mu_i, i = 1, \cdots, N\}$, learned with K-means, and a set of local descriptors

- ① assign: $NN(x_t) = \arg\min_{\mu_i} \|x_t - \mu_i\|$
- ②③ compute: $v_i = \sum_{x_t : NN(x_t) = \mu_i} x_t - \mu_i$
- concatenate v_i

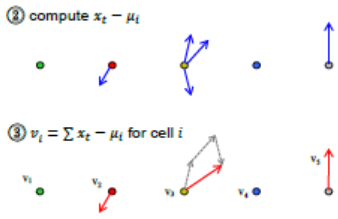


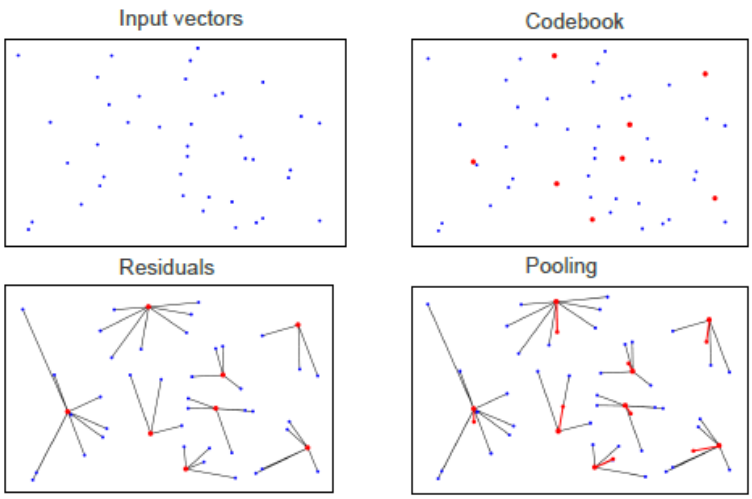
0/1 assignment of x_t to cluster i

$$v_i = \sum_t a_i(x_t)(x_t - c_i)$$

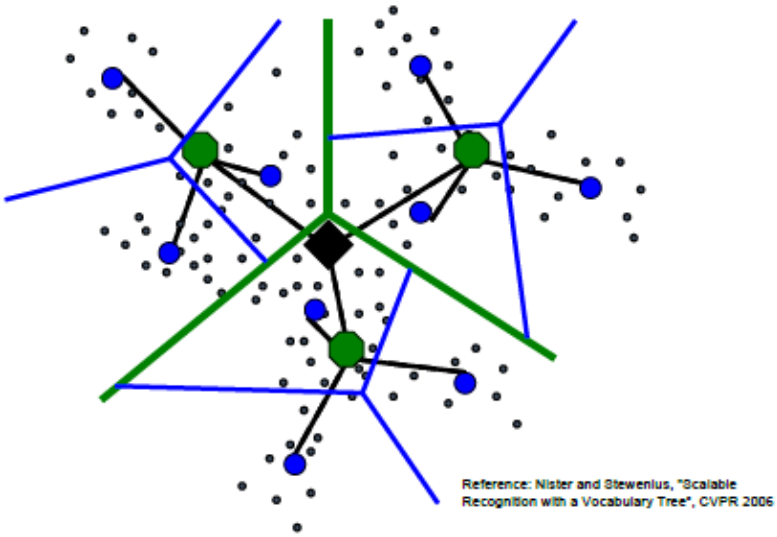
Residual vector

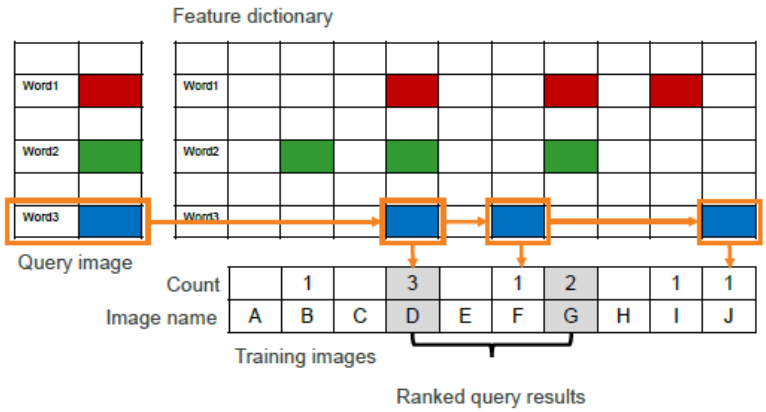
Sum over all (blue) descriptors in each cell. Then, all (red) residual vectors are normalized.



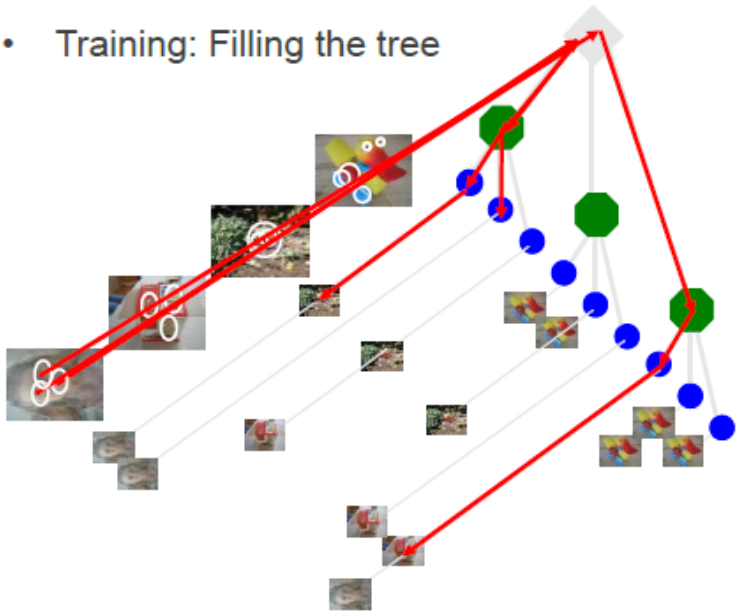


- Feature indexing(search)
  - Vocabulary trees: Hierarchical clustering for large vocabularies (fast to compute)

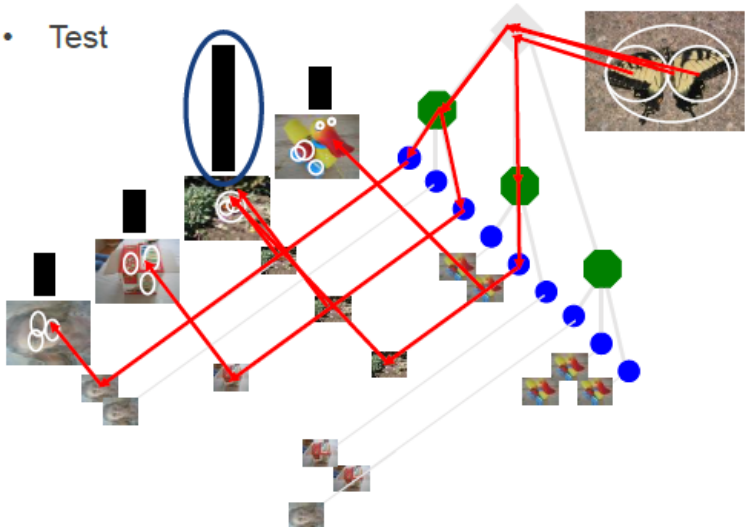




- Training: Filling the tree



- Test



- Post-processing

  - Re-ranking: Perform spatial matching only on top-ranking images, and re-ranking according to a score based on geometry, e.g. number of inliers.
  - Query expansion (QE): A number of top-ranked images from the original rank list are employed to issue a new query which is in turn used to obtain a new rank list.

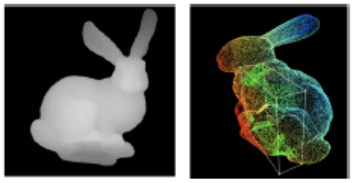
- 3D image

Comparison of 3D imaging technologies

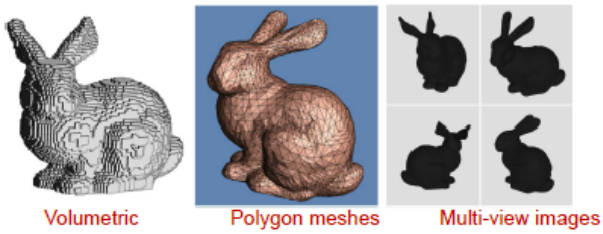
	Time of Flight (ToF)	Stereoscopic vision	Fixed structured light
Operational principle	**IR pulse, measure light transit time**	**Two 2D sensors emulate human eyes**	**Single pattern visible or IR illumination, detects distortion**
Point cloud generation	Direct out of chipset	High SW Processing	Medium SW processing
Latency	Low	Medium	Medium
Active illumination	Yes	No	Yes
Low light performance	Good	Weak	Good
Bright light performance	Medium	Good	Medium / weak *Depends on illumination power*
Power consumption	Medium/high *Scales w/ distance*	Low	Medium
Range	Short to long range *Depends on laser power & modulation*	Mid range *Depends on spacing between cameras*	Very short to mid range *Depends on illumination power*
Resolution	QQVGA, QVGA -> Roadmap to VGA	Camera Dependent	Projected pattern dependent
Depth accuracy	mm to cm *Depends on resolution of sensor*	mm to cm *Difficulty with smooth surface*	mm to cm
Scanning speed	Fast *Limited by sensor speed*	Medium *Limited by software complexity*	Medium *Limited by SW complexity*

- Indoors, e.g., low-cost sensors such as the Microsoft Kinect
  - Short range sensors for indoor scenes
  - Real-time 30 FPS depth maps based on structured light or time of flight in infrared
  - Integrated RGB camera is better aligned and provides better quality under indoor conditions than LiDAR systems
  - RGB-D image grid makes it well suited for traditional 2D computer vision techniques on image frames from a single view
- Outdoors, e.g., LiDAR (Light Detection and Ranging)
  - Long range sensors for outdoor scenes
  - Fast scans at low resolution or slow scans at high resolution, depending on number of individual sensors
  - Moving sensors and registration from multiple scans result in unstructured point cloud data with no adjacency grid
- 3D data representation

Rasterized form (regular grids)	Multi-view images	A 2D projected view
	Depth image	A 2.5D format
	Volumetric	A occupancy version of 3D points
Geometric form (irregular)	Polygonal mesh	A surface version of 3D points
	Point cloud	3D points

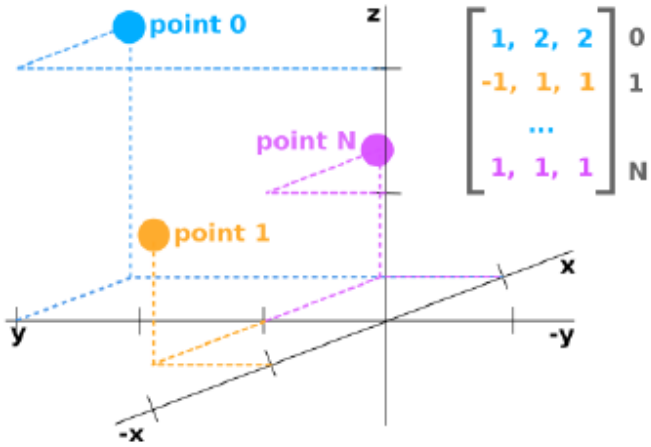


Depth map Point cloud

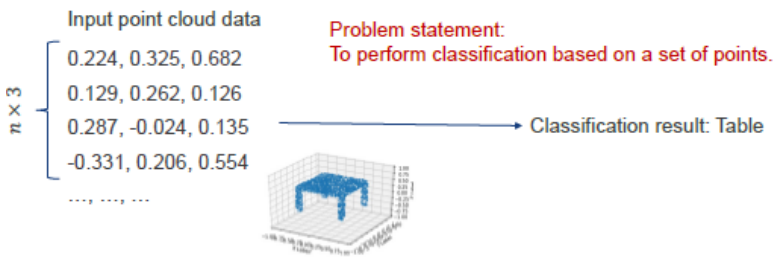


Volumetric Polygon meshes Multi-view images

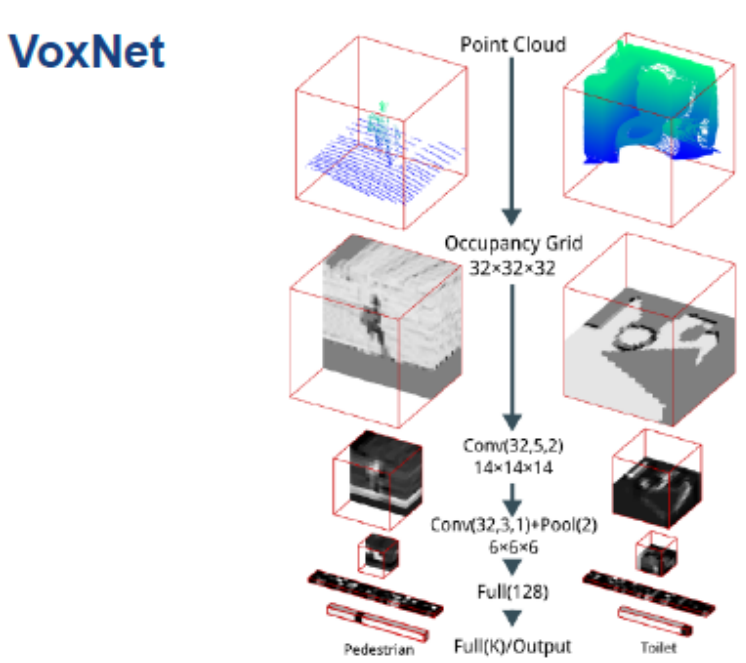
- Image
  - The coordinate values in an image are always positive.
  - The origin is located on the upper left hand corner.
  - The coordinates are integer values.
- Point cloud



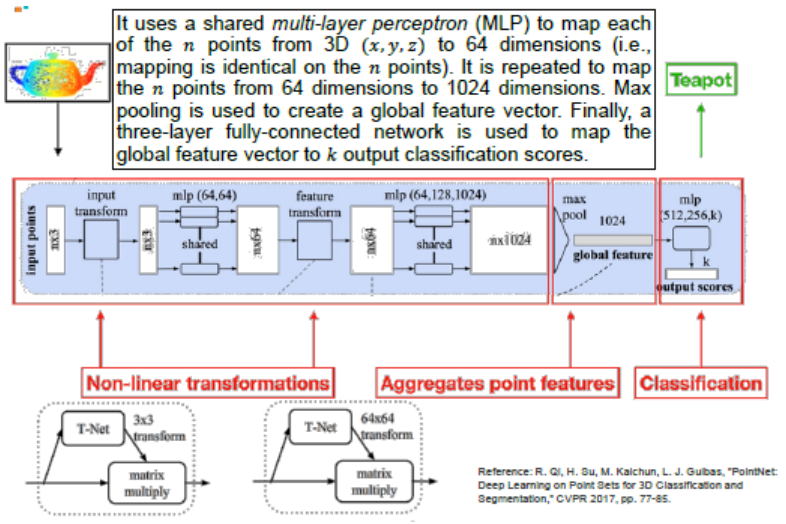
  - The coordinate values in point cloud can be positive or negative.
  - The coordinates can take on real numbered values.
  - The positive x axis represents forward.
  - The positive y axis represents left.
  - The positive z axis represents up.
  - MODEL

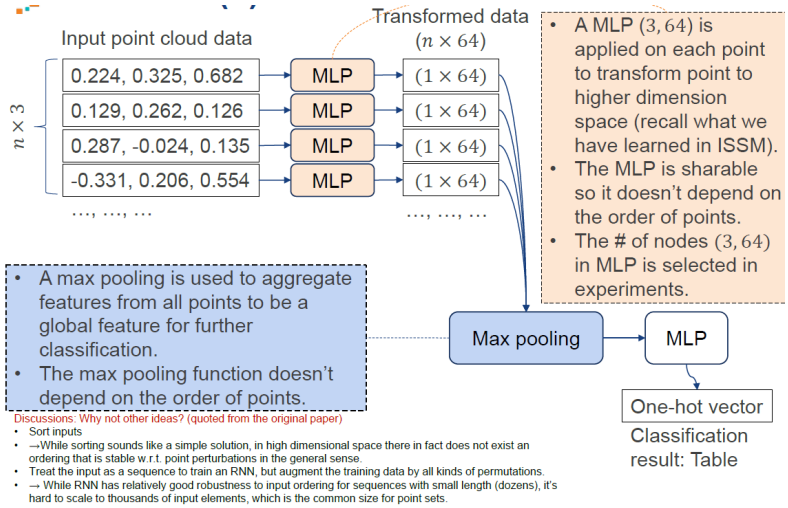


    - VoxNet



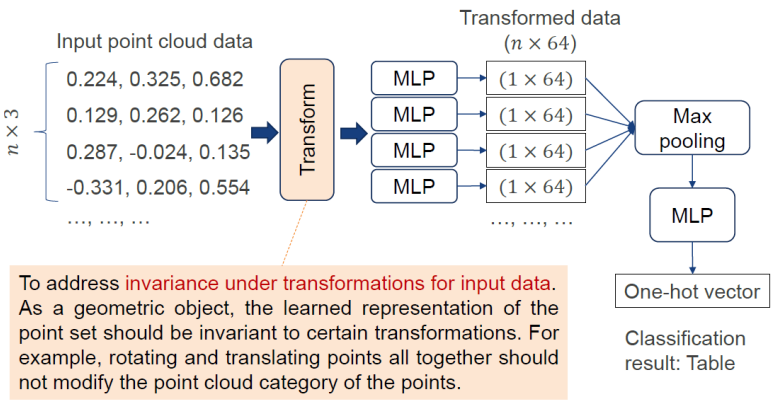
- PointNet



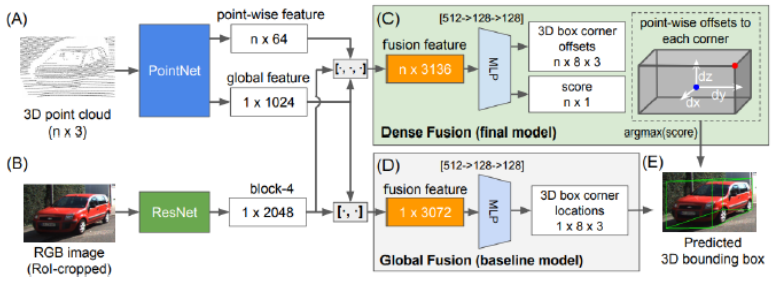


Discussions: Why not other ideas? (quoted from the original paper)

- Sort inputs
- →While sorting sounds like a simple solution, in high dimensional space there in fact does not exist an ordering that is stable w.r.t. point perturbations in the general sense.
- Treat the input as a sequence to train an RNN, but augment the training data by all kinds of permutations.
- → While RNN has relatively good robustness to input ordering for sequences with small length (dozens), it's hard to scale to thousands of input elements, which is the common size for point sets.

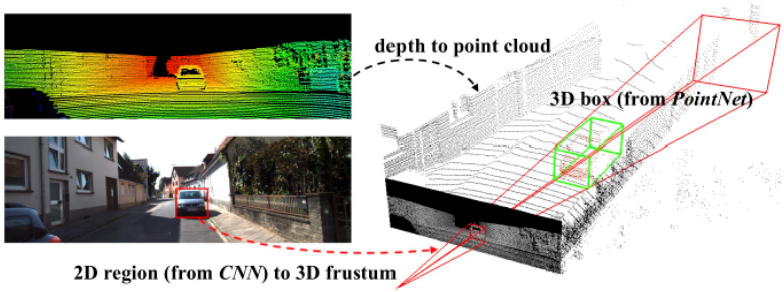


- Point fusion



  - Point Fusion has two feature extractors: a PointNet variant that processes raw point cloud data, and a CNN that extracts visual features from an input image. We present two fusion network formulations: a vanilla global architecture that directly regresses the box corner locations (D), and a novel dense architecture that predicts the spatial offset of each of the 8 corners relative to each input point, as illustrated in (C): for each input point, the network predicts the spatial offset (white arrows) from a corner (red dot) to the input point (blue), and selects the prediction with the highest score as the final prediction (E).

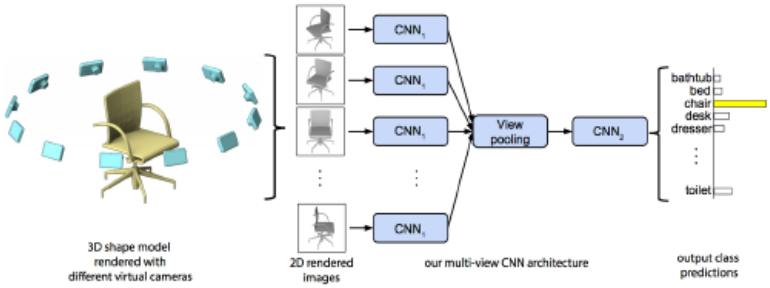
- Frustum PointNet



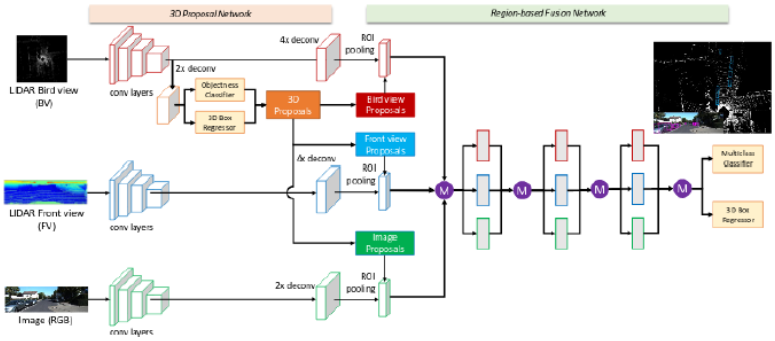
  - firstly build object proposals with a 2D detector running on RGB images, where each 2D bounding box defines a 3D frustum region. Then based on 3D point clouds in those frustum regions, we achieve 3D instance segmentation and amodal 3D bounding box estimation, using PointNet network.

- Multi-view images
  - MVCNN

- First CNN: Extract image features from individual view
- View pooling: Element-wise max-pooling across all views.
- Second CNN: Extract shape features from pooled image

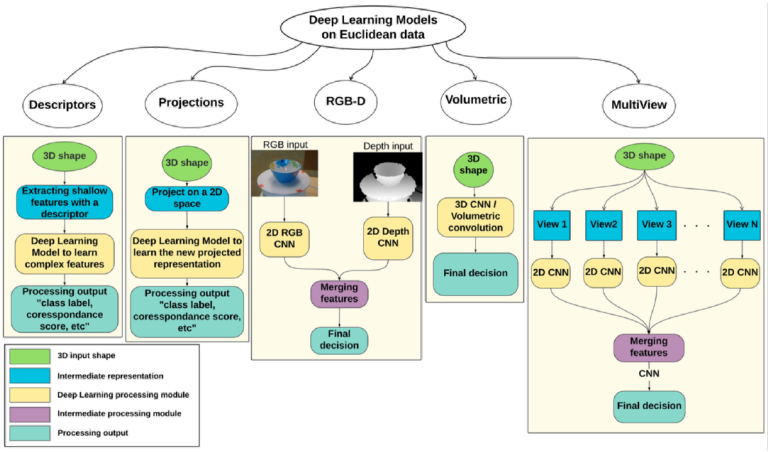


- MV3D

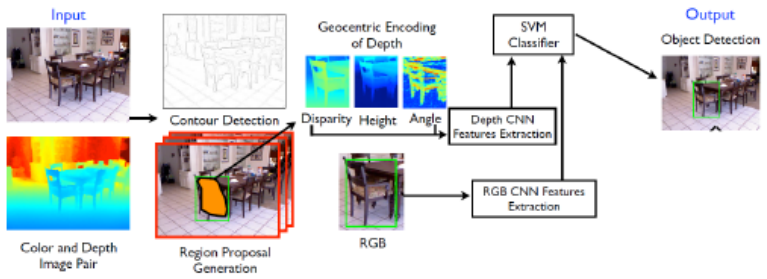


  - Multi-view V3D (MV3D) simply takes two separate 2D views of the point cloud: one from the front and one from the top (birds' eye). MV3D also uses the 2D camera image associated with each LiDAR scan. That provides three separate 2D images (LiDAR front view, LiDAR top view, camera front view).

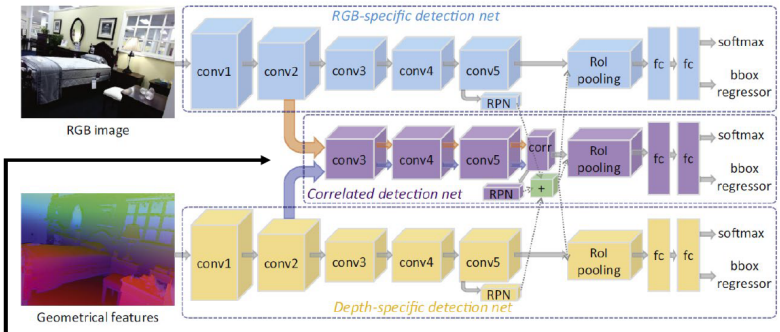
- 3D data machine learning



- 3D Object detection



Objective: How to add depth image as input with RGB input image together?

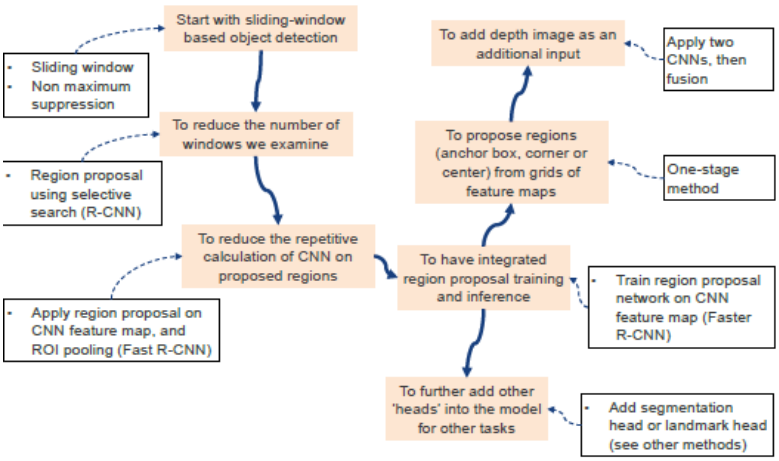


- Shared convolution layers ($conv3$, $conv4$, $conv5$, suggested by the authors)
- $corr$ means a parameter-free correlation layer, which performs multiplicative comparisons (element-wise product) between feature maps of two modalities.
- Geometrical features are 3-channel features calculated from the 1-channel depth image.

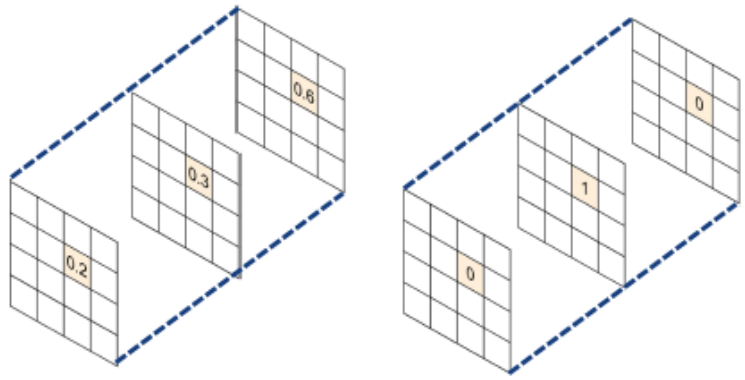
Input: RGB-D data

Output: A modal 3D bounding boxes and semantic class labels for objects in the scene.

	Advantages	Disadvantages
2D Object Detection	Well established datasets and detection architectures. Usually RGB only input can achieve accurate results in the image plane.	Limited information: lack of object's pose, occlusion and 3D position information.
3D Object Detection	3D bounding box provides object size and position in world coordinates. These detailed information allows better environment understanding.	Requires depth estimation for precise localization. Extra dimension regression increases model complexity. Scarse 3D labelled datasets.



- Semantic segmentation
  - How to represent the target label image?
    - Grayscale images: The pixel intensity represents the class id.
    - Color images: Different classes of objects will have some arbitrary RGB color for visualization. It requires pre-processing to be converted into either class labels, or One Hot Vector representations, before it can be used in a deep learning pipeline.
    - One hot vectors: Each pixel is encoded as a one-hot vector, with a value of 1 for the class it represents.
  - Performance metric

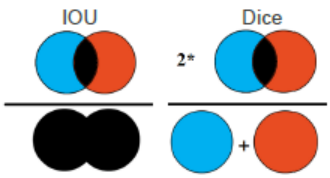


Prediction for a selected pixel from a machine learning model

Ground truth for the corresponding pixel (one hot vector)

- Pixel-wise cross-entropy loss is calculated based on model prediction (e.g., 0.2, 0.c, 0.6) and the ground truth one hot vector (e.g., 0, 1, 0).
- This calculation is repeated over all pixels and averaged.
- IOU(intersection over union) + Dice coefficient

  - IOU (intersection over union): Overlapping region divided by combine region.
  - Dice coefficient: $\frac{2|A \cap B|}{|A|+|B|}$, where $|A \cap B|$ represents the common elements between regions A and B, $|A|$ and $|B|$ represent the number of elements in set A and set B, respectively.



$$\text{Dice loss} = 1 - \frac{2\sum_{\text{pixels}}(y_{\text{true}} * y_{\text{pred}})}{\sum_{\text{pixels}}(y_{\text{true}}) + \sum_{\text{pixels}}(y_{\text{pred}})}$$

Example

$$A = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \quad |A \cap B| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.41$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad |A| = \begin{bmatrix} 0.01 & 0.03 & 0.02 & 0.02 \\ 0.05 & 0.12 & 0.09 & 0.07 \\ 0.89 & 0.85 & 0.88 & 0.91 \\ 0.99 & 0.97 & 0.95 & 0.97 \end{bmatrix} \xrightarrow{\text{sum}} 7.82 \quad |B| = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{\text{sum}} 8$$

$$\text{Dice loss} = 1 - \frac{2|A \cap B|}{|A| + |B|} = 1 - \frac{2 * 7.41}{7.82 + 8} = 0.0632$$

Note: For multiple classes, the Dice loss score is repeated over all pixels/classes and averaged.

- model

Semantic segmentation is fundamentally a pixel labelling task, can be used for other pixel-level labelling, such as colorization, depth estimation.

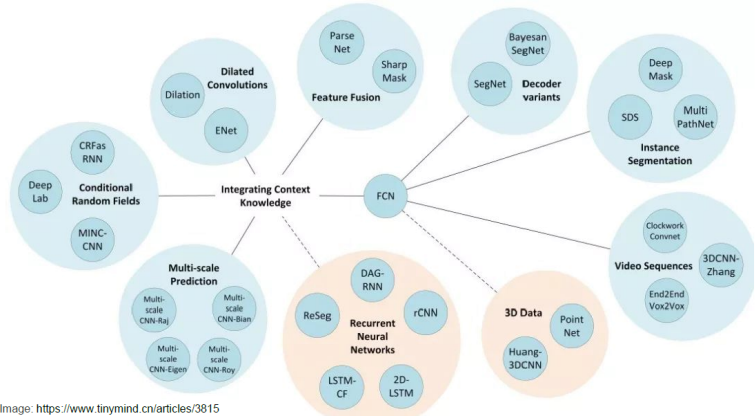
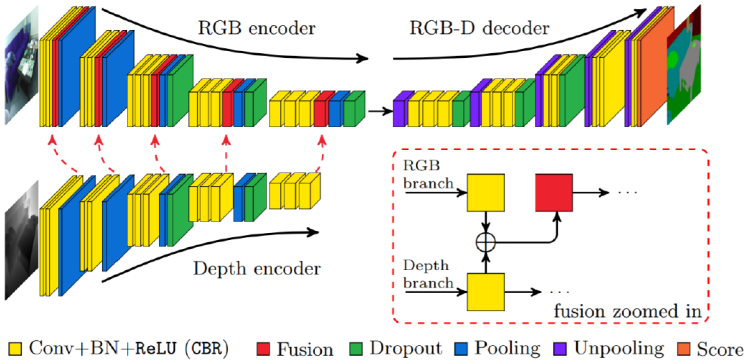


Image: https://www.tinymind.cn/articles/3815

- RGB-D semantic segmentation

- Idea 1: Add depth as the 4th channel of the input image.
- Idea 2: Fuse two streams of RGB and depth image, respectively.