

Computer Vision - summary

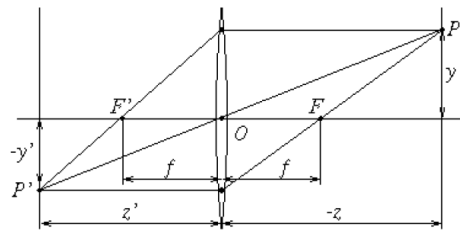
Part1intro	2
Part2image-processing.....	3
Part3image-processing.....	7
Part4edge-detection	12
Part5structure-extraction.....	14
Part6segmentation.....	18
Part7segmentation.....	21
Part8categorization	24
Part9categorization	28
Part10local-features.....	31
Part11local-features.....	34
Part12local-features.....	37
Part13local-features.....	38
Part14categorization	40
Part15categorization	42
Part16categorization	43
Part17reconstruction	46
Part18reconstruction	49
Part19reconstruction	53
Part20motion	54
Part21reconstruction	57
Questions:	61

Part1intro

- Pinhole camera:

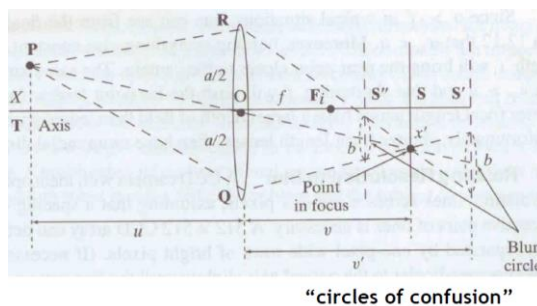
Pinhole size: big: many directions average, blur; small: diffraction effect, blur

- Lens:

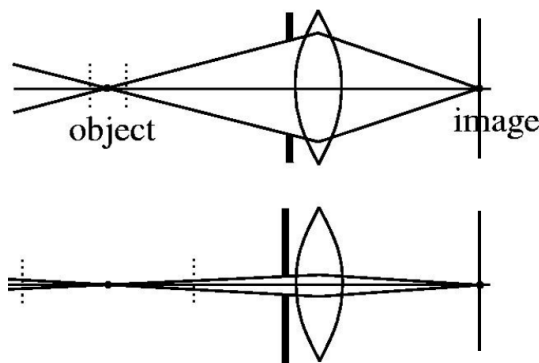


$$f/z + f/z' = 1$$

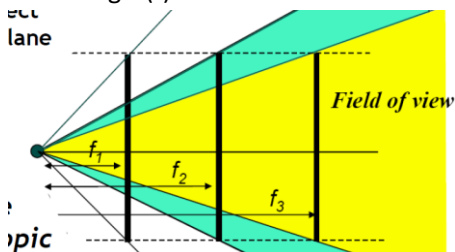
- Depth of field: distance between image planes where blur is tolerable(下图两圆之间的距离)



- Aperture(光圈):small: increase depth of field; big: decrease



- Focal length(f):



f gets larger, smaller part of the world, image becomes more telescopic

Part2image-processing

- Thresholding:

Two-sided

$$F_T[i, j] = \begin{cases} 1, & \text{if } T_1 \leq F[i, j] \leq T_2 \\ 0, & \text{otherwise} \end{cases} \quad (\text{grayscale image} \Rightarrow \text{binary mask})$$

- Global binarization(Otsu):

Search for threshold T that minimizes within-class variance(equivalent to maximize between-class variance)(n_1, n_2 是个数)

$$\begin{aligned} \sigma_{within}^2(T) &= n_1(T)\sigma_1^2(T) + n_2(T)\sigma_2^2(T) \\ \sigma_{between}^2(T) &= \sigma^2 - \sigma_{within}^2(T) \\ &= n_1(T)n_2(T) [\mu_1(T) - \mu_2(T)]^2 \end{aligned}$$

Algorithm:

1. Precompute a cumulative grayvalue histogram h
 2. For each potential threshold T
 - i. Separate the pixels into 2 clusters according to T
 - ii. Look up n_1, n_2 in h and compute both cluster means
 - iii. Compute variance between
 3. Choose $\arg \max T$
- Local binarization
在一个小的窗口 W 内，计算 μ 和 variance，以此确定局部 threshold， $k \in [-1, 0]$ is user-defined

$$T_W = \mu_W + k \cdot \sigma_W$$

- Additional improvement by surface fitting

Polynomial surface of degree d:

$$f(x, y) = \sum_{i+j=0}^d b_{i,j} x^i y^j$$

$b_{0,0} + b_{1,0}x_0 + b_{0,1}y_0 + b_{2,0}x_0^2 + b_{1,1}x_0y_0 + \dots + b_{0,3}y_0^3 = I_0$ (对于单个点 x_0, y_0 ，使用 polynomial 之后得到 I_0)

取 $1+2+3+4=10$ 个点，可以用 pseudo-inverse 求解 b, 即 $Ab=I \Rightarrow b=(A^T A)^{-1} A^T I$

$$\begin{bmatrix} 1 & x_0 & y_0 & x_0^2 & x_0 y_0 & \cdots & y_0^3 \\ 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & \cdots & y_1^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & y_n & x_n^2 & x_n y_n & \cdots & y_n^3 \end{bmatrix} \begin{bmatrix} b_{0,0} \\ b_{1,0} \\ \vdots \\ b_{0,3} \end{bmatrix} = \begin{bmatrix} I_0 \\ I_1 \\ \vdots \\ I_n \end{bmatrix}$$

接下来是如何使用上面的 polynomial surface fitting

Iterative algorithm:

1. Fit parametric surface to all points in the region

2. Subtract estimated surface
3. Apply global threshold
4. Fit surface to all background pixels in original region
5. Subtract estimated surface
6. Apply global threshold
7. Iterate further(4-6) if needed

说明：第一轮把 foreground 也给算进去了，之后不算，逐轮优化
上面可行的 assumption 是：most pixels belong to the background

- Morphological operators:

Dilation: if current pixel foreground, set all pixels under B to foreground

Erosion: if not every pixel under is foreground, set current pixel to background

p.s. 白色方框 foreground, dilation 之后是圆角, erosion 是直角

Opening(remove small objects): erosion, dilation; 就是在图内 roll 一圈外侧碰到的线路;

Closing(fill holes): dilation, erosion; 就是在图形外侧 roll 一圈, 看内测碰到的线路;

比如圆形, opening 就是使突出尖角变圆弧 (面积变小), closing 就是凹进去的角变圆弧 (面积变大)

- Morphological boundary extraction:

$$\beta(A) = A - (A \ominus B)$$

先腐蚀, 再求异; 如果用 3*3 方形来, 就会得到宽度为 1 的轮廓

- Dilation and erosion typically on binary images; if on grayscale, dilation take neighborhood max, erosion take min
- Connected components labelling:

Goal is to identify distinct regions

Connectedness: 4-connected, 8-connected

Algorithm(用的 4-connected, binary(black-1)):

Process image from left to right, top to bottom:

If the next pixel to process is 1

If only one of its neighbors(只考虑左侧以及上方) is 1, copy its label

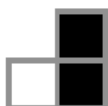
Else If both are 1

If have same label, copy it

Else copy label from the left, update the equivalence table

Else assign a new label

Re-label with the smallest of equivalent labels in the end



- Region properties:

$$A = \sum_{(x,y) \in R} 1$$

1. Area:

$$\bar{x} = \frac{1}{A} \sum_{(x,y) \in R} x$$

$$\bar{y} = \frac{1}{A} \sum_{(x,y) \in R} y$$

2. Centroid: (A 是上面的面积)

$$C = \frac{\mu_R}{\sigma_R}$$

3. Circularity: (分别是 centroid 到 **boundary points** 的 mean 和 variance, 只算最外一圈的点)

$$\mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \|(x_k, y_k) - (\bar{x}, \bar{y})\|$$

$$\sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} [\|(x_k, y_k) - (\bar{x}, \bar{y})\| - \mu_R]^2$$

4. Central moments(they are translation invariant):

0th central moment: area(注意是 0, 没 1)

2nd central moment: variance

3rd central moment: skewness

4th central moment: kurtosis

$$\mu_{jk} = \sum_{(x,y) \in S} (x - \bar{x})^j (y - \bar{y})^k$$

(central (j,k)th moment)

5. Hu moments

7 normalized central moments:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \gamma = \frac{p+q}{2} + 1$$

$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

上面 4 个外还有 3 个, 一般用 $\log_{10}(\Phi)$

Robust to translation, rotation, scaling; but don't expect wonders(仍然是汇总 summary 型数据)

6. 用 second moment 求 axis (为了实现 orientation 的矫正)

$$\begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} = V D V^T = \begin{bmatrix} v_{11} & v_{12} \\ v_{22} & v_{22} \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}^T$$

Mu 是按照上面计算的, 之后 svd, 给出的 eigenvector 就是目标, 红色是长轴的方向, 黄色是短轴 (可以看成是一个椭圆的定向)

- 对于本章 binary image 的处理手段
 - Pros: fast to compute, easy to store
 - Cons: hard to get clean silhouettes(轮廓)
 - Noise is common in reality
 - Too coarse(粗糙) for representation

Part3image-processing

- Common types of noise:
 - a) Salt&pepper noise: random black and white
 - b) Impulse noise: random white
 - c) Gaussian noise: variations in intensity drawn from a Gaussian

Basic assumption: noise is independent&identically distributed

- Correlation(其实就是一个方框，按照 filter 给的比重 H 求和):

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$$

Convolution 就是上面左右翻一下，上下翻一下(式子末尾加号都变减号)

Correlation 是圈承，convolution 是*

- If $H[-u, -v] = H[u, v]$, 那么 correlation 和 convolution 相同
- Property of convolution:

Shift invariant: behaves the same everywhere 只因 neighbor 内容不同而结果不同

Linear:

$$F(x_1 + x_2) = F(x_1) + F(x_2) \text{ additivity}$$

$$F(ax) = aF(x) \text{ homogeneity}$$

Commutative: $f * g = g * f$

Associative: $(f * g) * h = f * (g * h)$ 也就意味着 apply 多个 filters 等价于某一个 filter

Identity: $f * e = f$ (存在 $e = [\dots 0, 0, 1, 0, 0 \dots]$ 这个 filter 啥都不改变)

Differentiation:
$$\frac{\partial}{\partial x} (f \star g) = \frac{\partial f}{\partial x} \star g$$

- 然后用 filter 来和原图 correlation

Box filter(3*3 的正方形均为 1/9): ringing artifacts (这个问题是出现水平和竖直的线, 如果用圆形 filter, 就是圆形圈, 本质是转换到 frequency domain 有很多 artifact)

Gaussian filter: 没毛病

- Gaussian kernel:

$$G_{\sigma} = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Parameters:

Variance σ of gaussian: determine extent of smoothing

Size of kernel(mask): usually set filter half-width = 3σ

- Box filter and the Gaussian filter are separable

Gaussian(两个 1 维的):

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-x^2/(2\sigma^2)) \quad g(y) = \frac{1}{\sqrt{2\pi}\sigma} \exp(-y^2/(2\sigma^2))$$

可以看出是相乘的关系，box 则是 $3 \times 3 = 9$

- Boundary(在 filter 之后):

Full:增加一圈

Same:一样

Valid:缩小一圈

如果用 full, 怎么补外圈的几种策略:

Clip filter

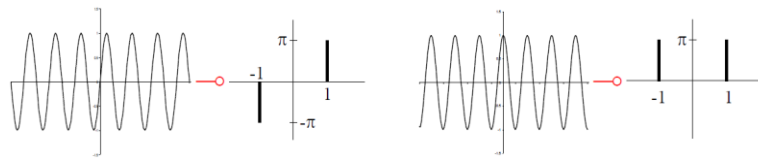
Wrap around

Copy edge

Reflect across edge

- Fourier transform of important functions

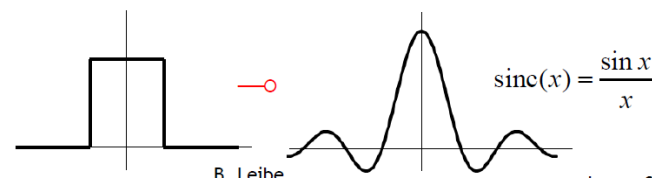
Sine and cosine->"frequency spikes"



Gaussian->Gaussian



Box filter->sinc



- Duality

The better a function is localized in one domain, the worse it is localized in the other

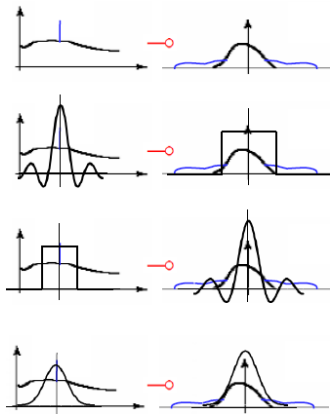
- Effect of filtering

Noise introduces high frequencies, we want to apply a "low-pass" filter

Ideal filter shape in frequency domain would be a box, but this transfers to a spatial sinc, which has infinite spatial support(难计算)

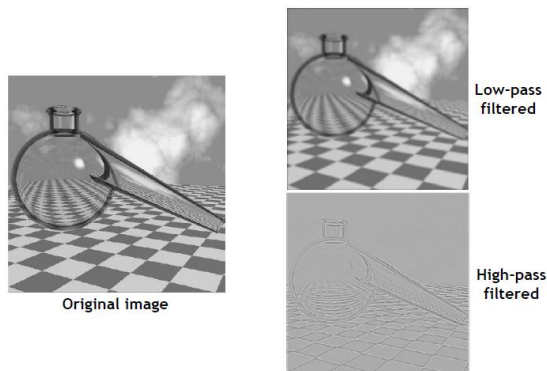
Compact spatial box filter transfer to a frequency sinc, creates artifacts

Gaussian has compact support in both domains. A good choice for a low-pass filter

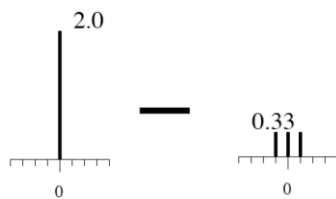


前列是 spatial， 后列是 frequency domain

- low frequencies in images mean pixel values that are changing slowly over space, while high frequency content means pixel values that are rapidly changing in space

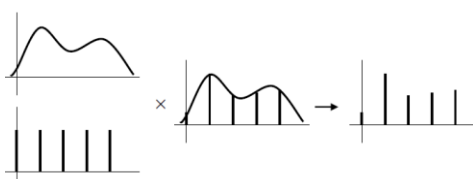


- sharpening filter

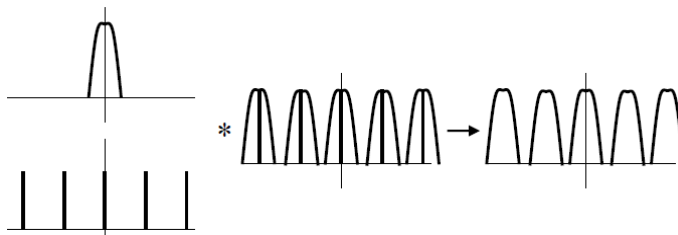


合起来就是中间 1 竖， 旁边两负， 类似于 laplacian filter

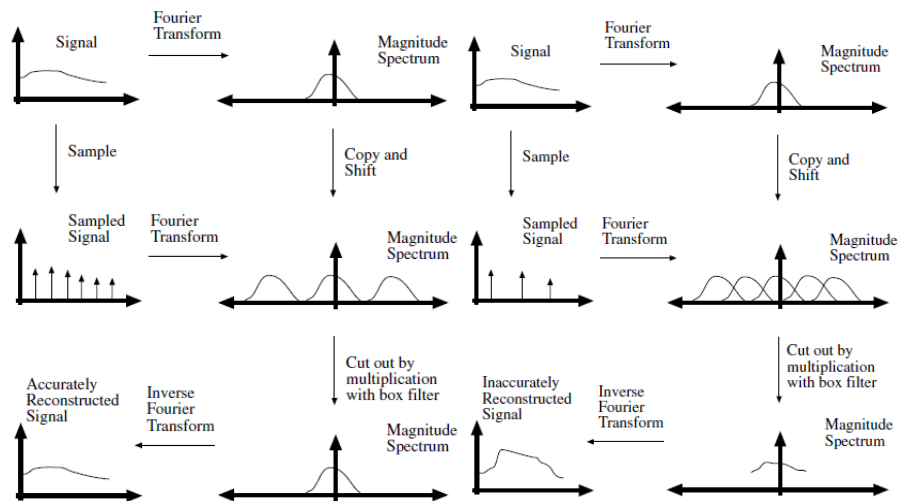
- high-pass, low-pass; band-pass 就是中间一条过， 用 low-pass 和 high-pass band-reject 就是中间一条不过， 用 low-pass 和 high-pass 的并联
- high frequency emphasis:
先 high-pass filtered, 然后加上原图, 然后 histogram equalization
- median filter(这个是 non-linear, 之前的都是 linear 的 filter):
replace each pixel by median of its neighbors
property:
remove spikes, good for impulse, salt&pepper noise
edge-preserving(而 mean 就是会使 edge 消失)
- sampling in spatial domain is like **multiplying** with a spike function



sampling in frequency domain is like **convolving** with a spike function



- sampling and aliasing(混淆)



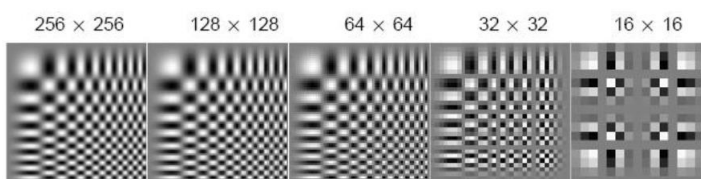
(第一个成功复原，第二个没有)

Nyquist theorem:

In order to recover a certain frequency f , we need to sample with at least $2f$

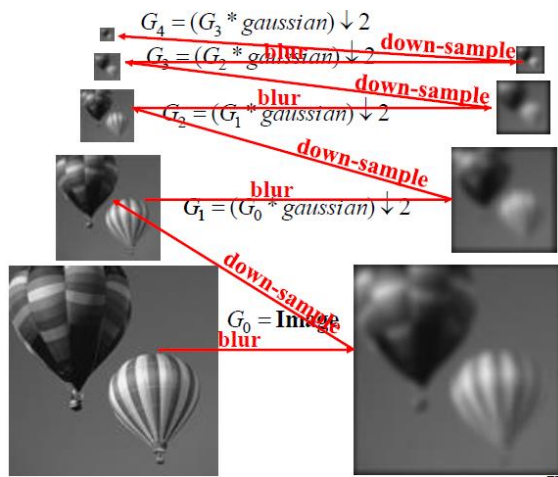
(这就是开始 overlap 的界限)

- aliasing: interference between copies of the original Fourier transform means that we cannot recover its value at some points
- 直接多次 resampling, 每次精度都下降, 最后结果不忍直视



We cannot recover the high frequencies, but we can avoid artifacts by smoothing before resampling

- Gaussian pyramid



就是先 blur(smooth)然后 down-sample

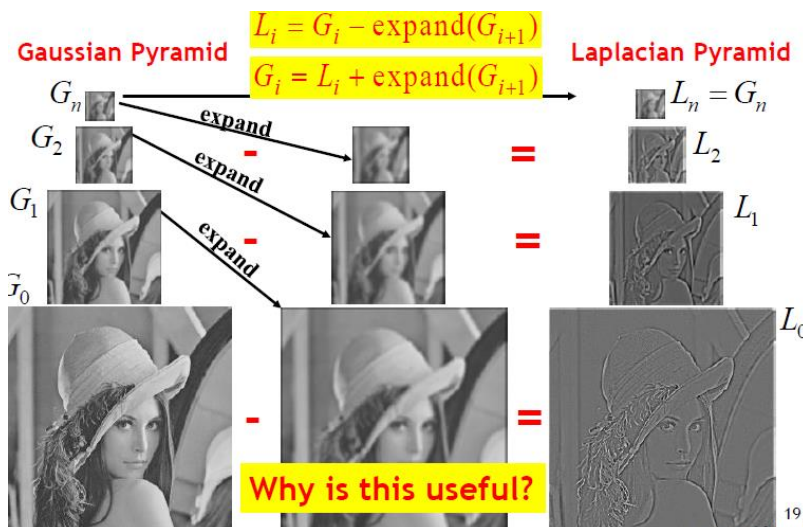
多出来这些层并不会显著增加 store and computation, 精度比原图低, 个数不多, 大小不大

用 gaussian 因为:

1. A Gaussian * Gaussian = another gaussian

2. $G(\sigma_1) * G(\sigma_2) = G(\sqrt{\sigma_1^2 + \sigma_2^2})$

- Laplacian pyramid:



最高层 L_n 和 G_n 一样, 之后都是上一层 gaussian expand 之后和同层 gaussian 做差

- Filters look like the effects they intended to find
- Correlation(filters) can be seen as dot product between image and template

$$a \cdot b = |a| |b| \cos \theta \quad \cos \theta = \frac{a \cdot b}{|a| |b|}$$

点乘完看角度, 就知道哪里最像

- Summary:

Smoothing: sum to 1; low-pass filter

Linear filtering: examples(box filter, Gaussian, finding a derivative, searching for a template)

Part4edge-detection

- Partial derivatives of an image:

X: [-1,1]

Y:

-1
1

- Gradient direction:

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

Gradient magnitude:

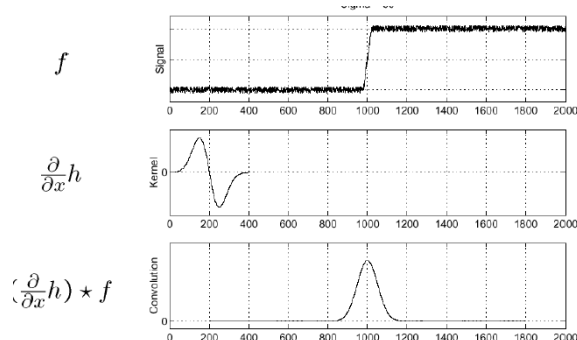
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- 当有大量随机 noise 时，noise 的跳变使我们检测不到真正 edge 的跳变，所以又以下几个方法

- Gaussian:

先 gaussian 来 smooth，之后再求 1 阶 derivative，看峰值即可

- 先把 gaussian(就是 h)给 derivative 了，然后再和 f 作用，这样更简便

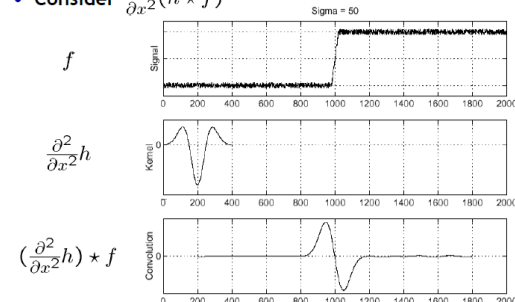


Derivative of gaussian 经常被近似了，比如 sobel，或者下面这种

$$\begin{bmatrix} 1 & -1 \end{bmatrix} * \begin{bmatrix} 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0219 & 0.0983 & 0.1621 & 0.0983 & 0.0219 \\ 0.0133 & 0.0596 & 0.0983 & 0.0596 & 0.0133 \\ 0.0030 & 0.0133 & 0.0219 & 0.0133 & 0.0030 \end{bmatrix}$$

- Laplacian of Gaussian(is 2nd derivative of Gaussian; 不过通常用 difference of Gaussian 近似)注意下图就是找 zero-crossing 了

- Consider $\frac{\partial^2}{\partial x^2}(h * f)$



- Difference of Gaussian 就是两个 Gaussian 分别作用图像，然后求差，也就是 laplacian pyramid 里的一层，这是个 band-pass filter，增强了 edge
- Edge detection steps:
 - 1) Smoothing(决定 edge 精细度): suppress noise
 - 2) Edge enhancement: filter for contrast
 - 3) Edge localization:
 - a) Determine which local maxima are edges
 - b) Thresholding, thinning
- Canny edge detector:
 - a) Filter image with derivative of Gaussian
 - b) Find magnitude and orientation of gradient
 - c) Non-maximum suppression
Check if pixel is local maximum along gradient direction, select single max across width of the edge(require linear interpolation based on gradient direction)
 - d) Linking and thresholding(hysteresis):
Define 2 thresholds: low and high
Use high to start edge curves and the low to continue them. Typically, $k_{high}/k_{low}=2$

Part5structure-extraction

- Fitting as template matching:
- Edge template:

$$D_{\text{corr}}(x, y) = - \sum_{u, v} T[u, v] I[x + u, y + v]$$

像之前 template T 和 image I 来个 correlation
感觉就是 T 里有 1，按着 T 的轮廓求一遍和
但是 1pixel 歪了，匹配就失败了

- 改进用 chamfer distance，式子变成如下

$$D_{\text{Chamfer}}(x, y) = \frac{1}{|T|} \sum_{u, v: T[u, v]=1} d_t(x + u, y + v)$$

dt()这个函数就是求这个点到最近 edge 点的距离

- 所以需要算整个图片各点到 edge 点距离的算法
1 维:

1. Initialize: For all j

➤ $D[j] \leftarrow 1_p[j]$ // 0 if j is in P, infinity otherwise

2. Forward: For j from 1 up to n-1

➤ $D[j] \leftarrow \min(D[j], D[j-1]+1)$

+1 0

3. Backward: For j from n-2 down to 0

➤ $D[j] \leftarrow \min(D[j], D[j+1]+1)$

0 +1

很简单，是 edge 设 0，不是设 infinity，然后左一编，右一遍设置
2 维：类似，不过改成如下 2*2 的看

-	1
1	0

0	1
1	-

从左到右，从上到下，只看上，左
从右到左，从下到上，只看下，右

- 实现：
不再用 correlation，而是 sample 一圈边框，look up 一下 image of distance 即可

$$D_{\text{chamfer}}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t)$$

- 这个方法被用于路标和 pedestrians 的识别

Pros:

1. Fast and simple

2. Works well for matching upright shapes with little intra-class variation
3. Good method for finding candidate matches

Cons:

1. Chamfer averages over contour, not discriminative, so need further verification
 2. Low matching cost in cluttered regions, false positives
 3. In order to match rotated&rescaled, can be expensive
- Fitting as parametric search: Hough transform
 - Polar representation for lines: because usual (m,b) parameter space can take on infinite values, undefined for vertical lines

Point in image space 对应 sinusoid segment in Hough space

$$x \cos \theta + y \sin \theta = d$$

d: perpendicular distance from line to origin

theta: angle the perpendicular makes with x-axis

- Basic hough transform algorithm:
 1. Initialize $H[d, \theta] = 0$
 2. For each edge point (x,y) in the image
 - For $\theta = 0$ to 180

$$d = x \cos \theta + y \sin \theta$$

$$H[d, \theta] += 1$$

3. Find the values of (d,theta) where $H[d, \theta]$ is maximal

4. The detected line in the image is given by $d = x \cos \theta + y \sin \theta$

- 之前算法的问题在于，即使都是 noise 点，也能投出 peaks，所以要减少投票如下

Extension:

1. Same
2. For each edge point $I[x, y]$ in the image
 - Theta = gradient at (x,y)
 - $d = x \cos \theta - y \sin \theta$
 - $H[d, \theta] += 1$
3. Same
4. Same

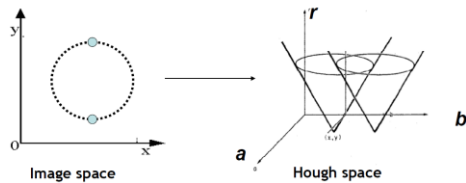
Extension2:

Give more votes for stronger edge(use magnitude of gradient)

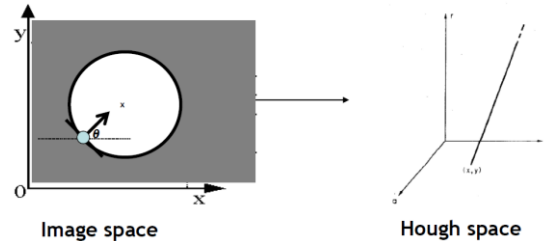
- Hough transform for circles

$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Unknown radius r, unknown gradient



For an unknown radius r , known gradient



- Hough transform for algorithm:

For every edge pixel (x, y) :

For each possible radius value r :

For each possible gradient direction θ :

// or use estimated gradient

$$a = x - r \cos(\theta)$$

$$b = y + r \sin(\theta)$$

$$H[a, b, r] += 1$$

end

end

讲道理我觉得用 each a 或者 b 也可以，但是应该是空间比角度大

- Pros:

1. Can cope with occlusion
2. Robustness to noise
3. Can detect multiple instances of a model in a single pass

Cons:

1. Complexity grow exponentially with parameters
2. Non-target shapes can produce spurious peaks in parameter space
3. Quantization: hard to pick good grid size

- Practical tips:

1. Minimize irrelevant tokens(挑 edge points 看 gradient magnitude)
2. Choose good grid
3. Vote for neighbors
4. Utilize gradient to reduce 1 dimension
5. Keep tags on votes to read back which points voted for "winning" peaks

- Generalized Hough transform(detect arbitrary shapes defined by boundary points and a reference point a):

At each boundary point,

Compute displacement vector: $r = a - p_i$

Store these vectors in a table indexed by gradient θ

(下面的算法 assuming orientation and scale are fixed,所以只有位移)

Algorithm:

For each edge point

Index into table with its gradient orientation θ

Use retrieved r vectors to vote for position of reference point

Peak in the hough space is reference point

Part6segmentation

- K-Means Clustering:

Algorithm:

1. Randomly initialize the cluster center, c_1, \dots, c_k
2. Given cluster centers, determine points in each cluster
-For each point p , find the closest c_i , put p into cluster i
3. Given points in each cluster, solve for c_i
-Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat from step 2

特别的，我们优化的是下式，所以 step3 可以设置为最优解 mean.如果是别的式子，还是得考量怎么最小

$$\sum_{\text{clusters } i} \sum_{\text{points } p \text{ in cluster } i} \|p - c_i\|^2$$

Property:

- a) Always converge
- b) Can be local minimum
- c) optimal solution to the k-means problem is NP-hard

- K-Means++

1. Randomly choose first center
2. Pick new center with probability proportional to $\|p - c_i\|^2$
(this is the square of distance to nearest selected centers)
3. Repeat until k centers

Then go on as K-mean

解释，就是个 k centers 的初始化算法，就是挑离谁都特远的当，以此避免特别差的 local minimum

Expected error = $O(\log k) * \text{optimal}$

- 之前的是只通过 intensity 来 clustering。为了减少 noises 跟错了 cluster，可以 grouping based on intensity + position 这样 3 维就可以 smooth 一下

- Pros:

1. Simple, fast
2. Converges to local minimum

Cons:

1. Setting k ?
2. Sensitive to initial centers
3. Sensitive to outliers
4. Detects spherical clusters only(那种外圈环抱内圈就分不开)
5. Assuming means can be computed

- Probabilistic clustering

Expectation maximization(EM) algorithm:

Generative model is a mixture of Gaussians as follows:

$$p(\mathbf{x}|\theta_j) = \frac{1}{(2\pi)^{D/2}|\Sigma_j|^{1/2}} \exp \left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_j)^T \Sigma_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j) \right\}$$

K Gaussians blob with mean, covariance, and dimension D

Blob j is selected with probability π_j , sum of $\pi_j = 1$

虽然算法是各自优化自己，但是总的目标其实是下面这个式子最大

$$p(\mathbf{x}|\theta) = \sum_{j=1}^K \pi_j p(\mathbf{x}|\theta_j) \quad \theta = (\pi_1, \boldsymbol{\mu}_1, \Sigma_1, \dots, \pi_M, \boldsymbol{\mu}_M, \Sigma_M)$$

E-step: softly assign samples

$$\gamma_j(\mathbf{x}_n) \leftarrow \frac{\pi_j \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_j, \Sigma_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_k, \Sigma_k)} \quad \forall j = 1, \dots, K, \quad n = 1, \dots, N$$

M-step: re-estimate the parameters based on soft assignments

$$\hat{N}_j \leftarrow \sum_{n=1}^N \gamma_j(\mathbf{x}_n) = \text{soft number of samples labeled } j$$

$$\hat{\pi}_j^{\text{new}} \leftarrow \frac{\hat{N}_j}{N}$$

$$\hat{\boldsymbol{\mu}}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^N \gamma_j(\mathbf{x}_n) \mathbf{x}_n$$

$$\hat{\Sigma}_j^{\text{new}} \leftarrow \frac{1}{\hat{N}_j} \sum_{n=1}^N \gamma_j(\mathbf{x}_n) (\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})(\mathbf{x}_n - \hat{\boldsymbol{\mu}}_j^{\text{new}})^T$$

- EM 简而言之就是根据生成模型，估计 sample 分给谁的比重，根据比重和 sample，调整参数，然后 check 有没有收敛，迭代
- EM takes more iterations than K-means, and require more computation. 可以先跑 K-mean，以此来初始化 EM。EM 也是找到 local minima。优点是 useful for all sorts of problems (clustering, model estimation, missing data, finding outliers, segmentation)
- Pros:
 1. Probabilistic interpretation
 2. Soft assignments
 3. Generative model, can predict novel data points
 4. Relatively compact storage

Cons:

1. Local minima
 2. Initialization
 3. Need to know number of components(K): solutions are model selection and Dirichlet process mixture
 4. Need to choose generative model
- Mixture of Gaussian 还有个应用就是用户选定 2 个框当前后背景，计算框内 Gaussian，以此来分类其他 pixels。
 - Mean-shift clustering(model-free)

Algorithm:

1. Initialize random seed, and window W

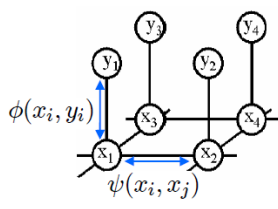
2. Calculate center of gravity of W: $\sum_{x \in W} xH(x)$ (这里例子里 x 是数值, H(x)是个数)
3. Shift search window to the mean
4. Repeat until convergence

实际跑的时候, 把 space 遍布 windows, 然后并行跑, merge 跑到一起的 windows

- Mean shift clustering:
根据 attraction basin(region which all trajectories 弹道 lead to the same mode)划分区域
- Speedups:
Many windows, many computation redundant
 1. Assign all points within radius r of end points to the mode
 2. Assign all points within radius r/c of the search path to the mode(c 应该是个自己设的系数)
- Pros:
General tool
 1. Model-free, does not assume prior shape on data clusters
 2. Single parameter(window size)
 3. Find variable number of modes
 4. Robust to outliers
- Cons:
 1. Output depends on window size(bandwidth)
 2. Window size selection no trivial
 3. Computationally expensive
 4. Not scale well with dimensions of feature space
- 总结:
K-means:先来 k 个 centers
EM: soft assign K 个 generative model
Mean shift:定窗口大小, 填满, 然后移动

Part7segmentation

- Markov random field:



(scene, image)的概率如下

$$p(\mathbf{x}, \mathbf{y}) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j)$$

转变成 minimize energy(two fi are called potentials):

$$E(\mathbf{x}, \mathbf{y}) = \sum_i \phi(x_i, y_i) + \sum_{i,j} \psi(x_i, x_j)$$

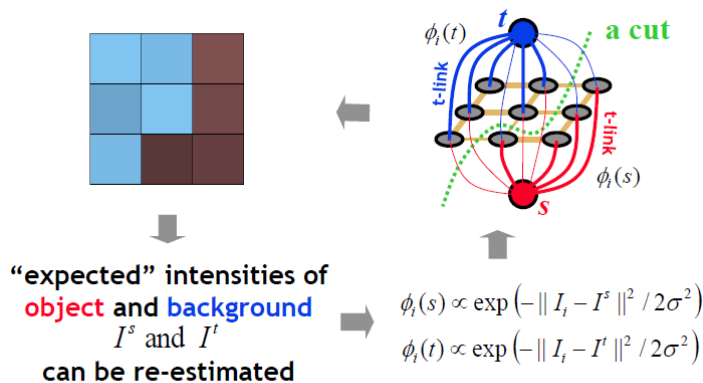
- Graph cuts(inference algorithm for optimal labelling of MRF):

We first convert MRF into source-sink graph

$$E(\mathbf{x}, \mathbf{y}) = \underbrace{\sum_i \phi_i(x_i)}_{\text{Unary terms}} + \underbrace{\sum_{i,j} w_{ij} \cdot \delta(x_i \neq x_j)}_{\text{Pairwise terms}}$$

$$w_{ij} = \exp \left\{ -\frac{\Delta I_{ij}}{2\sigma^2} \right\}$$

上面都是大致示意，实际算法流程如下



EM-style optimization

1. 刚开始 I^s 和 I^t 是给的，比如后面说的是用户划的或者框的
2. 之后按照 energy function 计算 energy
3. 然后按照 s-t cut 看最小 energy 下是怎么划分的
4. 按照此划分，重新估计 I^s, I^t ，迭代

Potential function 可以很随意，比如下面：

$$\phi_i(L_i) = -\log p(I_i|L_i)$$

就是点 i 被分成类 L 的概率的 **negative-log**，这就当作 **energy**
再比如下面 **mixture of Gaussian** 型，这是 **unary potential**

$$\phi(x_i, y_i; \theta_\phi) = \log \sum_k \theta_\phi(x_i, k) p(k|x_i) \mathcal{N}(y_i; \mu_k, \Sigma_k)$$

Edge potential(pairwise potential)

$$\psi(x_i, x_j, g_{ij}(\mathbf{y}); \theta_\psi) = -\theta_\psi g_{ij}(\mathbf{y}) \delta(x_i \neq x_j)$$

where

$$g_{ij}(\mathbf{y}) = e^{-\beta \|y_i - y_j\|^2} \quad \beta = \frac{1}{2} (\text{avg}(\|y_i - y_j\|^2))^{-1}$$

Parameters θ_ϕ, θ_ψ need to be learned, too!

上面这句话就是 **EM** 精髓，之前是 **intensity**，现在 **generalize** 到各种模型的参数
Minimum cost cut can be computed in polynomial time

- **s-t cut:**

只求和 **cut** 线路上 **S to T** 的 **cost**

每个边的数字为 **cost**，切开时算不在一起付出的代价

Maximum flow equals the cost of the st-mincut

Algorithm:

1. Find path from s to t with positive capacity
2. Push maximum possible flow through this path
3. Repeat until no path can be found

(注意， **edge weights** 必须非负)

简而言之就是挑 **s->t** 线路 (要看箭头方向的)，把路上最小的那些设成 0，别的设成和它们的差，这样一点点都变成 0，最后 **cut** 全 0 的那些 (只看 **s->t**)

Maximum flow 就是一路算下来，减去的那些最小 **cost** 的和

$$E(L) = \underbrace{\sum_p E_p(L_p)}_{\text{t-links}} + \underbrace{\sum_{pq \in N} E(L_p, L_q)}_{\text{n-links}} \quad L_p \in \{s, t\}$$

注意上图 **t-links, n-links** 的定义

s-t graph cuts only globally minimize binary energies that are submodular

$E(L)$ can be minimized by s-t graph cuts	\iff	$E(s,s) + E(t,t) \leq E(s,t) + E(t,s)$
Submodularity ("convexity")		

- **α -expansion move(generalize s-t mincut to multi-class):**

Not globally optimal, just approximation;

Idea 就是 **break multi-way cut computation into a sequence of binary s-t cuts**

Algorithm:

1. Start with any initial solution
2. For each label " α " in any order:
 Compute optimal α -expansion move(s-t graph cuts)

注意这里就是当前类 α 从别人那夺取地盘，使 energy 降低

Decline the move if there is no energy decrease

3. Stop when no expansion move would decrease energy

- GrabCut(graph cut 的应用):

User marks foreground and background with a brush or bounding box

Create initial segmentation and then can be corrected with additional brush strokes

$$\psi(x, y) = \gamma \sum_{(m,n) \in C} \delta[x_n \neq x_m] e^{-\beta \|y_m - y_n\|^2}$$

Neighborhood 的建模，但是 γ 的选择还是很重要，并且不知道怎么选

- Pros:

- a) Powerful
- b) Applicable for a wide range of problems
- c) Efficient algorithm available for vision problems

Cons:

- a) Solve only submodular energy functions(waste expressiveness of MRFs)
- b) Only approximate algorithms for multi-label case

Part8categorization

- Object recognition challenges:
 1. Viewpoint changes:
 - Translation
 - Image-plane rotation
 - Scale changes
 2. Illumination
 3. Noise
 4. Clutter(嘈杂背景)
 5. Occlusion
- Detection 的方法: Sliding window(window fixed, search over space and scale of image) + a binary classifier
- Detection via distance to eigenspace, identification via distance in eigenspace
(就是分类器需要提取特征, 检测人脸看和人脸近不近, 判断是谁看和谁近)
- Feature 的选择: color/grayscale 这种对光照变换太敏感
基于 gradient 的比较好
 1. locally orderless: invariant to small shifts and rotations
 2. localized histograms: offer more spatial information than a single global histogram
 3. contrast-normalization: correct for variable illumination(后面 HOG 里就是这一招)
- SVM:
Maximize the margin between positive and negative training examples
Samples on the margin are called support vectors

$$\begin{aligned} &\text{Minimize } \frac{1}{2} \mathbf{w}^T \mathbf{w} \\ &\text{Subject to } t_n (\mathbf{w}^T \mathbf{x}_n + b) \geq 1 \end{aligned}$$

这里 margin 是 1, t_n 是 $\{-1, 1\}$

Solution:

$$\mathbf{w} = \sum_{n=1}^N a_n t_n \mathbf{x}_n$$

Learned weight Support vector

(这里是所有点还是 support vector 的点存疑, 看 prml)

Classification:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T \mathbf{x} + b)$$

$$= \text{sign} \left(\sum_{n=1}^N a_n t_n \boxed{\mathbf{x}_n^T \mathbf{x}} + b \right)$$

Optimization 要 $\mathbf{x}_n^T \mathbf{x}_m$ 对所有训练数据对

Classification 则要计算 test 值 \mathbf{x} 和所有 \mathbf{x}_n 的点乘

- Property of SVM:

1. Can generalize to d-dimensions
2. Non-linear SVMs map to high dimensional feature space by special kernels, and can deal with data not linearly separable.

Kernel trick: 实际算的时候, 不用算升维 ϕ , 直接算 kernel 就行, classification 时也直接用

$$K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$$

$$\sum_n a_n t_n K(\mathbf{x}_n, \mathbf{x}) + b$$

Often used kernels:

Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$

Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$

Gaussian:

3. Multi-class SVM

用多个 binary 分类器来构成

One vs all (每一个学一个分类器, 用时都用, 看最自信的那个) or one vs one (每一对都学一个分类器, 用的时候都用来投票)

- SVM algorithm:

1. Define your representation for each example
2. Select a kernel function
3. Compute pairwise kernel values between labeled examples
4. Pass this "kernel matrix" to SVM optimization software to identify support vectors & weights
5. To classify a new example: compute kernel values between new input and support vectors, apply weights, check sign of output

- Histogram of Oriented Gradients(HOG):

1. Gamma compression: reduce overly strong gradients

Each pixel $x \mapsto \sqrt{x}$

2. Gradient computation:

Compute gradient on all color channels, take strongest one

用这俩 filter 即可，不需要在 filter 前 gaussian smooth

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

3. Weighted vote in Spatial/ orientation cells

一般是 8*8 pixel 组成的 cells，投 9 个方向

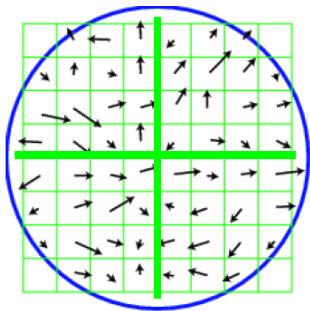
通过如下计算方向，并且根据 magnitude 有 vote 权重

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

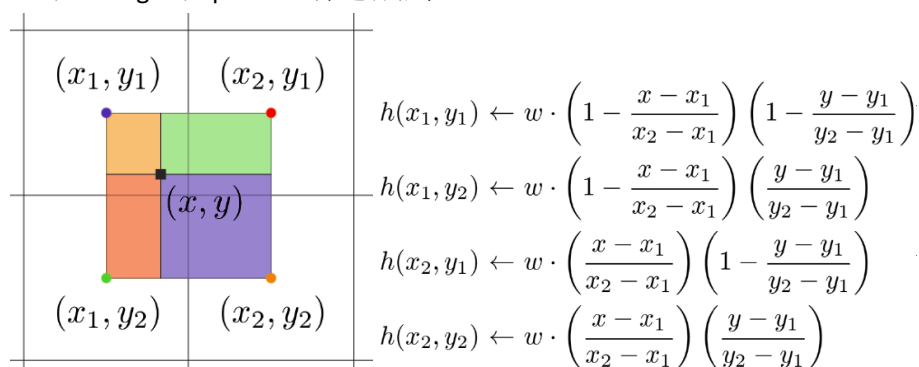
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x} \right)^2 + \left(\frac{\partial f}{\partial y} \right)^2}$$

4. Contrast normalize over overlapping spatial cells: L2-norm followed by clipping (limiting the maximum values of v to 0.2) and renormalizing 先变成 unit length, 把 0.2 以上的分量 clipped 了，然后再化成 unit length

此外，每个 cell 在 4 个 block 里，每个 block 用自己的 Gaussian 还给一个权重



此外，angle 和 position 都进行插值



一个 pixel 对 4 个 cell 有作用，看 cell 中心和 pixel 距离

角度对最多 2 个邻接的 orientation 投票

5. Collect HOGs over detection windows:

如下所示，一个 cell 在 4 个 block 中，所以 contribute 4 次，或者说 descriptor 是 4 个不同 normalization 的 concatenation

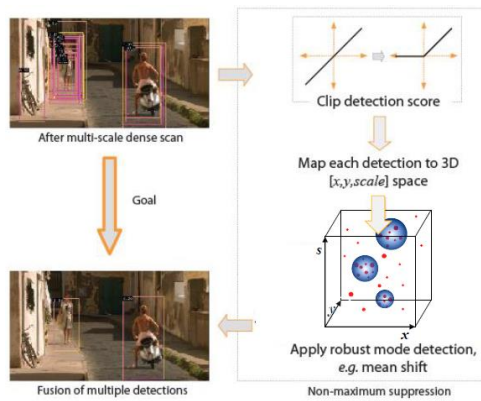
$$\# \text{ features} = 15 \times 7 \times 9 \times 4 = 3780$$

orientations

cells # normalizations by neighboring cells

6. Linear SVM

- 实际 sliding windows 操作，会有同一物体，多 scale 响应



如上图，从右侧走一圈，先去掉分不高的，然后投射到特征空间，然后 mean shift

Part9categorization

- Boosting: build strong classifier H by combining a number of weak classifiers
- AdaBoost:

1. Initialization: Set $w_n^{(1)} = \frac{1}{N}$ for $n = 1, \dots, N$.



2. For $m = 1, \dots, M$ iterations

a) Train a new weak classifier $h_m(x)$ using the current weighting coefficients $\mathbf{W}^{(m)}$ by minimizing the weighted error function

$$J_m = \sum_{n=1}^N w_n^{(m)} I(h_m(x_n) \neq t_n) \quad I(A) = \begin{cases} 1, & \text{if } A \text{ is true} \\ 0, & \text{else} \end{cases}$$

b) Estimate the weighted error of this classifier on \mathbf{X} :

$$\epsilon_m = \frac{\sum_{n=1}^N w_n^{(m)} I(h_m(x_n) \neq t_n)}{\sum_{n=1}^N w_n^{(m)}}$$

c) Calculate a weighting coefficient for $h_m(x)$:

$$\alpha_m = \ln \left\{ \frac{1 - \epsilon_m}{\epsilon_m} \right\}$$

d) Update the weighting coefficients:

$$w_n^{(m+1)} = w_n^{(m)} \exp \{ \alpha_m I(h_m(x_n) \neq t_n) \}$$

这里 m 既是第几轮，也是第几个 weak classifier

式子 d 就是分类对了，sample 的 w 不变，否则乘上 $(1-\epsilon)/\epsilon$ ，也就是 weak classifier 越对，但还是分错我了，我就乘以越大

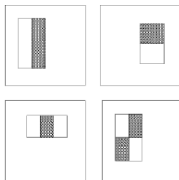
式子 a 是 Error 的计算，但是从它怎么得到 weak classifier 就因问题而异了

在最后的 test 中，大家一起上。分类结果*自己的正确率（权重）

$$H(\mathbf{x}) = \text{sign} \left(\sum_{m=1}^M \alpha_m h_m(\mathbf{x}) \right)$$

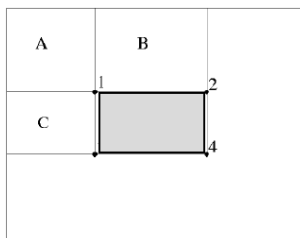
- Viola-Jones face detector

Filters(获取 feature):



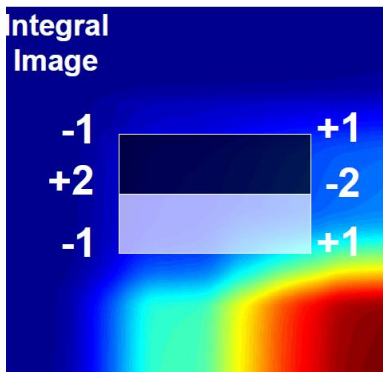
白色区域面积-黑色区域面积

Integral image(上面 filter 可以用此快速计算):



点(x,y)存储从左上角到自己所有像素的 sum

灰色区域=1+4-2-3



上面这种常见 feature filter 就可以通过 6 个数飞快计算（白色减黑色复合一下）
 这里 weak classifier 由单个 feature 构成, 所以 adaboost 优化 weak classifier 也挑选 feature
 Adaboost:

For each round of boosting

1. Evaluate each rectangle filter on each example
2. Sort examples by filter values(就是 filter 放 24×24 输入上出一个值, 输入训练集都是 rescaled 到 24×24 的)
3. Select best threshold for each filter

(根据 error 的式子 $e = \min (S^+ + (T^- - S^-), S^- + (T^+ - S^+))$, 挑个最小的)

各系数意思 sum of positive example weights T^+ , the total sum of negative example weights T^- , the sum of positive weights below the current example S^+ and the sum of negative weights below the current example S^-

理解: the minimum of the error of labeling all examples below the current example negative and labeling the examples above positive versus the error of the converse.

就是把当前下面都标成-, 会有多少错, 标成+, 会有多少错。Classifier 不关心正负, 只关心分类错误

4. Select best filter/ threshold combo

Classifier 就新鲜出炉: θ_t 是 shreshold

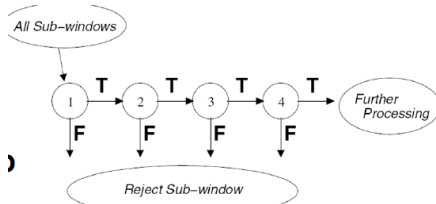
$$h_t(x) = \begin{cases} +1 & \text{if } f_t(x) > \theta_t \\ -1 & \text{otherwise} \end{cases}$$

5. Weight on this features is a simple function of error rate
6. Reweight examples

下面是论文里面除了 integral image 和 adaboost, 第三个工作

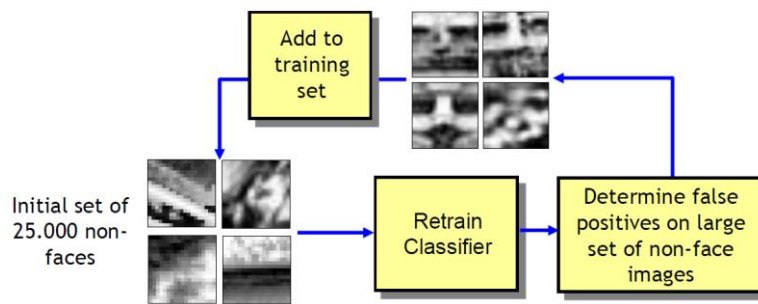
Cascading classifiers for detection:

本身 adaboost 然后来个 200feature 的复合分类器, 计算非常慢



Build a chain of classifiers, choosing cheap ones with low false negative rates early in the chain, then progressively more complex and lower false positive rates

实际上，用了 38 层，最后一层由 6061 features 组成 classifier
Bootstrapping(挑 negative 数据集):



其实就是把 false positive 加训练集上去

- Sliding-windows

Pros:

1. Simple to implement
2. Good detectors available(Viola-Jones, HOG)

Cons:

1. High computational complexity
2. False positive must be really low

- SVM detectors

1. Same computations for each image window
2. Pays off to precompute the features once

AdaBoost cascade detectors

1. Potentially different computations for each window location
2. Maybe more efficient to evaluate the features on-the-fly for each image window

- HOG is dominated by computations of block histogram

- Sliding windows limitations:

1. 经常是 24×24 这种低 resolution 的，高的算不起
2. 没法应对不同长宽比(aspect ratio)
3. Not good for objects with less-regular textures
4. Context is lost
5. Need large cropped training set
6. Sensitive to partial occlusions

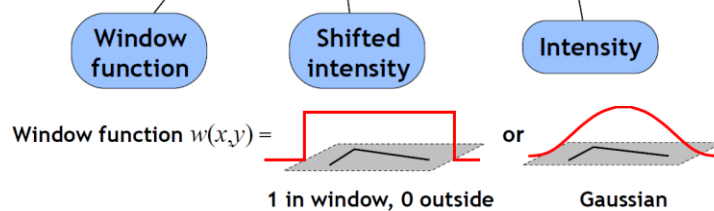
Part10local-features

- 上一章末尾我们看到 global 的特征表示有很多局限性，这一章就是 local 匹配
- 特征点匹配的步骤:
 1. Find a set of distinctive key-points
 2. Define a region around each key-point
 3. Extract and normalize the region content
 4. Compute a local descriptor from the normalized region
 5. Match local descriptors
- 特征点要求:
 1. Invariant to translation, rotation, scale changes, robust to lighting variations, noise, blur
 2. Locality(局部可算)
 3. Quantity 数量多
 4. Distinctiveness 不一样，可以匹配
 5. Efficiency 好算

● Harris detector

注意到角在一个 window 下各方向移动都会明显改变 intensity

$$E(u, v) = \sum_{x, y} w(x, y) [I(x+u, y+v) - I(x, y)]^2$$



左侧就是把点选出来，右侧有 gaussian smoothing 的效果，harris 和 hessian 都用它

$$M = \sum_{x, y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Gradient with respect to x , times gradient with respect to y

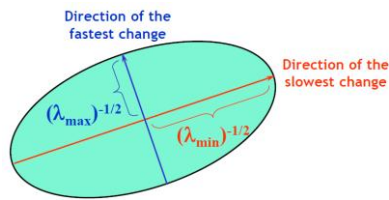
Sum over image region - the area we are checking for corner

实际 E 都不用，就求上面个这个 M, second moment matrix，注意是 1 阶导的乘积

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

M 是对称矩阵，可以特征值分解成上面这种，注意分解后右边是竖列的 eigenvector，左边是逆

M 可以看成如下椭圆



(eigenvector of larger eigenvalue points in direction of fastest intensity change)

我们要找的角就是两个特征值都大的，实际应用时不特征值分解，而是如下：

Conner response function:

$$R = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2 = \det(M) - \alpha \text{trace}(M)^2$$

直接算 \det 和 trace 就好， α 是经验设置的 0.04~0.06

外面加圈 Gaussian 不仅能 weighted sum，而且还 rotation invariant

$$M = g(\sigma) * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

总结下流程：

1. Image derivative(1 阶导)
2. Square of derivatives(乘起来)
3. Gaussian filter
4. Conner response function:

$$\begin{aligned} R &= \det[M(\sigma_I, \sigma_D)] - \alpha [\text{trace}(M(\sigma_I, \sigma_D))]^2 \\ &= g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha [g(I_x^2) + g(I_y^2)]^2 \quad (\text{trace 是任意方阵主对角线的和}) \end{aligned}$$

5. Non-maximum suppression

- Harris is rotation invariant, but not scale invariant
- Hessian detector

$$\text{Hessian}(I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

$$\det(\text{Hessian}(I)) = I_{xx} I_{yy} - I_{xy}^2$$

Hessian 求二阶导，然后看 \det 就成了。Search for strong derivatives in 2 **orthogonal** directions

- 2nd derivative:

$$\mathbf{a} = \frac{d\mathbf{v}}{dt} = \frac{d^2 \mathbf{x}}{dt^2}$$

- Harris locations are more **specific** to corners, while the Hessian detector also returns many responses on regions with **strong texture variation**. In addition, Harris points are typically more **precisely located** as a result of using first derivatives rather than second derivatives and of taking into account a larger image neighborhood

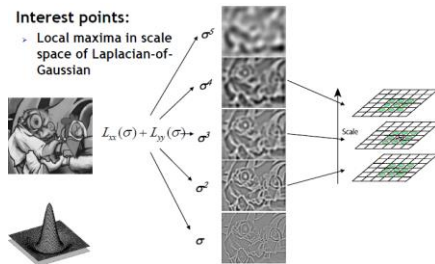
Thus, Harris points are preferable when looking for exact corners or when precise localization is required, whereas Hessian points can provide additional locations of interest that result in a denser coverage of the object

- Scale invariant region selection

Find a signature function(function of region size) which is “scale invariant” and select local maximum of it to rescale and match

- 唯一的 signature function is Laplacian-of-Gaussian

Characteristic scale is the scale who produces peak of Laplacian response



Sigma 是不同的 scale，构成 3d，(x,y,sigma)的 scale space，找里面的 local maxima
Laplacian-of-Gaussian 可以用来选 scale，也可以同时选位置 x,y，也就是可以单独使用
实际计算中都是用 DoG

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma) \right)$$

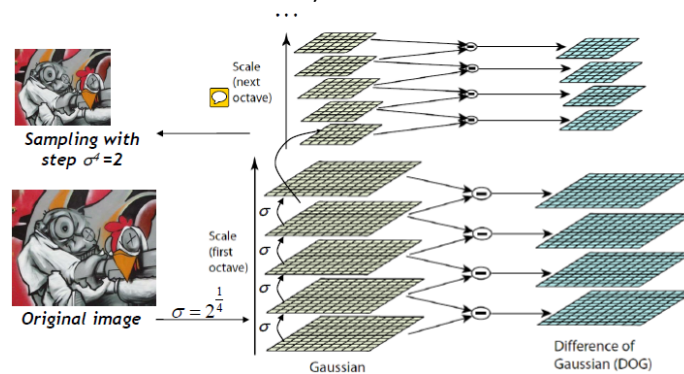
(Laplacian)

$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

(Difference of Gaussians)

本来就比二阶导快，而且 Gaussian pyramid 本来就有算

1. Detect maxima of DoG in scale space
2. Reject points with low contrast(一个下限 threshold)
3. Eliminate edge responses(LoG 和 DoG 对 edge 有反应，去掉 ratio between principal curvatures > threshold)



上图就是 DoG 怎么算的，就是每层用 sigma 的高斯来 smooth，smooth4 次(例子而已)，就完全模糊了二分之一，可以 sampling 减小 resolution 而没影响，继续。

- Harris-Laplace
 - a) Multiscale Harris corner detection
 - b) Scale selection based on Laplacian

Hessian-Laplace 同

Part11local-features

- Orientation normalization:
 1. Compute orientation histogram(pixel 的 gradient 投票, 权重自己设计, 什么加 block 的 gaussian 和 gradient 的 magnitude)
 2. Select dominant orientation(多个一样的时候, 就看 shreshold, 然后每个角度都加一份)
 3. Rotate to fixed orientation
- Affine invariant feature extraction
- Affine adaptation
 - a) Use a circular window to compute second moment matrix
 - b) Compute eigenvectors to adapt the circle to an ellipse
 - c) Recompute second moment matrix using new window and iterate(一个圆, 变成椭圆, 然后一点一点挪动, 椭圆变了又变)
- Affine normalization:
 - a) Rotate the ellipse's main axis to horizontal(这里的 rotate 并不是旋转矫正, 只是变圆方便)
 - b) Scale the x axis, such that it forms a circle
- 总步骤
 - a) Extract regions(选点和选 scale, 这是之前 Harris, Hessian, LoG, DoG 做的, 然后 affine adaptation)
 - b) Normalize regions(就是 affine normalization)
 - c) Rotation
 - d) Compare descriptors
- 这里是 affine covariant, 即 $\text{features}(\text{transform}(\text{image})) = \text{transform}(\text{features}(\text{image}))$
- HOG, SIFT, GIST:

本质上这 3 一样, 但是本课程应用时

SIFT is typically used for **matching local regions** (ones that have been chosen by an interest point detector like the DoG detector in Lowe's SIFT paper, which extracts x,y location, scale and orientation for each point) in two images for purposes of alignment / reconstruction / structure from motion, though it has been used for recognition as well (e.g. in a Bag-of-features fashion).

HOG is typically used in a **sliding window** fashion in object detection systems (e.g. pedestrian detection).

GIST is typically computed over the **entire image** (i.e. it is a global image descriptor) for the purposes of scene classification.
- Scale Invariant Feature Transform(SIFT):
 - a) Divide patch into 4×4 sub-patches: 16 cells
 - b) Compute histogram of gradient orientations(8 angles) for all pixels inside each sub-patch
 - c) Descriptor: $4 \times 4 \times 8 = 128$ dimensions
- Pros:
 - a) Robust to rotation of viewpoint, illumination changes

b) Fast and efficient

单个 patch 参数

1. Position (x,y)
2. Scale parameters (1)
3. Orientation(1)
4. 128-dimensional descriptor

- SURF

Fast approximation of SIFT using 2D box filters("Haar wavelets") & integral images

- Warping: given image and a transformation, see output

Alignment: given 2 images with corresponding features, find what is the transformation

- 2*2 matrix (only linear 2D transformation possible) :

Scaling:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Rotation:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Shearing:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & sh_x \\ sh_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Mirror:
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$
 (这是(0,0), 还可以别的, 需要稍微变一下)

But cannot translation!

- Homogeneous coordinates

3*3 matrix

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation

别的同理, 就是底下加一行[0,0,1], 输入加一个 1

- Affine transformations = linear transformations + translations

(parallel lines remain parallel)

- Projective transformation = affine transformation + projective warps

(may not parallel anymore)

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

- If exactly 3 points correspondences, $b=A^{-1}X$;

If data noise, then use more than 3 points, then do pseudo-inverse, this minimize squares error

$$b = (A^T A)^{-1} A^T X_B = A^\dagger X_B$$

Homography 里面 4 个的同理

Affine transformation 解下式

$$\begin{bmatrix} x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} x'_i \\ y'_i \\ \dots \end{bmatrix}$$

- A projective transform is a mapping between any two perspective projections with the same center of projection: not parallel lines, but still straight lines

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}_{p'} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}_p$$

Set scale factor to 1
⇒ 8 parameters left.

- 求解 Homography matrix

$$\mathbf{x}_B = \frac{1}{z'_B} \mathbf{x}'_B \quad \text{with} \quad \mathbf{x}'_B = \mathbf{H} \mathbf{x}_A$$

$$\begin{bmatrix} x_B \\ y_B \\ 1 \end{bmatrix} = \frac{1}{z'_B} \begin{bmatrix} x'_B \\ y'_B \\ z'_B \end{bmatrix} \quad \begin{bmatrix} x'_B \\ y'_B \\ z'_B \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_A \\ y_A \\ 1 \end{bmatrix}$$

$\mathbf{x}_A, \mathbf{x}_B$ 分别是原图，新图的坐标，第三项都 rescale 是 1

$$x_B = (h_{11}x_A + h_{12}y_A + h_{13}) / (h_{31}x_A + h_{32}y_A + 1)$$

就是矩阵 H 乘上 \mathbf{x}_A 再除以 z'_B 的结果

可以转化为=0 的式子如下

$$x_B \cdot h_{31} \cdot x_A + x_B \cdot h_{32} \cdot y_A + x_B - h_{11} \cdot x_A - h_{12} \cdot y_A - h_{13} = 0$$

同理 y_B 也有这么一个式子

$$\begin{bmatrix} x_{B1} & y_{B1} & 1 & 0 & 0 & 0 & -x_{A1}x_{B1} & -x_{A1}y_{B1} & -x_{A1} \\ 0 & 0 & 0 & x_{B1} & y_{B1} & 1 & -y_{A1}x_{B1} & -y_{A1}y_{B1} & -y_{A1} \\ & & & & & & \dots & & \\ & & & & & & \dots & & \\ & & & & & & \dots & & \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$\mathbf{A}\mathbf{h} = \mathbf{0}$

把 A 给 SVD 成 $\mathbf{U}\mathbf{D}\mathbf{V}^T$

因为是 A 右乘 h，所以 h 是 V 的最后一列（不要转置），也就是特征值 0 对应的特征向量

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T = \mathbf{U} \begin{bmatrix} d_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & d_{99} \end{bmatrix} \begin{bmatrix} v_{11} & \dots & v_{19} \\ \vdots & \ddots & \vdots \\ v_{91} & \dots & v_{99} \end{bmatrix}$$

（这里右上角少了一个转置号）

$$\mathbf{h} = \frac{[v_{19}, \dots, v_{99}]}{v_{99}}$$

Part12local-features

- Least-squares fitting sensitive to outliers

- Random Sample Consensus(RANSAC):

The set of inliers obtained for the fitting model is called **consensus set**

1. Sample a minimal subset of m correspondences
2. Estimate model from these m correspondences
3. Verify the model with others and calculate inliers(base on a threshold)
4. If good, store the model
5. Repeat until termination criterion is met(比如轮数足够多, 或者 inliers 够多)

上面其实就是一个一个 model 去试, 因为初始是 minimal 的集合, 所以最优解很可能 outlier 比较少

多少轮:

w is inliers 比例; n 是 define model 最少的 correspondence 数量, k 是轮数

All k samples fail at rate $(1-w)^k$

所以轮数 k 足够大就成了(这里单个 model 如果选的全都是 inliers 即 w^n , 就看做胜利)

在 RANSAC 粗糙的最好 model 后面优化:

Over all inliers(of the model), least-squares minimization 来优化 model, 之后 inliers/outliers

判定会发生变化, 所以 iteratively, 直到不错

- Generalized Hough Transform:

A single feature match estimate a model of transform, vote in Hough space

Cons:

noise/clutter lead to many votes

Bin size need to be chosen carefully

实际应用: broad bins and spread votes to nearby bins

Part13local-features

- inverted index: is an index data structure storing a mapping **from content**, such as words or numbers, **to its locations** in a database file, or in a document or a set of documents (named in contrast to a Forward Index, which maps from **documents to content**).
- Visual words:
Extract some local features from a number of images
Clustering (e.g. K-means), let centers be the words
A novel image's features can be translated into words (Euclidean distance)
- 上面 clustering 实际经常是 uses hierarchical k-means to recursively subdivide the feature space, assigning new image features to words - from linear to log of size of vocabulary, so vocabulary can be larger
但是太大也不行, the scores decrease with increasing database size as there are more images to confuse with
Branch factor 大会小幅提升效果

- Term frequency – inverse document frequency(tf-idf) weighting:
Describe frame by frequency of words, weight by below

$$t_i = \frac{n_{id}}{n_d} \log \frac{N}{n_i}$$

Number of occurrences of word i in document d → n_{id}

Number of words in document d → n_d

Total number of documents in database → N

Number of occurrences of word i in whole database → n_i

- Application(找有这个领带的场景):
 - a) Collect all words within query region
 - b) Inverted file index to find relevant frames
 - c) Compare word counts(到这都是常规匹配)
 - d) Spatial verification(这是加入位置关系, 比如 spatial pyramid bag-of-words)
- Bags of visual words:
Descriptors-> words-> 1D histogram of word occurrence(vector)

看俩像不像用啥 histogram 比较方法都行, 比如

$$\text{sim}(\vec{d}_j, \vec{q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| \times |\vec{q}|}$$

可以用 BoW 来训练 SVM 得到分类器

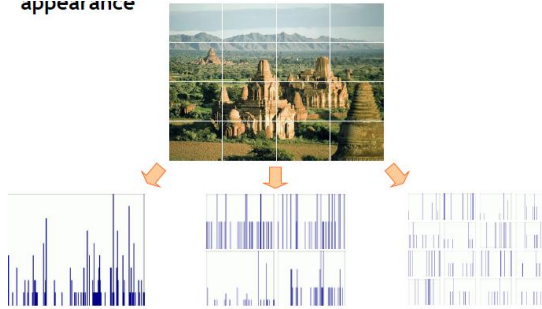
BoW removes spatial layout- both strength and weakness

为了加入一定 spatial information 可以:

- a) Visual phrases
- b) Let position be part of feature
- c) After matching, verify neighbors are the same

Spatial pyramid representation

appearance



BoW 的优缺点

Pros:

- a) Flexible to geometry/ deformation/viewpoint
- b) Provides vector representation
- c) Compact summary of image

Cons:

- a) Ignore geometry and have to be verified
- b) Optimal vocabulary formation remains unclear

Part14categorization

- Implicit shape model

Learn an appearance codebook

Learn a star-topology structural model(features are independent, link to ogj.center)

1. Learn appearance codebook

就是提取 local features, 之后再 cluster, 提取成 word

2. Learn spatial distributions(就是上面说的 structural model)

Match codebook to training image

Record matching positions on object(一个指向中心的向量)

在基础上可以拓展到 scale invariant, 就是加一个 scale 维, 另外俩减一下倍数

$$x_{vote} = x_{img} - x_{occ}(s_{img}/s_{occ})$$

$$y_{vote} = y_{img} - y_{occ}(s_{img}/s_{occ})$$

$$s_{vote} = (s_{img}/s_{occ}).$$

最后一个是 scale, xocc 是 codebook 中的关系长度, 乘上倍数, 和当前点 ximg 一做差, 就是 obj 中心

在这基础上可以进一步拓展到 rotation invariant, 就是把 x, y 换成 polar 坐标系, 看旋转角和长度 (非必要不要拓展到 rotation, false positive 比较多)

流程:

Local features->matched codebook entries->probabilistic voting->cluster voting space->backprojection of maxima-> backprojected hypotheses(+ backproject meta-information)-> pixel contributions-> segmentation

这里的亮点就是最后加了 segmentation

是训练时有些 images 就配有 polygonal object boundary 这种数据, 所以能从 patch 方块中变成 pixel 的边界

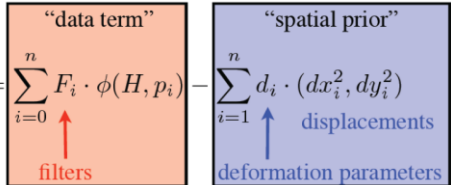
- Deformable part-based model

上面那个在 generalized hough transform,那么这个就是基于 HOG

DPM detector= global root filter+ 5~6 part filters(2 倍 resolution)

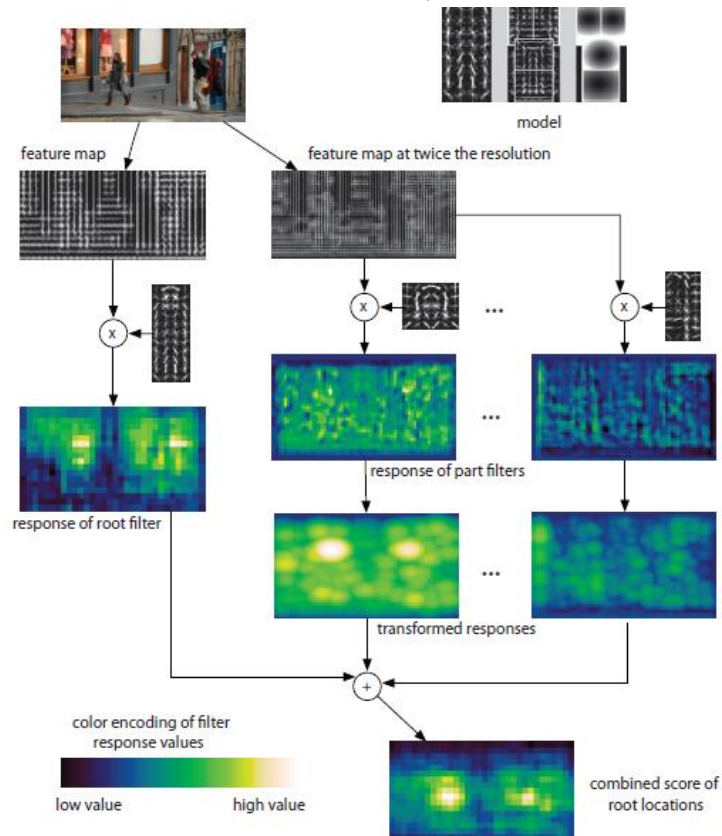
Part appearance + location distributions are learned from training

Part 都是通过 bounding box 单独训练的

$$\text{score}(p_0, \dots, p_n) = \sum_{i=0}^n F_i \cdot \phi(H, p_i) - \sum_{i=1}^n d_i \cdot (dx_i^2, dy_i^2)$$


p 是区域, H 是 hog, F 是 filter, 前面是加和的 root 和 part filter 的总分, 减去估计位置偏差, 就是最后得分

这种基于整体和局部的局限是 **false positives**，比如俩瓶盖当自行车



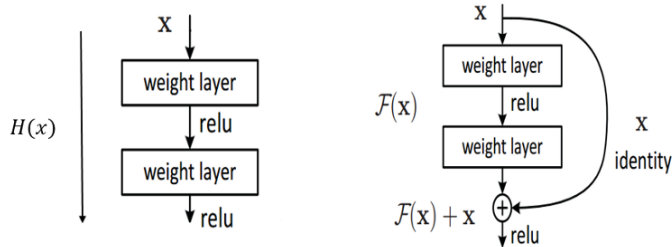
识别流程如上

Part15categorization

- Traditional recognition approach is using hand-designed features, 如 HOG, SIFT
现在我们学习 feature hierarchy all the way from pixels to classifier
- AlexNet 和 VGGNet 都是普通 cnn, 不过 VGG 更长, 每个 filter 3×3 , 更小, stride1, 也 smaller
- Cnn:
CNN learns the values of these filters on its own during the training process (although we still need to specify parameters such as number of filters, filter size, architecture of the network etc. before the training process)
- All neural net activations arranged in 3 dimensions(width, height, depth)
- Convolution 之后要引入 non-linear. tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations
- Pooling 时没有重复的, convolution 时可以 stride 1,2,3..
- 可以 0-padding,或者没有 padding, 就是 filter 不涉及方框外
- 最后 Fully connected layer is multi layer perceptron. Use softmax(好处是 sum of output probabilities==1)(or SVM) in the output layer
- Filter weights are shared between locations=> gradients are added for each filter location
- we can have multiple Convolution + ReLU operations in succession before a having a Pooling operation
- input $32 \times 32 \times 3$, convolution filter $5 \times 5 \times 3$, 12 个, 输出可以是 $32 \times 32 \times 12$ (意思就是 filter 一定管所有 depth, 但是输出仍然是 1 个)
- Prefer a stack of small filter CONV to one large receptive field CONV layer, 3 个 3×3 的 conv layer 叠在一起, 第三层就有 7×7 的视野, 但用 $3 \times 3 \times 3$ 的参数, 而不是 7×7 . 而且还有层间 non-linearities, more expressive. 当然实际运用时也有缺点, 就是 need more memory to hold all the intermediate CONV layer results

Part16categorization

- residual network(ResNet):



2 weight layers used to fit $H(x)$, now fit $F(x)$

- If identity is optimal, it is easy to set weights as 0;
- If optimal mapping closer to identity, easier to find small fluctuations
- Direct path for gradient to flow to the previous

- Applications of cnn

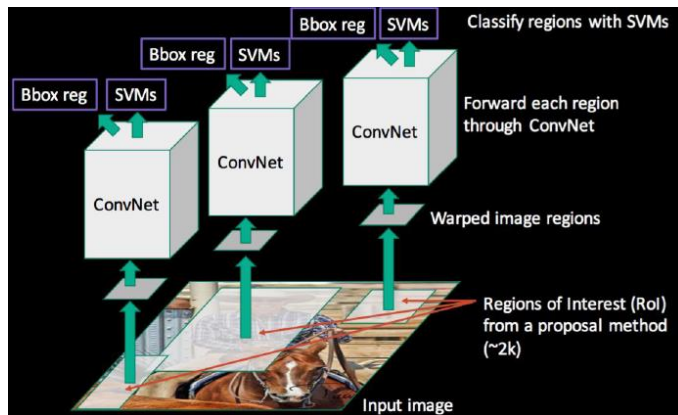
Transfer learning with cnn:

If small dataset: fix all weights, retrain classifier(最后的 FC 和 softmax)

If medium: use old weights as initialization, train full or some of higher layers(后几个 conv 以及 fc 以及 softmax)

- R-cnn(regions with cnn features):

- extract region proposals(selective search 这是单独的一个算法)
- Use pre-trained&fine-tuned classification network as feature extractor on those regions(bounding box regression and SVMs)



注意上面 Bbox regression, 就是根据 pool5 layer 的 features 来确定物体的 bounding box (和 proposal region 不一样)

Softmax (log loss)

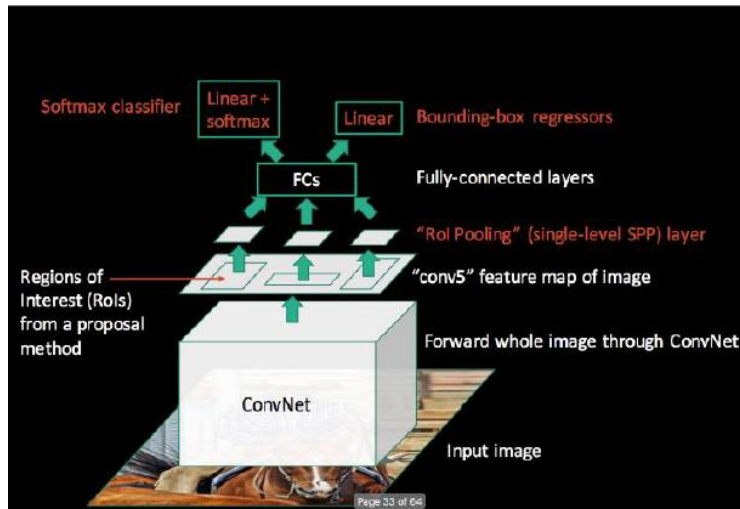
SVMs (hinge loss)

Bbox regressor (squared loss)

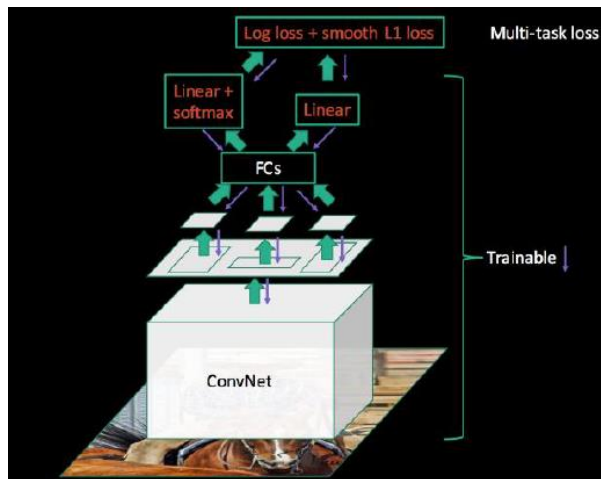
这个方法训练、使用都花大量时间, 也耗大量存储

- Fast r-cnn:

Forward pass:



Backward pass:



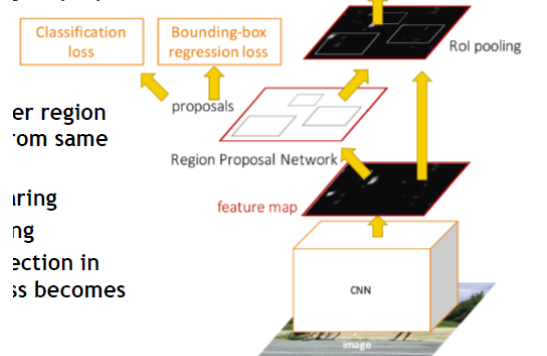
相比 rcnn，增加了 RoI pooling layer。Region proposal 在 cnn 提取的 feature map 上进行，然后 pooling。Fast rcnn take in entire image。

● Faster r-cnn(=fast rcnn + region proposal network):

infer region proposals from same CNN, feature sharing, joint training, object detection in a single pass

4, four losses

dependence on region proposal



四个 loss

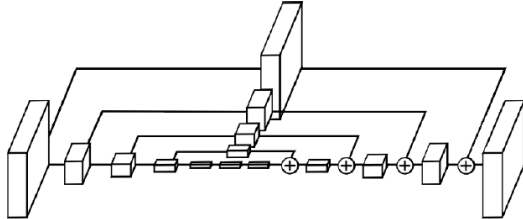
- Fully convolutional networks(FCN) for semantic segmentation

All operations formulated as convolutions

Can process arbitrarily sized images

FCN can be think of a sliding-window classification, producing a heatmap of output scores for each class

FCN output has low resolution; perform upsampling to get back to desired resolution and use skip connections to preserve higher-resolution information (底下是 encoder-decoder architecture)



- FCN for pose estimation
把关节 joint 定为 keypoints, 和上面一样 infer heatmaps for the joints
- Triplet loss network

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$

Use for face recognition; negative, anchor, positive

这个能做到笑脸女-中脸女+中脸男=笑脸男

Part17reconstruction

- Cues to construct stereo vision:

1. Shading
2. Texture
3. Focus
4. Perspective
5. motion

- 叉乘: $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \sin(\theta)$

- Camera calibration:

Extrinsic: rotation matrix + translation vector

Intrinsic: focal length, pixel sizes, image center point, radial distortion parameters

- Disparity: $x_r - x_l$, 指各自成像坐标的差

- Epipolar geometry:

Baseline: 俩相机中心连线

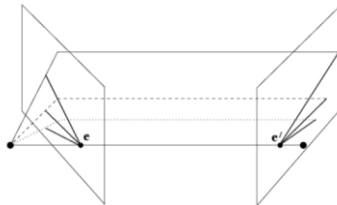
Epipole: baseline intersections with image plane(俩成像面各一个点)

Epipolar plane: plane containing baseline and world point

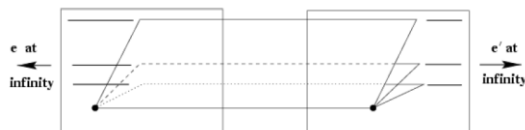
Epipolar line: intersection of Epipolar plane with image plane

- 示例三种情况

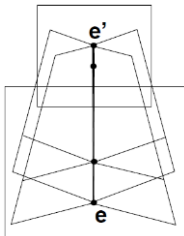
Converging(epipolarline 不平行,绕着 epipole 旋转)



Parallel(和 image plane 平着移动, 这时 epipolarline 都是平行的, epipole 在无限远)



Forward motion(epipole 在俩图里坐标一样, 它们也就是消失点 focus of expansion)



- 3D transformation: rotation + translation

$$\begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

$$X' = RX + T$$

(X 是向量，三维坐标，也是相机中心指向点的向量)

R 是 rotation，由如下三种围绕不同轴的旋转相乘得到

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

将 $X' = RX + T$ 左右都做叉乘转化得到 essential matrix E 如下

$$E = T_x R$$

$$X'^T E X = 0$$

其中 T_x 是将 T 叉乘表示成 skew symmetric matrix 的形式

$$\Omega = [\omega]_{\times} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

对于 image1 任意一点 p (不用 x 了)

$$p'^T E p = 0$$

因为 p' 落在 epipolarline l' 上面，所以 $p'l' = 0$

所以 $l' = E p$ 。我们对于一个点，左乘 essential matrix 就得到它对应的 epipolarline，然后就可以搜索这条线找最佳对应点了

● 平行变换的例子

各参数如下

$$R = I$$

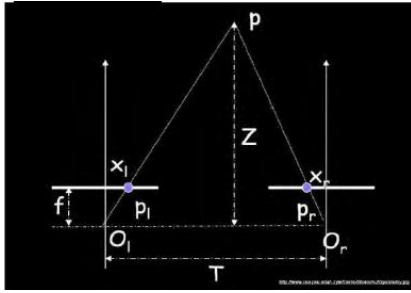
$$T = [-d, 0, 0]^T$$

$$E = [T_x] R = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & d \\ 0 & -d & 0 \end{bmatrix}$$

T 只有 x 的水平横移

$$\begin{bmatrix} x' & y' & f \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & d \\ 0 & -d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = 0$$

上面是对应关系式，注意到 z 轴，深度，都是 f



由式子解得 $y=y'$

也就是 parallel 的动，两图中高度 y 保持不变

此外景深 $Z=fT/(x_l-x_r)$

- Stereo image rectification

Algorithm:

Reproject image planes onto a common plane parallel to baseline

Pixel motion is horizontal now(就是 rectify 完了就可以水平 scan)

One homography(3*3) for each reprojection

- Dense correspondence search

1. Rectify images

2. For each pixel in left image:

Find corresponding Epipolar line in the right image(就是水平线)

Examine all pixels on the line and pick best match(比如用 correlation, sum of squared difference(SSD))

Triangulate the matches(即 disparity, 根据之前那个倒数关系) to get depth information

- Window search

本质就是用了上面的方法来找 disparity, 不过换成了 window 的匹配

Window size:

Smaller: good precision, more detail; but sensitive to noise

Larger: robust to noise; but reduced precision, less detail

- Sparse correspondence search

Sparse set of detected features(不用 pixel 用 feature descriptor, 一下少很多)

- Sparse:

Efficient

Reliable feature matches, robust

But have to know how to pick good features

Dense:

Simple

More depth estimates, can be useful for surface reconstruction

But not good with different viewpoints

- Stereo reconstruction steps

1. Calibrate cameras

2. Rectify images

3. Compute disparity

4. Estimate depth

Part18reconstruction

- Camera coordinate system:

Principal axis: from camera center perpendicular to image plane

Principal point(p): point principal axis intersects the image plane

Camera coordinate system: origin at principal point

Image coordinate system: origin is in the corner

$x=PX$

x 是 image 里面的坐标, X 是世界坐标, P 是转换矩阵

$P=K[I|0]$ (这个也是暂时的); K 如下

$$K = \begin{bmatrix} m_x & & & \\ & m_y & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} f & \textcolor{red}{S} & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \textcolor{red}{S} & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

(p_x, p_y) 就是 camera 原点在 image 里面的偏置

(m_x, m_y) 是横纵每米多少个像素

f 是 pinhole 模型的焦距, 像物比例就是 f/Z (都是到 camera center 的距离)

s 是 skew(non-rectangular pixel 会有径向畸变); radial distortion

之后再加上世界坐标系到 camera 坐标系的转换

$$x = K[I|0]X_{\text{cam}} = K[R|-R\tilde{C}]X$$

$$P = K[R|t], \quad t = -R\tilde{C}$$

R 是 3×3 的, 包括 rotation + translation

t 是 3×1 的

最后 P 的完全体:

note system)

$$P = K[R|t] = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix}$$

Intrinsic parameters:

1. Principal point coordinates 2
2. Focal length 1
3. Pixel magnification factors 1
4. Skew and radial distortion 1

Extrinsic parameters:

1. Rotation R 3
2. Translation t 3

12 只用 11, 因为 scale 是随便的

General camera: 11 degrees of freedom

General pinhole: 9

CCD with square pixels: 10

- Calibrating(构建 3D, 知道坐标, 看相机 image, 然后算 P):

Algorithm to make calibration points subpixel accuracy for checkerboard(跳棋盘) pattern:

1. Perform canny edge detection
2. Fit straight lines to edges
3. Intersect lines to obtain corners

一条经验

Constraints should be *5 larger than unknowns

P has 11, so 28 points, $28 \times 2 > 11 \times 5$

Direct linear transform (DLT):

$$\lambda \mathbf{x}_i = \mathbf{P} \mathbf{X}_i \quad \lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{bmatrix} \begin{bmatrix} X_{i,1} \\ X_{i,2} \\ X_{i,3} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^T \\ \mathbf{P}_2^T \\ \mathbf{P}_3^T \end{bmatrix} \mathbf{X}_i$$

$$\mathbf{x}_i \times \mathbf{P} \mathbf{X}_i = 0$$

$$\begin{bmatrix} 0 & -\mathbf{X}_i^T & y_i \mathbf{X}_i^T \\ \mathbf{X}_i^T & 0 & -x_i \mathbf{X}_i^T \\ -y_i \mathbf{X}_i^T & x_i \mathbf{X}_i^T & 0 \end{bmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} = 0$$

上面的式子用了 skew symmetric matrix 来把 \mathbf{x}_i 叉乘给换成了矩阵

$$\begin{bmatrix} 0^T & \mathbf{X}_1^T & -y_1 \mathbf{X}_1^T \\ \mathbf{X}_1^T & 0^T & -x_1 \mathbf{X}_1^T \\ \dots & \dots & \dots \\ 0^T & \mathbf{X}_n^T & -y_n \mathbf{X}_n^T \\ \mathbf{X}_n^T & 0^T & -x_n \mathbf{X}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} = 0$$

SVD 解出来, 最少要 5.5 个点, 因为每个点出俩 independent 的解出来和具体参数不对应, 还得处理一下

- need calibration points not:
 1. points all lie on the union of a plane and a single straight line containing the camera center
 2. The camera and points all lie on a twisted cubic

- Revisiting triangulation

估计 image 点 \mathbf{x}, \mathbf{x}' 的原世界点 \mathbf{X} 方法:

1. find shortest segment connecting the two viewing rays, let X be the midpoint of the segment
2. 线性代数解法

$$\lambda_1 x_1 = P_1 X \quad x_1 \times P_1 X = 0 \quad [x_{1 \times}] P_1 X = 0$$

$$\lambda_2 x_2 = P_2 X \quad x_2 \times P_2 X = 0 \quad [x_{2 \times}] P_2 X = 0$$

我们知道 x_1, P_1 (世界点到 image 转化矩阵), 3 维中 2 个 independent 式子, 所以上面俩就有 4 个式子解 3 个未知数, SVD

这个方法拓展到多相机很容易, 就是再加几个式子

3. 最精确的, 但是不能解, 只能用 Gauss-Newton 来 iterative 估计的

Find X that minimizes

$$d^2(x_1, P_1 X) + d^2(x_2, P_2 X)$$

Calibrated case:

$$x \cdot [t \times (R x')] = 0 \quad \Rightarrow \quad x^T E x' = 0 \quad \text{with} \quad E = [t_{\times}] R$$

$E e' = 0$ and $E^T e = 0$

E rank 2, 5 自由度, 3 rotation + 3 translation - 1 scale

Uncalibrated case:

$$\hat{x}^T E \hat{x}' = 0 \quad \Rightarrow \quad x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

Calibration matrices K, K' 是未知的, 实际上世界点转到 x, 需要 K, 但是 x^\wedge 就不用转 K 那么多, 所以 $x K^{-1} = x^\wedge$, 注意上面负号

$F e' = 0$, $F^T e = 0$

F rank 2, 7 自由度

要算 F, 如下:

$$x = (u, v, 1)^T, \quad x' = (u', v', 1)^T$$

$$(u, v, 1) \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \quad \Rightarrow \quad [u'u, u'v, u', uv', vv', v', u, v, 1] \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0$$

• Taking 8 correspondences:

$$\begin{bmatrix} u'_1 u_1 & u'_1 v_1 & u'_1 & u'_1 v'_1 & v'_1 v'_1 & v'_1 & u_1 & v_1 & 1 \\ u'_2 u_2 & u'_2 v_2 & u'_2 & u'_2 v'_2 & v'_2 v'_2 & v'_2 & u_2 & v_2 & 1 \\ u'_3 u_3 & u'_3 v_3 & u'_3 & u'_3 v'_3 & v'_3 v'_3 & v'_3 & u_3 & v_3 & 1 \\ u'_4 u_4 & u'_4 v_4 & u'_4 & u'_4 v'_4 & v'_4 v'_4 & v'_4 & u_4 & v_4 & 1 \\ u'_5 u_5 & u'_5 v_5 & u'_5 & u'_5 v'_5 & v'_5 v'_5 & v'_5 & u_5 & v_5 & 1 \\ u'_6 u_6 & u'_6 v_6 & u'_6 & u'_6 v'_6 & v'_6 v'_6 & v'_6 & u_6 & v_6 & 1 \\ u'_7 u_7 & u'_7 v_7 & u'_7 & u'_7 v'_7 & v'_7 v'_7 & v'_7 & u_7 & v_7 & 1 \\ u'_8 u_8 & u'_8 v_8 & u'_8 & u'_8 v'_8 & v'_8 v'_8 & v'_8 & u_8 & v_8 & 1 \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$A f = 0$$

Solve using... SVD!

This minimizes:

$$\sum_{i=1}^N (x_i^T F x'_i)^2$$

每一对点, 出一个式子(注意顺序, 是反的)

SVD 完 A 之后, 得到 V 最右边一列作为 F

因为数据嘈杂, 但我们知道是 rank2 所以把第三特征值设为 0, 把 F 再给 SVD 了

SYD

$$F = \mathbf{U} \mathbf{D} \mathbf{V}^T = \mathbf{U} \begin{bmatrix} d_{11} & & & \\ & d_{22} & & \\ & & d_{33} & \\ & & & \ddots \end{bmatrix} \begin{bmatrix} v_{11} & \cdots & v_{13} \\ \vdots & \ddots & \vdots \\ v_{31} & \cdots & v_{33} \\ \vdots & \ddots & \vdots \end{bmatrix}^T$$

Set d_{33} to zero and reconstruct F

Set 完简单的再 $F_{\text{new}} = \mathbf{U} \mathbf{D}_{\text{new}} \mathbf{V}^T$ 即可

再加 normalization:

1. Center image data at the origin, scale it so mean squared distance between origin and data points is 2 pixels
2. Use 8-point algorithm to compute F
3. Enforce rank-2 constraint by SVD setting 0
4. Normalizing transformations are T, T' . then fundamental matrix is $T^T F T'$

上面 8 点法，是 minimize 了如下，因为都设置为 0 了

$$\text{error} \sum_{i=1}^N (x_i^T F x'_i)^2$$

类似求 x 时，对于 F 也可以 nonlinear 求解，minimize 如下式子即可：

$$\sum_{i=1}^N [d^2(x_i, F x'_i) + d^2(x'_i, F^T x_i)]$$

使用 8 点法流程：

1. Find interest points(by Harris corners)
2. Match points
3. Use RANSAC, select 8 correspondences, compute F , see inliers. Choose most F with most inliers

知道点 x 在 1,2 的投影，求在第三个，下式的两线交点

$$l_{31} = F_{13}^T x_1$$

$$l_{32} = F_{23}^T x_2$$

Part19reconstruction

- active Stereo

Kinect: replace one camera by a projector

Project structured light patterns onto object(simplifies correspondence, use only one camera)

Other approach:

Laser scanning(a single stripe of laser light)

加快，可以 multi-stripe(为了辨别哪一条对应,need time-coded light patterns:a unique illumination code over time)

Part 20 motion

- motion estimation:

Direct methods: recover image motion at each pixel from spatio-temporal image brightness variation; dense; when motion small

Feature-based methods: extract visual features and track them over multiple frames; sparse; when motion large

- $V=[V_x, V_y, V_z]$; 这个是世界点的速度. $P=fP/Z$; 这是图像点和世界点的关系

$$\mathbf{v} = f \frac{Z\mathbf{V} - V_z\mathbf{P}}{Z^2} = \frac{f\mathbf{V} - V_z\mathbf{p}}{Z}$$

Image 速度如上, $(fP/Z)'$ 得到的变化一下:

$$\mathbf{v} = \frac{1}{Z}(\mathbf{v}_0 - V_z\mathbf{p}),$$

$$\mathbf{v}_0 = (fV_x, fV_y)$$

V_z is non-0: 所有 motion vector 指向或者背离 \mathbf{v}_0 , \mathbf{v}_0 是消失点

V_z is 0: 运动和 image 面平行, 所有 motion vector 平行

- Optical flow:

根据 brightness 的变化来判断 motion field; 一般是一样的, 但比如球, 以及发廊, 都是有不一样的情况, the aperture(孔洞) problem

Assumptions 有 3 个:

1. Brightness constancy: 同一个点, 不同时间或 frame 都长一个样
2. Small motion
3. Spatial coherence: 和邻居运动一致

Brightness constancy equation:

$$I(x, y, t-1) = I(x+u(x, y), y+v(x, y), t)$$

u, v 是 x, y 方向的速度, 这里是说俩不同时间, 移动后的点和移动前是一样的 intensity

$$\partial_x (I_x) \cdot u + \partial_y (I_y) \cdot v + \partial_t (I_t) \approx 0$$

Spatial derivatives

Temporal derivative

对于一个点来说, 一个式子, 两个未知数

所以假装邻居运动一样, 5×5 , 得到如下一组方程

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_{25}) & I_y(p_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(p_1) \\ I_t(p_2) \\ \vdots \\ I_t(p_{25}) \end{bmatrix} \quad \begin{matrix} A & d = b \\ 25 \times 2 & 2 \times 1 & 25 \times 1 \end{matrix}$$

要解上面就得常规的 $(A^T A)^{-1} A^T b$, 下面是两边同乘 A^T , 叫做 Lucas-Kanade equation

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$A^T A \qquad A^T b$

注意到左边 $A^T A$, 和 Harris corner detector 中 second moment matrix 是一个东西, 其实和 Harris corner 就是一个套路, 看微移的 intensity 变化, 所以 harris 的一套这边也适用
所以 edge(一大一小 eigenvalue)和 low-texture(俩小)的点不好算 optical flow, 要用 corners 或者 high-texture areas

实际运用时算法:

1. estimate v at each pixel using one iteration of Lucas-Kanade estimation
2. warp one image toward the other using the estimated flow field
3. refine estimate by repeating

注意事项:

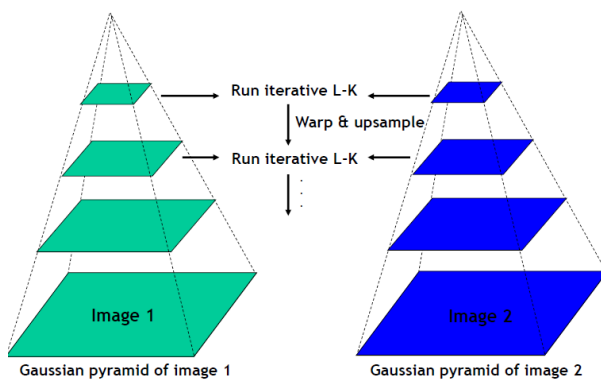
1. Ensure errors in warping < estimate refinement
2. Warp one image, take derivatives of the other(so you don't need to re-compute the gradient each iteration)
3. Low-pass filter image before motion estimation(这应该和底下的意思一致)

Lucas-Kanade 中可能的问题以及解决方法(就是之前三条要求的一一对应):

1. Brightness constancy does not hold: do exhaustive neighborhood search with normalized correlation
2. Motion is large: coarse-to-fine estimation
3. Point not move like its neighbor: motion segmentation

Motion is large, temporal aliasing(就是找最近相同 intensity 点)问题的解决:

Coarse-to-fine optical flow estimation(从上到下):



高斯 pyramid 确实是每层 blur, 从下到上

- KLT feature tracking

由两张图拓展到多个 frame 上的 tracking，就是第一张到第二张，挨个来可能的问题以及解决方法：

1. Ambiguity of optical flow: use good features to track
2. Large motion: discrete search instead of Lucas-Kanade
3. Change in shape, orientation, color: allow matching flexibility
4. Occlusions, disocclusions: deleting old, add new features
5. Drift(error 累积):need to know when to stop

这里展开第二点讲：

1. Define a small area around a pixel as the template(就是区域特征当 feature)
2. Match template against each pixel within a search area in next image(match measure like SSD or correlation)就是特征匹配
3. After finding the best location, can use Lucas-Kanade to get sub-pixel estimate (特征匹配完了可以用 25 个更好的算 motion)

Shi-Tomasi feature tracker (KLT tracking)

1. Find good features using eigenvalues of second moment matrix(应该就是找特征值大的)
2. From frame to frame, track with Lucas-Kanade and a pure translation model
3. Check consistency of tracks by affine registration to the first observed instance of the feature(就是看变化太大就得换了)

Part21reconstruction

- Reconstruction Ambiguity:

给场景一个 Q, 给 camera matrices 一个 Q 的逆, 图像不变。如果没有别的限制, 我们就只能得到 projective 级别的, 所以很难构建原场景

$$\mathbf{x} = \mathbf{P}\mathbf{X} = (\mathbf{P}\mathbf{Q}^{-1})\mathbf{Q}\mathbf{X}$$

Projective 15 (保持了交点和 tangency)

Affine 12 (这是算上 3 rotation, 3 translation, 3 scale, 3 skew)

Similarity 7

Euclidean 6(可以看做是全等构建)

- Affine structure from motion

Perspective 是透视变换, weak perspective 是近似于平行

Affine camera= orthographic + parallel projection

Orthographic projection(正射):

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Affine camera combines affine of 3D space + orthographic projection + affine of image:

$$\mathbf{P} = [\mathbf{3 \times 3 \text{ affine}}] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [\mathbf{4 \times 4 \text{ affine}}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

注意, 是 8 个未知量

m images; n fixed 3D points:

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, j = 1, \dots, n$$

我们用 2mn 这种关系, 来估计 m 个 projection 矩阵 A, 以及 translation vector b (看上上个式子, A 是 2*3, b 是 2*1), 和 n 个世界点 X

8m+3n 个未知量, 要减去 12 个 affine ambiguity 的自由度

$$2mn \geq 8m + 3n - 12$$

第一步是 image points 全放中心, 我们假设世界点质心就是原点, 则

$$\hat{\mathbf{x}}_{ij} = \mathbf{A}_i \mathbf{X}_j \quad (\text{把 } \mathbf{b} \text{ 给消失了})$$

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

Cameras
($2m \times 3$)

Points ($3 \times n$)

我们拿到的是 D，把它 SVD 了，因为知道是 rank3，只去 3 列，3*3,3 行(转置后的)

$$\begin{matrix} 2m \\ \updownarrow \end{matrix} \begin{matrix} \mathbf{D} \end{matrix} = \begin{matrix} \downarrow 3 \\ \mathbf{U}_3 \end{matrix} \times \begin{matrix} \leftarrow 3 \\ \mathbf{W}_3 \end{matrix} \times \begin{matrix} \leftarrow n \\ \mathbf{V}_3^T \end{matrix} \begin{matrix} \updownarrow 3 \end{matrix}$$

Possible decomposition:
 $\mathbf{M} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \quad \mathbf{S} = \mathbf{W}_3^{1/2} \mathbf{V}_3^T$

把中间拆成一半，各自乘到左右

我们这种 decomposition 不是唯一的，任何 3*3 的 C，M 右乘，S 左乘，结果就保持不变

Euclidean upgrade

因为 orthographic，所以 perpendicular & scale=1:

$$\begin{cases} \hat{a}_{i1} \cdot \hat{a}_{i2} = 0 \\ |\hat{a}_{i1}| = 1 \\ |\hat{a}_{i2}| = 1 \end{cases} \Leftrightarrow \begin{cases} a_{i1}^T C C^T a_{i2} = 0 \\ a_{i1}^T C C^T a_{i1} = 1, \quad i=1, \dots, m \\ a_{i2}^T C C^T a_{i2} = 1 \end{cases}$$

$$\mathbf{L} = \mathbf{C} \mathbf{C}^T \quad \mathbf{A}_i = \begin{bmatrix} a_{i1}^T \\ a_{i2}^T \end{bmatrix}, \quad i=1, \dots, m$$

$$\mathbf{A}_i \mathbf{L} \mathbf{A}_i^T = \mathbf{I}, \quad i=1, \dots, m$$

把 L 解出来，然后 cholesky decomposition: $\mathbf{L} = \mathbf{C} \mathbf{C}^T$

Update M and S: $\mathbf{M} = \mathbf{M} \mathbf{C}$, $\mathbf{S} = \mathbf{C}^{-1} \mathbf{S}$

Dealing with missing data:

Decompose matrix into dense sub-blocks, factorize each sub-block and fuse the results
(incremental bilinear refinement)

- Projective structure from motion:

$$\mathbf{z}_{ij} \mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j,$$

No calibration information

$$2mn \geq 11m + 3n - 15$$

Two-camera case:

First camera matrix: $[I|0]Q^{-1}$

Second camera matrix: $[A|b]Q^{-1}$

$$\mathbf{F} = [\mathbf{b}_\times] \mathbf{A} \quad \mathbf{b}: \text{epipole } (\mathbf{F}^T \mathbf{b} = 0), \quad \mathbf{A} = -[\mathbf{b}_\times] \mathbf{F}$$

这表明我们如果知道 \mathbf{F} , 就知道两个 projection matrices

如果知道 projection matrices, 我们就能计算任意点 \mathbf{X}

$$\mathbf{D} = \begin{bmatrix} z_{11}\mathbf{x}_{11} & z_{12}\mathbf{x}_{12} & \cdots & z_{1n}\mathbf{x}_{1n} \\ z_{21}\mathbf{x}_{21} & z_{22}\mathbf{x}_{22} & \cdots & z_{2n}\mathbf{x}_{2n} \\ & & \ddots & \\ z_{m1}\mathbf{x}_{m1} & z_{m2}\mathbf{x}_{m2} & \cdots & z_{mn}\mathbf{x}_{mn} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}$$

Cameras $(3m \times 4)$ Points $(4 \times n)$

$\mathbf{D} = \mathbf{MS}$ has rank 4

Iterative approach:

1. Initialize motion from 2 images using fundamental matrix
2. Initialize structure
3. For each additional view
 - Determine projection matrix of new camera using all 3D points visible in its image
 - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera
4. Refine structure and motion: bundle adjustment

Bundle adjustment:

Minimize error using maximum likelihood solution assuming Gaussian noise

$$E(\mathbf{P}, \mathbf{X}) = \sum_{i=1}^m \sum_{j=1}^n D(\mathbf{x}_{ij}, \mathbf{P}_i \mathbf{X}_j)^2$$

Euclidean upgrade if we know about calibration or about the scene

Self-calibration

Use constraint on calibration: square pixels, zero skew, fixed focal length

或者都是自己一个相机拍的, 可以 find projective transformation \mathbf{Q} such that all camera matrices are in the form $\mathbf{P}_i = \mathbf{K}[\mathbf{R}_i | \mathbf{t}_i]$

● Practical issues:

Baseline: small 则 large depth error, big 则 difficult search problem

Solution: Track features between frames until baseline is sufficient

Points close together produce less stable solution

Solution: subdivide image, extract same number of features per grid

- Sfm limitations:
 1. Difficult to reliably estimate metric sfm unless large motion of large field-of-view
 2. Camera calibration important for Euclidean reconstruction
 3. Need good feature tracker

Questions:

1. 2,P19, what hidden assumption?
2. 2, P34, how to deal with line?那个问题
3. 4, P49, 第三句话, 什么 canny has shown...
4. 当使用 convolution 时, $2*1$ 这种算在哪一个上面啊?
5. 18, P56, yellow