希文 23 春 计算机视觉 23 年 6 月 14 日

图像形成

齐次坐标表示: $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) \in \mathcal{P}^2$, 其中 $\mathcal{P}^2 = \mathcal{R}^3 - (0,0,0)$ 为二维射影空间. 齐次坐标(x,y,w)对应笛卡尔坐标$\left(\frac{x}{w}, \frac{y}{w}\right)$. $\tilde{\mathbf{x}} = (\tilde{x}, \tilde{y}, \tilde{w}) = \tilde{w}(x,y,1) = \tilde{w}\bar{\mathbf{x}}$. 其中 $\bar{\mathbf{x}} = (x,y,1)$ 被称为增广向量.当齐次向量 $\tilde{\mathbf{x}}$ 的 $\tilde{w}=0$ 时, 也可称为理想点或**无穷远点**, 它没有对应的非齐次表示.

Translation 平移 2D translations can be written as $\mathrm{x}' = \mathrm{x} + \mathrm{t}$ or $\mathbf{x}' = [\mathbf{I} \quad \mathbf{t}]\bar{\mathbf{x}}$, $\mathbf{I}$ is the (2×2) identity matrix or $\bar{\mathbf{x}}' = \begin{bmatrix} \mathbf{I} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}\bar{\mathbf{x}}$. 2 自由度, orientation 不变.

Rotation+Translation 平移+旋转,also known as $2D$ rigid body motion or the $2D$ Euclidean transformation. It can be written as $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$ or $\mathbf{x}' = [\mathbf{R} \quad \mathbf{t}]\bar{\mathbf{x}}$ where $\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is an orthonormal rotation matrix with $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $|\mathbf{R}| = 1$. 3 自由度, lengths 不变. 注意这个是逆时针旋转.

Scaled Rotation 缩放旋转; 相似变换 imilarity transform, this transformation can be expressed as $\mathbf{x}' = s\mathbf{R}\mathbf{x} + \mathbf{t}$, where s is an arbitrary scale factor. also as $\mathbf{x}' = [s\mathbf{R} \quad \mathbf{t}]\bar{\mathbf{x}} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix}\bar{\mathbf{x}}$, we no longer require that $a^2 + b^2 = 1$. The similarity transform preserves angles between lines. 4 自由度, angles 不变.

Affine 仿射 The affine transformation is written as $\mathbf{x}' = \mathbf{A}\bar{\mathbf{x}}$, where $\mathbf{A}$ is an arbitrary 2×3 matrix, i.e., $\mathbf{x}' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix}\bar{\mathbf{x}}$. Parallel lines remain parallel under affine transformations. 6 自由度.

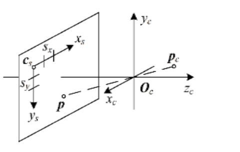
Projective 投影 This transformation, also known as a perspective transform (透视变换) or homography, operates on homogeneous coordinates, $\tilde{\mathbf{x}}' = \tilde{\mathbf{H}}\tilde{\mathbf{x}}$, where $\tilde{\mathbf{H}}$ is an arbitrary 3×3 matrix. Note that $\tilde{\mathbf{H}}$ is homogeneous, i.e., it is only defined up to a scale, and that two $\tilde{\mathbf{H}}$ matrices that differ only by scale are equivalent. The resulting homogeneous coordinate $\tilde{\mathbf{x}}'$ must be normalized in order to obtain an inhomogeneous result x, i.e., $x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}}$, $y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}$. Perspective transformations preserve straight lines (i.e., they remain straight after the transformation).8 自由度

三维变换: Translation: $[\mathbf{I} \quad \mathbf{t}]_{3\times4}$, 3 自由度; Rigid(Euclidean): $[\mathbf{R} \quad \mathbf{t}]_{3\times4}$, 6 自由度; Similarity: $[s\mathbf{R} \quad \mathbf{t}]_{3\times4}$, 7 自由度; Affine: $[\mathbf{A}]_{3\times4}$, 12 自由度; Projective: $[\tilde{\mathbf{H}}]_{4\times4}$, 15 自由度.

三维到二维投影: 正交投影是最简单的投影模型. 去掉三维坐标 $\mathbf{p}$ 的 z 分量, 得到对应的二维点 $\mathbf{x}$, 记作$\mathbf{x} = [\mathbf{I}_{2\times2} \mid \mathbf{0}]\mathbf{p}$. 齐次坐标的表示形式为: $\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\tilde{\mathbf{p}}$, 去掉了 z 分量, 但保留了 w 分量. 缩放正交投影: $\mathbf{x} = [s\mathbf{I}_{2\times2} \mid \mathbf{0}]\mathbf{p}$.

透视投影: $\bar{\mathbf{x}} = \mathcal{P}_z(\mathbf{p}) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$, 齐次坐标表示形式: $\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\tilde{\mathbf{p}}$.去掉了$w$分量, 因此无法从图像中恢复距离信息. 如针孔相机的投影模型: $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$.

相机内参数:例如焦距, 光学中心, 图像高宽比;描述是什么类型的相机.相机外 (姿态) 参数:包括旋转和平移;描述相机在什么位置.



orientation of the sensor array relative to the camera projection center $\mathbf{O}_c$ with an origin $\mathbf{c}_s$ and a 3D rotation $\mathbf{R}_s$ 则$\mathbf{p} = [\mathbf{R}_s \quad \mathbf{c}_s]\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \mathbf{M}_s\bar{\mathbf{x}}_s$. 但是这样自由度太高了, impractical. The relationship between the 3D pixel center $\mathbf{p}$ and the 3D camera-centered point $\mathbf{p}_c$scaling s, $\mathbf{p} = s\mathbf{p}_c$.

更简化的形式: $\mathbf{x} = [sI_{2\times2} \mid 0]\mathbf{p}$, 齐次坐标$\tilde{\mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\tilde{\mathbf{p}}$; 相机内参数矩阵.$\mathbf{x} = [s\mathrm{R}_{3\times3} \mid \mathrm{t}]\mathrm{p}$, $\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}[\mathbf{R} \mid \mathbf{t}]\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$.相机外参数矩阵. 标定校准, u, v 代表二维分量, X, Y, Z 代表三维分量, 取代 x, p, 使用已知的三维空间点及其像素位置, 求解矩阵中的 11 个未知数.

汇总表示: $\mathbf{x} = \mathbf{K}[\mathbf{R} \quad \mathbf{t}]\mathbf{X}$, 其中$\mathbf{x}$: Image Coordinates: $(u,v,1)$;$\mathbf{K}$: Intrinsic Matrix (3×3);$\mathbf{R}$: Rotation (3×3); $\mathbf{t}$: Translation (3×1); $\mathbf{X}$: World Coordinates: $(X,Y,Z,1)$ 内参矩阵就是将图像坐标系转化为像素坐标系, $\begin{bmatrix} f/dx & 0 & u_0 \\ 0 & f/dy & v_0 \\ 0 & 0 & 1 \end{bmatrix}$, 其中 u0,v0,主点的实际位置, 单位也是像素.外参矩阵就是真实物理空间坐标到相机坐标系的旋转加平移变换, 本质就是一个变换矩阵$\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}^\top & 1 \end{bmatrix}$.内外参数矩阵形式: $\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f/dx & 0 & u_0 \\ 0 & f/dy & v_0 \\ 0 & 0 & 1 \end{bmatrix}[R \quad t]\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = M\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$优点: 相机所有特性都集中在一个矩阵中; 能够计算任意三维点在图像中的成像位置; 缺点: 不能单独计算特定参数; 内部参数与外部参数混合, 导致相机位置不能变, 一旦变了就失效. 此时可以使用运动校准, 也就是利用运动标定板来实现, 不需要知道位置和朝向;

将中间的 image plane 坐标系的坐标移动到左上角, 我们就需要增加一些映射关系

$$\begin{pmatrix} x_s \\ y_s \end{pmatrix} = \begin{pmatrix} f_x\, x_c/z_c + s\, y_c/z_c + c_x \\ f_y\, y_c/z_c + c_y \end{pmatrix} \quad \Leftrightarrow \quad \tilde{\mathbf{x}}_s = \begin{bmatrix} f_x & s & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\bar{\mathbf{x}}_c$$

这个映射矩阵就成为相机内参矩阵 (camera instrinsic), 表示把三维坐标映射到图像平面。

图像的颜色空间. additive colors red, green, and blue; subtractive colors cyan, magenta, and yellow. 加色混合: 红色、 绿色和蓝色可以混合产生青色、 品红色、 黄色和白色. 减色混合: 蓝、 洋红和黄色可以混合产生红色、 绿色、 蓝色和黑色 **RGB**: $R = \int L(\lambda)S_R(\lambda)d\lambda$, $G = \int L(\lambda)S_G(\lambda)d\lambda$. where $L(\lambda)$ is the incoming spectrum of light at a given pixel and $\{S_R(\lambda), S_G(\lambda), S_B(\lambda)\}$ are the red, green, and blue spectral sensitivities of the corresponding sensors.

负值问题, 以及很难区分, 故 CIE 定义了 **XYZ** 空间.The transformation from RGB to XYZ is given by $\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{0.17697}\begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0.00 & 0.01 & 0.99 \end{bmatrix}\begin{bmatrix} R \\ G \\ B \end{bmatrix}$

Because the response of the human visual system is roughly logarithmic (we can perceive relative luminance differences of about 1%), the CIE defined a non-linear re-mapping of the XYZ space called L*a* b* (also sometimes called **CIELAB**), where differences in luminance or chrominance are more perceptually uniform, as shown in Figure 2.30 b. The L* component of lightness is defined as $L^* = 116f\left(\frac{Y}{Y_n}\right)$, where Y_n is the luminance value for nominal white (Fairchild 2013) and $f(t) = \begin{cases} t^{1/3} & t > \delta^3 \\ t/(3\delta^2) + 2\delta/3 & \text{else} \end{cases}$ is a finite-slope approximation to the cube root with $\delta = 6/29$. The resulting 0 ... 100 scale roughly measures equal amounts of lightness perceptibility.In a similar fashion, the a^* and b^* components are defined as $a^* = 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right]$ and $b^* = 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right]$ where again, (X_n, Y_n, Z_n) is the measured white point.

If we divide the XYZ values by the sum of X + Y + Z, we obtain the chromaticity coordinates $x = \frac{X}{X+Y+Z}$, which sum to 1. 此外注意$r = \frac{R}{R+G+B}$.

其他:BRDF: Bidirectional Reflectance Distribution Function. 表面光反射分布函数.BRDF:$f(\theta_i, \phi_i; \theta_r, \phi_r) = \frac{L_{\text{surface}}(\theta_r, \phi_r)}{E_{\text{surface}}(\theta_i, \phi_i)}$.像素值 = 体反射 + 表面反射.枕状畸变: 内凹型; 桶状畸变: 外凸型; 薄透镜成像: $\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$. 曝光 = 光圈 + 快门速度: 光圈的直径 D 限制光的入射范围 (光圈可以在透镜的任一侧);快门速度决定从光圈射入的光的量;更改光圈尺寸影响景深; 小光圈能增加成像基本清晰的范围

图像处理

数字图像: 数字图像可以视为一种函数: $f: R^2 \to R$.通常情况下, 我们期望图像包含有限数量的像素, 且幅值在有限范围内。例如 $f: [a,b] \times [c,d] \to [0,255]$.彩色图像可以视为三个函数的组合.数字图像经过采样和量化处理: 像素坐标和幅值均离散化.二维采样: 图像空间坐标的离散化;幅值量化: 图像幅度坐标的离散化.

点运算符: 常见点运算符有控制对比度、gamma 变换、直方图均衡化.图像合成与蒙版. **直方图均衡化**: histogram: $h(I)$; $c(I) = \frac{1}{N}\sum_{i=0}^{I} h(i) = c(I-1) + \frac{1}{N}h(I)$, where N is the number of pixels in the image. For any given grade or intensity, we can look up its corresponding percentile $c(I)$ and determine the final value that the pixel should take. When working with eight-bit pixel values, the I and c axes are rescaled from [0,255]. 完全直方图均衡化: $f(I) = c(I)$; 部分直方图均衡化: $f(I) = \alpha c(I) + (1-\alpha)I$ 蒙版$C = (1-\alpha)B + \alpha F$.

邻域运算: 线性滤波使用邻域的**线性组合**代替原来的像素值 $g(i,j) = \sum_{k,l} f(i+k, j+l)h(k,l)$, 其中$h(k,l)$为滤波核. 等价的**卷积**形式: $g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l)$, 或记为$g = f * h$.

箱式滤波: 滑动平均 -- 通过滑动窗口的平均数得到新的像素值. 以局部均值代替原像素值; 也称为平方平均滤波; 具有平滑和模糊的效果. 滤波核是$\frac{1}{9}$乘以一个全是 1 的3×3矩阵.**高斯滤波**: 加权滑动平均 -- 使用滑动窗口的加权平均得到新的像素值. 从二维连续的高斯函数采样得到, 滤波器的权值从中心到外围逐渐减小. $G_\sigma = \frac{1}{2\pi\sigma^2}e^{-\frac{(x^2+y^2)}{2\sigma^2}}$高斯核使得滤波后的图像更加平滑, σ 决定平滑程度.性质: 去除图像中的"高频"成分 (低通滤波器); 高斯核与自身的卷积仍然是高斯核. 如何选择滤波核的尺寸?尽量使边缘处的值接近 0; 经验法则: 滤波核宽度为 6σ.

若图像尺寸为 $M\times M$, 卷积核尺寸为 $N\times N$, 采用不可分滤波器需要进行多少次**乘法运算**? 每像素 $N\times N$ 次, 共 $N^2\times M^2$ 次. 采用可分滤波器需要进行多少次乘法运算? 每像素 $2\times N$ 次, 共 $2\times N\times M^2$ 次. 可分卷积核: 一些二维卷积核可写成行向量和列向量的乘积, 比如前边箱式滤波那里.

傅里叶变换. 连续域$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(x,y)e^{-j(\omega_x x + \omega_y y)}dxdy$, 离散域 $H(k_x, k_y) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} h(x,y)e^{-j2\pi(k_x x/M + k_y y/N)}$, 其中$M,N$ 分别表示图像的长和宽.$\mathrm{F}[g*h] = \mathrm{F}[g]\mathrm{F}[h]$空域卷积等价于频域乘积! 为什么需要傅里叶变换? 使用快速傅立叶变换允许我们使用在时间上与内核大小无关的大型内核卷积; 此外,转换到频域可以帮助我们理解滤波器的特性. 用傅里叶变换可以理解为什么高斯滤波比箱式滤波更平滑.

全局运算: 高斯滤波后降采样: 原始图片, 先高斯滤波, 再删去偶数行, 偶数列, 获得图片 1. 对图片 1 先高斯, 再删去偶数行, 偶数列. 高斯金字塔的构建: 1) 滤波 2) 降采样 3) 重复 1,2 直到达到目标分辨率

几何变换: 在 Point Process 是对 **range** of the image 做变换, $g(\mathbf{x}) = h(f(\mathbf{x}))$, 而这里我们要转换 **domain**, $g(\mathbf{x}) = f(\mathbf{h}(\mathbf{x}))$.参数变换是指由少量参数控制的几何变换, 比如前边那些二维三维变换.

前向算法, 输入f, $\mathbf{h}$, 输出g. 则 For every pixel $\mathbf{x}$ in $f(\mathbf{x})$, 1)Compute the destination location $\mathbf{x}' = \mathbf{h}(\mathbf{x})$, 2)Copy the pixel $f(\mathbf{x})$ to $g(\mathbf{x}')$. 前向算法存在的问题:x' 具有非整数值时, 将像素 $f(x)$复制到 g 中的位置 x' 的过程没有得到很好的定义;若将 x' 四舍五入到整数, 则将导致锯齿;一些 x' 没有赋值,导致裂缝和孔洞;用近邻填补孔洞会产生模糊和混叠.**后向算法**,输入f, $\mathbf{h}$, 输出g. For every pixel $\mathbf{x}'$ in $g(\mathbf{x}')$, 1)Compute the source location $\mathbf{x} = \hat{\mathbf{h}}(\mathbf{x}')$, 2)Resample $f(\mathbf{x})$ at location $\mathbf{x}$ and copy to $g(\mathbf{x}')$.

模型拟合与优化

Regularization: $\tilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{m=1}^{M}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2$. 选择合适的 λ 约束权值, 可以避免欠拟合或过拟合.

高斯分布$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}}\exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right\}$, MLE $\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n$, $\sigma_{\mathrm{ML}}^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{\mathrm{ML}})^2$.

In more detail, the interpolated function f is a weighted sum (or **superposition**) of basis functions centered at each input data point $\mathbf{f}(\mathbf{x}) = \sum_k \mathbf{w}_k\, \phi(\|\mathbf{x} - \mathbf{x}_k\|) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathrm{x})$ where the $\mathbf{x}_k$ are the locations of the scattered data points, the ϕ are the **radial basis functions** (or kernels), and $\mathbf{w}_k$ are the local **weights** associated with each kernel. 也即 $\mathbf{w} = (w_1, w_2, \cdots w_N)^T$为基函数权重. $\boldsymbol{\phi}(\mathrm{x}) = (\phi(\|\mathrm{x} - \mathrm{x}_1\|), \phi(\|\mathrm{x} - \mathrm{x}_2\|), \cdots, \phi(\|\mathrm{x} - \mathrm{x}_N\|))^T$ 为数据点附近的径向基函数. 拥有该特性的函数可作为径向基函数: $\phi(\|x\|) = \phi(\mathrm{x})$. 即函数取值仅取决于径向距离. 如 Gaussian:$\phi(r) = \exp(-r^2/c^2)$. 多二次基函数$\phi(r) = \sqrt{r^2 + c^2}$. 逆二次基函数: $\phi(r) = \frac{1}{\sqrt{r^2+c^2}}$. 薄板样条函数: $\phi(r) = r^2 In(r)$.

If we want our function to exactly interpolate the data values, we solve the linear system of equations $\mathbf{f}(\mathbf{x}_k) = \sum_l \mathbf{w}_l\, \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) = \mathbf{d}_k$ to to obtain the desired set of weights $\mathbf{w}_k$. 但权值$\mathbf{w}$的求解条件可能是病态的: 数据位置或数据值的微小改变可能引起插值函数的巨大变化.正则化: $E(\{w_k\}) = E_{\mathrm{D}} + \lambda E_{\mathrm{W}} = \sum_k \|\sum_l \mathbf{w}_l\, \phi(\|\mathbf{x}_k - \mathbf{x}_l\|) - \mathbf{d}_k\|^2 + \lambda\sum_k \|\mathbf{w}_k\|^p$. $p = 2$时, 得到纯最小二乘问题, 可以用 normal equations 求解:$\mathbf{w} = (\Phi^T\Phi + \lambda I)^{-1}\Phi^T D$. $p = 1$时, 称为 Lasso 正则化。权重项趋近于零, 使用较为稀疏的基函数集合。

稳健的数据拟合. 比起使用二次惩罚, 我们可以使用一个鲁棒损失函数 ρ, $E_{\mathrm{R}} = \sum_k \rho(\|\mathbf{r}_k\|)$, 其中$\mathbf{r}_k = \mathbf{f}(\mathbf{x}_k) - \mathbf{d}_k$.该损失函数给较大的数据拟合误差赋予较低的权重.**过拟合**: 增加训练集数据; 降低模型复杂度; 增加正则化约束项; 集成学习; Dropout、BatchNorm; early stopping. 欠拟合: 增加特征数; 增加模型复杂度; 减小正则化系数.

深度学习

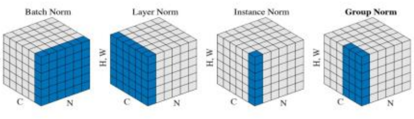
感知器: Minsky 证明感知器本质上是一种线性模型, 只能处理线性分类问题, 就连最简单的 XOR(异或)问题都无法正确分类. 可以使用二层感知器来解决.一种思路$y_1 = \mathrm{ReLU}(x_1 - x_2)$, $y_2 = \mathrm{ReLU}(x_2 - x_1)$, $y_3 = \mathrm{ReLU}(y_1 + y_2)$. 或者$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$, 则用两层感知器处理这个问题: $A \wedge^{-} B$, $\mathrm{sgn}_1(f_1(A,B))$,$f_1(A,B) = w_0 + w_1 A + w_2 B$. $A \oplus B$: $\mathrm{sgn}_{xor}(f_{xor}(A,B))$, $f_{xor}(A,B) = w_0 + w_1\mathrm{sgn}_1(f_1(A,B)) + w_2\mathrm{sgn}_1(f_1(B,A))$.没有隐藏层的网络在它们可以学习建模的输入输出映射方面非常有限.更多的线性单元层并没有帮助, 它仍然是线性的;固定输出的非线性是不够的.

激活函数: ReLU: 线性整流函数. $\max(0,x)$. Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$. tanh: $\tanh(x)$. Leaky ReLU: $\max(0.1x, x)$.
Softmax function:$p(C_k \mid \mathrm{x}) = \frac{p(\mathrm{x}|C_k)p(C_k)}{\sum_{j=1}^{K} p(\mathrm{x}|C_j)p(C_j)} = \frac{\exp(a_k)}{\sum_{j=1}^{K}\exp(a_j)}$.
求导: $\nabla_{\mathbf{x}}\mathbf{A}\mathbf{x} = \mathbf{A}^\top$, $\nabla_{\mathbf{x}}\mathbf{x}^\top\mathbf{A} = \mathbf{A}$, $\nabla_{\mathbf{x}}\mathbf{x}^\top\mathbf{A}\mathbf{x} = (\mathbf{A} + \mathbf{A}^\top)\mathbf{x}$
若$y = \boldsymbol{A}X$, 则$\frac{\partial y}{\partial \boldsymbol{A}} = x^\top$. $\tanh'(x) = 1 - \tanh^2(x)$. $\mathrm{sigmoid}(x) = \mathrm{sigmoid}(x)\cdot(1 - \mathrm{sigmoid}(x))$
$f(x) = \frac{1}{1+e^{-x}}$, $f'(x) = \frac{e^{-x}}{(1+e^{-x})^2}$; $g(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$; $g'(x) = \frac{4}{(e^x + e^{-x})^2}$

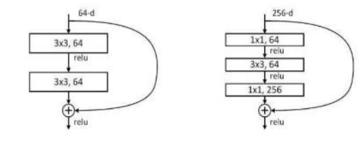
梯度下降: BGD: $L(W) = \frac{1}{N}\sum_{i=1}^{N} L_i(x_i, y_i, W) + \lambda R(W)$; $\nabla_W L(W) = \frac{1}{N}\sum_{i=1}^{N}\nabla_W L_i(x_i, y_i, W) + \lambda\nabla_W R(W)$.影响因素: 权值初始化方式、步数、学习率、batch size、data sampling; 当 N 很大时, 计算开销大.SGD 可能存在的问题: 如果损失在一个方向上快速变化, 在另一个方向上缓慢变化会怎样? 如果损失函数存在局部最小值或鞍点, 则梯度为 0, 梯度下降卡住. 此外, 我们的梯度来自于小批量数据. 因此可能会有噪声.随机梯度下降 (SGD) 比批量梯度下降 (BGD) 更快地接近最小值, 但它可能永远无法"收敛"到最小值。

优化器: SGD: $x_{t+1} = x_t - \alpha\nabla f(x_t)$;SGD+Momentum: $v_{t+1} = \rho v_t + \nabla f(x_t)$, $x_{t+1} = x_t - \alpha v_{t+1}$;Nesterov:$v_{t+1} = \rho v_t - \alpha\nabla f(x_t + \rho v_t)$,$x_{t+1} = x_t + v_{t+1}$;AdaGrad;RMSprop;Adam 约等于 RMSProp + Momentum. Adam 满足 momentum, adaptive learning rates; leaky second moments; bias correction for moment estimates.

归一化:$\mu_i = \frac{1}{|\mathcal{B}|}\sum_{n\in\mathcal{B}} s_i^{(n)}$,$\sigma_i^2 = \frac{1}{|\mathcal{B}|}\sum_{n\in\mathcal{B}}\left(s_i^{(n)} - \mu_i\right)^2$,$\hat{s}_i^{(n)} = \frac{s_i^{(n)} - \mu_i}{\sqrt{\sigma_i^2 + \epsilon}}$,$y_i = \gamma_i\hat{s}_i + \beta_i$.



数据增强: Distorted; Cutout, Mixup.
CNN 特点一: 稀疏连接; 特点二: 参数共享; 特点三: 等变表示
Pooling 可以分为 Max-pooling(返回池化窗口内最大值);Average pooling; Stochastic pooling.
多通道卷积: 图像有多个颜色通道,各通道分别进行卷积,再把卷积的结果相加. 一个卷积核提取一个特征,多个卷积核产生多个特征,每个特征对应于一个通道.
原始图$N\times N$, 卷积核$m\times m$, 卷积核$(N-m+1)\times(N-m+1)$.
池化: 最大池化, 平均池化, 随机池化.
令 O=输出图像的尺寸, I=输入图像的尺寸, K=卷积层的核尺寸, S=移动步长, P =填充数, 则输出图像尺寸$O = \frac{I-K+2P}{S} + 1$. 严格来说这个尺寸是要**向下**取整的
参数个数: (核长 x 核宽 x **输入通道数**+1)x 输出通道数
AlexNet; VGGNet 大的感受野被 (之间带有 ReLU 的) 连续的 3x3 卷积层所取代, 相似的感受野, 更少的参数, 更多的非线性; GoogLeNet: 1x1 3x3 5x5 convolutions 并联. 具有不同感受野大小和操作的平行路径旨在捕获特征图堆栈中稀疏相关模式. Inception v2, v3 使用批量归一化规范训练, 降低辅助分类器的重要性; ResNet 引入 skip or shortcut connections.



ResNet: 左侧残差结构称为 BasicBlock; 右侧残差结构称为 Bottleneck.其中第一层的 1x 1 的卷积核的作用是对特征矩阵进行降维操作, 将特征矩阵的深度由 256 降为 64.第三层的 1x1 的卷积核是对特征矩阵进行升维操作, 将特征矩阵的深度由 64 升成 256。降低特征矩阵的深度主要是为了减少参数的个数。如果采用 BasicBlock.参数的个数应该是: 256x256x3x3x2=1179648 采用 Bottleneck, 参数的个数是: 1x1x256x64+3x3x64x64+1x1x256x64=69632

识别

图像分类: 输入图片, 输出猫. 单个物体, 没有空间范围. 语义分割, 没有物体, 只有像素, 输入图片, 分割草, 猫, 树, 天空. 目标检测: 输入图片, 用框把狗和猫框住. 实例分割: 输入图片, 把狗和猫用不同颜色覆盖.图像分类: 基于特征的方法: 词袋(Bag of words). 在关键点处提取特征;量化;获取视觉词 (特征聚类中心) 上的直方图;用分类算法如支持向量机在特征分布直方图上学习决策面. 基于部件的模型(Part-based model). 原理: 通过寻找物体的组成部分并测量它们之间的几何关系来识别物体

目标检测: 输入: 单张 RGB 图像; 输出: 检测物体的集合; 每个物体的预测内容:类别标签 (属于已知类别集合);检测框 (Bounding Box)的位置 (x,y) +形状 (width, height). 挑战: 多输出, 多种类输出(需要同时预测 what 和 where), 图像尺寸大.**人脸识别**算法在鲁棒性上需要考虑: 光照、表情、尺度、姿态

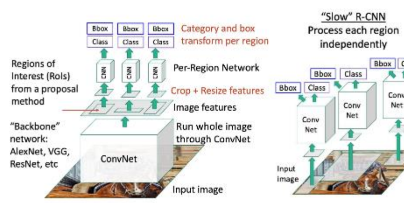
检测单个物体: 将定位看成一个回归问题. 网络在 ImageNet 上预训练(迁移学习), 之后一部分解决"What"的问题, 输出一个 class score, 和 correct label 计算出 softmax loss, 另一部分解决"where"的问题, 计算出 Box Coordinates, 和 Correct Box 计算出一个 L2 Loss. 两个 Loss 进行 Weighted Sum 之后计算出一个 Multitask Loss.

检测多个物体:滑窗 (Sliding Window).将不同的图像块输入卷积神经网络中.网络将每一个图像块分类为物体或背景.问题: 在一张尺寸为 $H\times W$ 的图像中有多少个可能的检测框?假设检测框的大小为 $h\times w$,x 可能的位置: $W - w + 1$,y 可能的位置: $H - h + 1$,可能的位置 $(W-w+1)\times(H-h+1)$. 则所有可能的检测框数: $\sum_{h=1}^{H}\sum_{w=1}^{W}(W-w+1)(H-h+1) = \frac{H(H+1)}{2}\frac{W(W+1)}{2}$.$800\times600$ 大小的图像有约 58M 个检测框! 全部计算是不现实的!

候选区域 (Region Proposals):找到几乎包括所有物体的检测框的集合,通常基于启发式 (heuristics): 如搜索"斑点状"的图像区域,运行速度相对较快: 选择性搜索 (selective search) 在 CPU 上 几秒内可以给出 2000 个候选区域.

R-CNN: Region-based CNN. 输入: 单张 RGB 图像. 1)运行候选区域算法, 计算得到约 2000 个候选区域; 2)将每个区域大小调整为 224x224, 并**分别**输入 CNN 中预测其类别得分和检测框变换参数(Forward each region through ConvNet). 3) 根据分类得分选择候选区域的子集进行输出 (有多种选择: 背景阈值, 类别阈值, 或者输出每张图像得分前 K 个) 4)跟真实的检测框标注进行比较. 问题: 非常慢! 对于每张图像需要约 2000 次的前向计算 (forward pass).解决方案: 在 warping 前经过 CNN

Fast R-CNN



Per-Region 网络是相对轻量级的;大多数计算在骨干网络 (Backbone Network)中进行; 为重叠候选区域 (region proposals)减小了工作量.如: AlexNet 作检测, 5 层卷积层作为 backbone , 2 层全连接层用作 per-region network;ResNet 作检测, 最后一个阶段用作 per-region network; 其余的部分用作 backbone. Fast R-CNN 运行时间被候选区域主导.

Faster R-CNN: 加入 Region Proposal Network (RPN)来从特征中预测候选区域;其他部分跟 Fast R-CNN 相同. Faster R-CNN 是一个 Two-stage object detector. 第一阶段, 每张图运行一次: Backbone Network; Region Proposal Network; 第二阶段: 每个区域运行一次. 裁剪特征: Rol Pool/Align; 预测物体类别; 预测检测框的偏移量. Fast R-CNN 计算时间: PropTime +1 * ConvTime+NumProp * fcTime

Feature Pyramid Networks (FPN). 如何提升多尺度目标的检测效果? Featurized image pyramid: 生成图片金字塔, 对不同尺寸的图片分别提取特征; Single feature map: 在网络的最后一层的特征图上进行预测 (SPPNet, Fast R- CNN, Faster R-CNN 等采用) ; Pyramid feature hierarchy: 从网络不同层抽取不同尺度的特征, 在不同尺度的特 征上分别进行预测 (SSD 采用的多尺度融合方法. FPN: 为了使不同尺度的特征都包含丰富的语义信息, 同时又不使计算成本过高, FPN 采用 bottom-up、top-down 和 lateral connection 的方式, 让底层高分辨 率低语义的特征和高层低分辨率高语义的特征融合在一起, 使得最终得到的不 同尺度的特征图都有丰富的语义信息

目标检测评估: $IoU = \frac{B_{\mathrm{pr}} \cap B_{\mathrm{gt}}}{B_{\mathrm{pr}} \cup B_{\mathrm{gt}}}$, $IoU > 0.5$: 合理; $IoU > 0.7$:好;$IoU > 0.9$:接近完美.

非极大值抑制 (Non-Max Suppression, NMS). 问题: 检测框重叠. 解决方案: 使用非极大值抑制对原始检测结果进行后处理. 算法:1)选择下一个得分最高的检测框; 2)加入阈值条件 IoU > threshold 来消除得分低的检测框; 3)如果还有多余的检测框, 重复 1-2. 问题: 当物体高度重叠时, NMS 可能会消除一些好的检测框.

Mean Average Precision (mAP) 1)在所有测试图像上运行带有 NMS 的目标检测器 2) 对于每一个类别, 计算 Average Precision (AP) = Precision vs Recall 曲线下的面积 a) 对于每个检测结果 (得分从高到低排序) i)如果跟某个 GT 框有 IoU>0.5, 标记为 positive 并消除掉该 GT 框 ii)否则, 标记为 negative iii)画出 PR 曲线上的点 b)Average Precision (AP) = PR 曲线下的面积 3) Mean Average Precision (mAP) = 所有种类 AP 的平均值 4)对于"COCO mAP": 计算每个 IoU 阈值下的 mAP@thresh (0.5, 0.55, 0.6, ..., 0.95) 然后取平均. 计算的时候 $\mathrm{precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$; $\mathrm{recall} = \frac{\mathrm{TP}}{\mathrm{P}}$. 示例: 假设有 3 个 ground truth boxes, 5 个候选框, 第一个候选框有匹配, 则此时 precision 为 1/1, recall 为 1/3; 第二个候选框有匹配, 则 precision 为 2/2, recall 为 2/3; 第三个候选框没匹配, 则 precision 为 2/3, recall 为 2/3; 第四个没匹配, 则 precision 为 2/4, recall 为 2/3; 第五个匹配了, 则 precision 为 3/5, recall 为 3/3.

若有病D^+, 检测阳性T^+, 则对应 TP(真阳性), 检测阴性T^-, 对应 FN 假阴性.
若无病D^-, 检测阳性T^+, 则对应 FP(假阳性), 检测阴性T^-, 对应 TN 真阴性.
Sensitivity 灵敏度: $\mathrm{P}(\mathrm{T}^+ \mid \mathrm{D}^+) = \mathrm{TP}/(\mathrm{TP} + \mathrm{FN})$ 为(1-犯第二类错误概率);
Specificity 特异度: $\mathrm{P}(\mathrm{T}^- \mid \mathrm{D}^-) = \mathrm{TN}/(\mathrm{TN} + \mathrm{FP})$. 为(1-犯第一类错误概率)
False discovery rate 误发现率: $\mathrm{P}(\mathrm{D}^- \mid \mathrm{T}^+) = \mathrm{FP}/(\mathrm{TP} + \mathrm{FP})$
Accuracy 精度/准确率: $(\mathrm{TP} + \mathrm{TN})/(\mathrm{D}^+ + \mathrm{D}^-)$
Other Jargons: Precision: $\frac{\mathrm{TP}}{\mathrm{TP+FP}}$; Recall: $\frac{\mathrm{TP}}{\mathrm{TP+FN}}$; Accuracy: $\frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$
ROC: Receiver Operating Characteristic. 横坐标为 False Positive Rate, 纵坐标为 True

Positive Rate. AUC: Area Under Curve, higher the better, 0.5: random guess.
一阶段检测器 (One-stage detectors):没有候选阶段 (proposal stage);直接输出检测框和相应的类别标签;比大多数两阶段 (two-stage) 的检测器快，但是不如它们表现好.One-stage：直接预测位置与类别.
YOLO (You Only Look Once): 将检测看成回归问题. YOLO 检测网络包括 24 个卷积层和 2 个全连接层，其中卷积层提取图像特征，全连接层用来预测图像位置和类别概率值.输入图像分成 $S\times S$ 个格子，每个格子负责检测落入该格子的物体、输出 B 个 bbox 信息以及 C 个物体属于某种类别的概率信息.Input: an image; Output: a $S\times S\times(B\times 5+C)$ tensor. 在 Training 阶段: compute MSE loss with a target tensor; 在 Test 阶段: Interpret as bboxes and labels
SSD: Single shot MultiBox Detector
FCN: 与 CNN 相比，FCN 有一个显著的特点就是输入图像尺寸不需要裁剪成指定大小，卷积和反卷积都不依赖图像的整体尺寸（CNN 的全连接层需要尺寸固定才能确定权重 weight）。先前的卷积、池化等操作会使输出尺寸变小，在 FCN 的图像语义分割领域中，需要输入输出图像大小一致，这时我们就需要一种扩大图像尺寸实现小分辨率到大分辨率的操作，这种操作被称为上采样（Upsample），反卷积就是常见的一种上采样实现算法。仔细观察整个计算过程，发现整个运算类似卷积运算的反运算（正向卷积运算的输出变成反向的输入，正向的输入变成反向的输出），但这个转置卷积 / 反卷积只能对矩阵的形状做逆向，函数值不能逆向求得。
原先输入一个图片，经过卷积层和全连接层后输出一个不同类别对应的概率，选择概率最大的类“Tabby Cat”. 现在不经过全连接层，最后出现一个 pixelwise prediction，输出一个 tabby cat heatmap. 比如一个 HxW 的图片, conv pool 后变为 H/4 x W/4，再 conv pool 后变为 H/8 x W/8，然后变成三个 H/16 x W/16，然后变成 H/32 x W/32，再经过 unsampling 后变为 HxW，即 pixelwise output+loss.

特征检测与匹配

第一步：提取特征;第二步：匹配特征;第三步：对齐图像.**局部特征**：找到局部对转换不变的特性. 几何不变性：平移，旋转，比例;光度不变性：亮度，曝光. (光度不变性通常被建模为线性变换：缩放+偏移).通用方法:检测：识别兴趣点;描述：提取每个兴趣点周围的向量特征描述符;匹配：确定两个视图中描述符之间的对应关系.什么是**好的角点检测算法**?检测出图像中“真实的”角点;准确的定位性能;很高的重复检测率（稳定性好）;具有对噪声的鲁棒性;具有较高的计算效率.
Harris 角检测：当我们平移 $[u,v]$ 时，中心像素处的强度变化为：$I(x+u,y-v)-I(x,y)$.变换 $[u,v]$ 时，窗口 $w(x,y)$ 的变化：$E(u,v)=\sum_{x,y}w(x,y)[I(x+u,y+v)-I(x,y)]^2$.但这个计算非常缓慢. $O(\text{window width}^2\times\text{shift range}^2\times\text{image width})$.如果运动 $[u,v]$ 很小，则可以进行泰勒级数展开，进行一阶近似.$I(x+u,y+v)\approx I(x,y)+[I_x\quad I_y]\begin{bmatrix}u\\v\end{bmatrix}$. 为什么要取小窗口?因为可能会有噪声，为了平滑.
定义 SSD“误差” $E(u,v)$: $E(u,v)=\sum_{(x,y)\in W}[I(x+u,y+v)-I(x,y)]^2\approx\sum_{(x,y)\in W}\left[I_xu+I_yv\right]^2\approx Au^2+2Buv+Cv^2$. 其中$A=\sum_{(x,y)\in W}I_x^2$; $B=\sum_{(x,y)\in W}I_x I_y$; $C=\sum_{(x,y)\in W}I_y^2$. 因此，$E(u,v)$ 局部近似为二次误差函数 $E(u,v)\approx Au^2+2Buv+Cv^2\approx[u\ v]\begin{bmatrix}A&B\\B&C\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}\approx[u\ v]H\begin{bmatrix}u\\v\end{bmatrix}$. 其中 H 为 2×2 矩阵，由$H=\sum_{x,y}w(x,y)\begin{bmatrix}I_x^2&I_xI_y\\I_xI_y&I_y^2\end{bmatrix}$. 将 H 对角化:$H=R^{-1}\begin{bmatrix}\lambda_1&0\\0&\lambda_2\end{bmatrix}R$. 椭圆的轴长由特征值决定，方向由$R$决定方向. 最快变化方向对应$(\lambda_{\max})^{-1/2}$，最慢变化方向对应$(\lambda_{\min})^{-1/2}$.
SSD 表面: $\sum\nabla I(\nabla I)^T$.纹理区域：λ_1,λ_2 都比较大，梯度不同，幅值较大;边缘：λ_1 较大，λ_2 较小，大梯度;均匀区域：较小 λ_1，较小 λ_2，梯度的幅值较小.我们可以用 H 的特征值进行图像点分类.定义角响应函数:$R=\det(\mathbf{A})-\alpha\,\mathrm{trace}(\mathbf{A})^2=\lambda_0\lambda_1-\alpha(\lambda_0+\lambda_1)^2$, with $\alpha=0.06$. 性质：R 只与 H 的特征值有关;角点：R 为大数值正数;边缘：R 为大数值负数;平坦区：R 为小数值. α 越小角点越多
哈里斯角检测: 1)计算每个图像窗口的 H 矩阵，得到它们的角分数 2)找到周围窗口给出大角响应的点 (>阈值) 3)取局部极大值点，即执行非极大值抑制
我们希望角的位置对光度变换是不变的，对几何变换是协变的. 不变性(Invariance):图像被变换了,而角点的位置不改变;协变性(Covariance):如果我们有相同图像的两个变换版本,应该在相应的位置检测到特征. 仿射强度变化，对图像强度变化的部分不变性.
对于平移角的位置是协变的，对于旋转角的位置是协变(二阶矩椭圆旋转，但它的形状(即特征值)保持不变),角的位置不是缩放的协变(放大后，所有的点将被分类为边).随尺度增加，哈里斯角点检测的性能下降.
尺度不变检测: 在区域(圆)上设计一个函数，该函数是“尺度不变的”:对于相应的区域，即使它们在不同的尺度上也具有相同的值.例如:平均强度。对于相应的区域(即使大小不同)，它将是相同的.给定一个图像中的一个点，我们可以把它看作是区域大小(圆半径)的函数. 常见的方法：取该函数的局部最大值; 重要的是：这个比例不变的区域大小在每个图像中都是独立的; 观察：得到最大值的区域大小应与图像尺度协变.
MOPS(Multiscale Oriented PatcheS descriptor):在检测到的特征周围选取 40x40 的方形窗口;缩放到 1/5 大小(使用预滤波);旋转到水平; 以特征为中心的 8x8 方形窗口进行采样;通过减去平均值，除以窗口中的标准差来规范化窗口.
DoG 特征检测 ＋ SIFT 描述子. 迄今使用最为广泛的一种特征.**特征点检出**主要是用了 DoG，就是把图像做不同程度的高斯模糊 blur，平滑的区域或点肯定变化不大，而纹理复杂的比如边缘，点，角之类区域肯定变化很大，这样变化很大的点就是特征点。当然为了找到足够的点，还需要把图像放大缩小几倍(Image Pyramids)来重复这个步骤找特征点。**特征点描述**就是一个简单版的 HOG，即以检出的特征点为中心选 16x16 的区域作为 local patch，这个区域又可以均分为 4x4 个子区域，每个子区域中各个像素的梯度都可以分到 8 个 bin 里面，这样就得到了 4x4x8=128 维的特征向量。特征点检出以后还需要一个很重要的步骤就是归一化，计算这个 patch 的主方向，然后根据这个主方向把 patch 旋转到特定方向，这样计算的特征就有了**方向不变性**，也需要根据 patch 各像素梯度大小把 patch 缩放到一定的尺度，这样特征就有了**尺度不变性**。
SIFT 特征的性质:不变性:对图像的旋转和尺度变化具有不变性;对三维视角变化和光照变化具有很强的适应性;局部特征，在遮挡和场景杂乱时仍保持不变性;辨别力强(特征之间相互区分的能力强，有利于匹配);数量稠密(一般 500×500 的图像能提取出约 2000 个特征点);良好的扩展性
SIFT 特征提取算法流程: 1)输入图像 2)多尺度空间极值点检测 3)关键点的精确定位 4)关键点的主方向计算 5)描述子的构造 6)特征向量.其中“多尺度空间极值点检测”与“关键点的精确定位” 为 SIFT 算法在 DoG 尺度空间中提取极值点并进行优化从而获取特征点.
HoG 构建过程: 由梯度方向和梯度强度构建梯度直方图.梯度方向将会取绝对值，因此梯度方向的范围是 0- 180 度，分成 9 个箱（bin） ，分别是 0，20，40...160.因为蓝圈的方向是 80 度 ，大小是 2，所以该点就分到 80 这个箱.因为红色圈圈的方向是 10，大小是 4，因为 10 距 离 0 点和距离 20 点都为 10 ，则均分到 0 和 20 箱（每 个分到 2）.区别: HoG 通常用于描述较大的图像区域。SIFT 用于关键点 匹配;SIFT 直方图根据主方向梯度归一化。HoG 则不然;HoG 的梯度使用邻域箱进行归一化.
尺度空间:目的：检测在尺度变化时仍然稳定的特征.定义图像 $I(x,y)$ 的尺度空间: $L(x,y,\sigma)=G(x,y,\sigma)*I(x,y)$,其中$G(x,y,\sigma)=\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2}$.尺度参数为 σ，当 σ 连续变化时，$G(x,y,\sigma)$ 构成图像的尺度空间.为了在尺度空间中检测稳定的关键点，构造高斯差分尺度空间. **高斯差分尺度空间**（高斯差分 DoG: Difference of Gaussian）: $D(x,y,\sigma)=\left(G(x,y,k\sigma)-G(x,y,\sigma)\right)*I(x,y)=L(x,y,k\sigma)-L(x,y,\sigma)$. 理由: 1)计算效率高：高斯卷积，减法 2)高斯差分是对尺度归一化 LoG 的一个很好的近似，而尺度归一化的 LoG 空间具有真正的尺度不变性 (Lindegerg 1994) 3)实验比较表明，从尺度归一化 LoG 空间中提取的图像特征的尺度稳定性最好，优于梯度、Hessian 或 Harris 角点函数。
SIFT 细节:基于梯度的描述符来捕获关键点附近的纹理;使用与关键点尺度相关的模糊图像 (本例中为 8x8 像素);在关键点附近取图像梯度;要成为旋转不变量，旋转梯度方向和位置 (通过关键点定向);现在我们已经取消了旋转，并在相对于关键点方向θ的位置表达梯度,我们也可以将整个图像旋转 $-\theta$，但那样会更慢.
SIFT 的形成:创建方向直方图数组 (以 4x4 数组为例);将旋转的梯度放入它们的局部方向直方图中;SIFT 作者发现，最好的结果是每个直方图有 8 个方向箱， 16X16 块和 4x4 直方图阵列.
SIFT 描述符分析每个直方图 8 个方向箱，一个 4x4 直方图数组，产生 8 x 4x4 = 128 个数字;因此，SIFT 描述符是一个长度为 128 的向量，它对旋转 (因为我们旋转了块)和缩放 (因为我们使用了来自 DoG 的缩放图像)是不变的;我们可以将图像 A 中的每个向量与图像 B 中的每个向量进行比较，以找到匹配的关键点;描述符向量之间的欧式“距离”是一个很好的相似性度量.
SIFT 对光照变化的鲁棒性:描述符是由梯度（像素之间的差异）组成的，因此它已经对亮度变化具有不变性（例如，将所有图像像素加 10 会产生完全相同的描述符）;更高对比度的照片将线性增加梯度的幅度。要校正对比度变化，可以将直方图归一化（缩放到幅度 = 1.0）;非常大的图像梯度通常来自不可靠的 3D 照明效果（眩光等）。为了减少它们的影响，可以将向量中的所有值限制为 ≤0.2（数值可根据实验调整），然后再次归一化向量。所得到的向量将对光照变化基本保持不变

特征匹配: 给定 I_1 中的一个特征，如何在 I_2 中找到最佳匹配?定义比较两个描述符的距离函数;测试 I_2 中的所有特征，找出距离最小的特征. 如何定义两个特征 f_1 和 f_2 之间的差异?简单方法: L2 距离: $\|f1-f2\|$.如何更好地定义两个特征之间的区别? ratio distance $=\|f_1-f_2\|/\|f_1-f_2'\|$:在 I_2 中，f_2 是 f_1 的最佳误差平方和 (SSD) 匹配;在 I_2 中，f_2' 是 f_1 的第二个最佳 SSD 匹配;为不明确的匹配赋予大值.
单应性变换: 给定 p 和 p' 求解单应性变换 H，也即求解以下形式的方程: $wp'=Hp$.线性未知数: w 和 H 的系数;H 在任意尺度因子下定义;解出 H 需要多少对匹配点?
求解单应性变换:对一对点: $\begin{bmatrix}x_i'\\y_i'\\1\end{bmatrix}\cong\begin{bmatrix}h_{00}&h_{01}&h_{02}\\h_{10}&h_{11}&h_{12}\\h_{20}&h_{21}&h_{22}\end{bmatrix}\begin{bmatrix}x_i\\y_i\\1\end{bmatrix}$, 也即$x_i'=\frac{h_{00}x_i+h_{01}y_i+h_{02}}{h_{20}x_i+h_{21}y_i+h_{22}}$,$y_i'=\frac{h_{10}x_i+h_{11}y_i+h_{12}}{h_{20}x_i+h_{21}y_i+h_{22}}$. 这是非线性的. 写成矩阵形式:

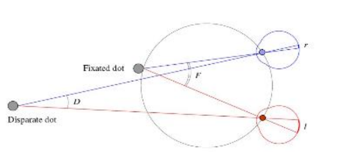
$$\begin{bmatrix}x_1&y_1&1&0&0&0&-x_1'x_1&-x_1'y_1&-x_1'\\0&0&0&x_1&y_1&1&-y_1'x_1&-y_1'y_1&-y_1'\\&&&&&\vdots&&&\\x_n&y_n&1&0&0&0&-x_n'x_n&-x_n'y_n&-x_n'\\0&0&0&x_n&y_n&1&-y_n'x_n&-y_n'y_n&-y_n'\end{bmatrix}\begin{bmatrix}h_{00}\\h_{01}\\h_{02}\\h_{10}\\h_{11}\\h_{12}\\h_{20}\\h_{21}\\h_{22}\end{bmatrix}=\begin{bmatrix}0\\0\\\vdots\\0\\0\end{bmatrix}$$

定义最小二乘问题: $\text{minimize}\ \|\mathbf{Ah}-\mathbf{0}\|^2$.由于 $\mathbf{h}$ 只是按尺度定义的，因此只需求解单位向量 $\hat{\mathbf{h}}$;解: $\hat{\mathbf{h}}=\mathbf{A}^T\mathbf{A}$ 具有最小特征值的特征向量;$n\ge 4$ (至少四对点).注意到，单应性矩阵 H 与 aH 其实完全一样. 所以单应性矩阵 H 虽然有 9 个未知数，但只有 8 个自由度。在求 H 时一般添加约束 $h_{33}=1$，所以还有 $h_{11}\sim h_{32}$ 共 8 个未知数。由于一组匹配点对应 2 组方程，那么只需要 $n=4$ 组不共线的匹配点即可求解 H 的唯一解. 涉及到的优化问题: 最小化$\mathbf{x}^T\mathbf{A}^T\mathbf{Ax}$ s.t. $\mathbf{x}^T\mathbf{x}=1$，也即求解$\mathbf{Ax}=\mathbf{0}$的非平凡最小二乘解. 解法: $[\mathbf{v},\lambda]=\mathrm{eig}(\mathbf{A}^T\mathbf{A})$，$\lambda_1<\lambda_{2..n}$:$\mathbf{x}=\mathbf{v}_1$.

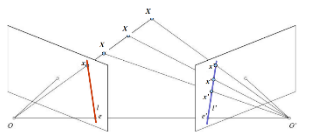
边缘与直线检测

图像边缘: 像素值的快速变化位置.边缘的来源:法线不连续;深度不连续;颜色不连续;光照不连续.如何检测边缘?边缘性质: 边缘是像素值的快速变化位置，则其导数值很大;对不同位置像素点求其对应导数值，选取导数值大的点;对离散图像求导，必须使用离线差分;求导公式: $f'(a)=\lim_{h\to 0}\frac{f(a+h)-f(a)}{h}$.**一维离散导数**: $\frac{df}{dx}=\lim_{\Delta x\to 0}\frac{f(x)-f(x-\Delta x)}{\Delta x}=f'(x)$, $\frac{df}{dx}=\frac{f(x)-f(x-1)}{1}=f'(x)$. 后向: $\frac{df}{dx}=f(x)-f(x-1)=f'(x)$; 前向: $\frac{df}{dx}=f(x)-f(x+1)=f'(x)$; 中心: $\frac{df}{dx}=f(x+1)-f(x-1)=f'(x)$.
二维离散导数:给定函数 $f(x,y)$，其梯度为: $\nabla f=\left[\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right]$，其中$\frac{\partial f}{\partial x}=f(x+1,y)-f(x,y)$，梯度方向 (gradient direction):$\theta=\tan^{-1}\left(\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x}\right)$.
边缘强度由梯度大小 (gradient magnitude)决定！$\|\nabla f\|=\sqrt{\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2}$
离散梯度滤波器对噪声的反应强烈:图像噪声会导致像素与相邻像素差异变大，无法确定真正的边缘像素;通常噪声信号越大，反应越强烈.如何解决这一问题呢？图像平滑（去噪）：迫使噪声像素更接近相邻的像素.通过寻找 $\frac{d}{dx}(f*g)$ 的峰值找到边缘位置.这个定理有一个非常有用的性质: $\frac{d}{dx}(f*g)=f*\frac{d}{dx}g$.
高斯函数导数（Derivative of Gaussian, DOG）. 二维高斯核乘上 $[1\quad 0\quad -1]$便获得了x方向导数，在平面上画图是山形的一高一低.平滑和定位的权衡较: 强的平滑度可以有效去除噪音，但会模糊边缘.在多个“尺度”寻找边缘，定位效果可能更好.
拉普拉斯（Laplacian）算子是一个二阶微分滤波器. 二阶有限差分: $f''(x)=\lim_{h\to 0}\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$. 则拉普拉斯滤波:1-2-1. 将拉普拉斯算子与高斯平滑结合起来，可以得到高斯拉普拉斯算子（Laplacian of Gaussian, LoG).“零交叉”点表示边缘.零交叉在定位边缘方面更准确，但使用不是很方便.**设计边缘检测器**:最优 “边缘检测器 ”的设计标准:良好的检测：最优检测器必须将假阳性（检测到由噪声引起的假边缘）和假阴性（丢失真正的边缘）的概率降到最低;良好的定位：检测到的边缘必须尽可能地接近真实的边缘;单一响应：检测器必须对每个真实边缘点只返回一个 点；也就是说，使真实边缘周围的局部最大值的数量 最小化.
Sobel 操作子:使用两个 3×3 的内核与原始图像进行卷积，计算出导数的近似值；一个用于检测水平变化，另一个用于检测垂直变化。$\mathbf{G}_x=\begin{bmatrix}+1&0&-1\\+2&0&-2\\+1&0&-1\end{bmatrix}$,$\mathbf{G}_y=\begin{bmatrix}+1&+2&+1\\0&0&0\\-1&-2&-1\end{bmatrix}$，事实上，可以将 Sobel 操作拆分成平滑+差分:1,2,1 和 1,0,-1 而幅度$\mathbf{G}=\sqrt{\mathbf{G}_x^2+\mathbf{G}_y^2}$,$\mathbf{\Theta}=\mathrm{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$.
Sobel 滤波的问题:定位能力差 (多个相邻像素都会触发响应);阈值倾向于某些方向斜边:倾斜边缘比水平或垂直边缘更容易丢失;假阴性(丢失真实边缘).
Canny 边缘检测算法流程: 1)高斯平滑：将图像与高斯滤波器进行卷积，以减少噪声并消除小的波动 2)梯度计算：使用 Sobel, Prewitt 或其他梯度算子计算平滑图像的梯度方向和大小 3)非极大值抑制：通过对比梯度幅值，仅保留梯度幅度中的局部最大值，使得边缘细化 4)双阈值判别：根据阈值和梯度幅值关系，将边缘分类为强边缘、弱边缘或非边缘 5)边缘连接：将弱边缘连接到强边缘以形成连续边缘线**高斯平滑**：将图像与高斯滤波器进行卷积，以减少噪声并消除小的波动高斯核大小为 (2k+1) X (2k+1)的计算公式: $H_{ij}=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{\left(i-(k+1)\right)^2+\left(j-(k+1)\right)^2}{2\sigma^2}\right);1\le i,j\le(2k+1)$. 高斯核的尺寸越大，对噪声的灵敏度就越低.高斯核大小 σ 的影响:大 σ 检测大尺度的边缘;小 σ 检测精细特征**非极大值抑制 (Non-Maximum Suppression, NMS)**: 通过对比梯度幅值，仅保留梯度幅度中的**局部最大值**，使得边缘细化。非极大值抑制流程:If $q>p,q>r$，且 p,q,r 在同一梯度方向上;Then，保留 q, set $p=0,r=0$ (将这两个像素点置为黑色，细化白色圆弧边缘)**双阈值判别**：非最大值抑制的结果并不完美，边缘像素是离散的点，并且存在一些噪点。使用单一阈值确定边缘的效果并不理想。双阈值判别流程:Set 高阈值 T，低阈值t:像素点幅值为 p;If $p>=T$，则 set p 为强边缘 (是边缘线的概率很大);If $T>p>=t$，则 set p 为弱边缘 (不确定这些是否是边缘线);If $p<t$, set p 不是边缘.**边缘连接**：对弱边缘进一步排除，确定哪些弱边缘是实际的边缘。边缘确定准则：能连接到强边缘的弱边缘将是实际的边缘线，将被保留。未连接到强边缘的弱边缘将被删除。**问题**:不同尺度边缘无联系，无语义，参数敏感. 改进方法: HED 基于深度学习
霍夫变换目的是找到图像中直线的位置;只有在参数方程已知的情况下，霍夫变换才能检测到线、圆和其他结构;可在噪声和部分遮挡的情况下进行检测.优点:概念简单实现容易;能够处理丢失和遮蔽，可以应用与非线条. **缺点**：计算量大（多参数对象），只能查找单一类型;无法确认直线的长度和位置;共线段无法分离. 假设已经对图像进行了边缘检测，**简单的直线检测**:对于每一对边缘像素:计算出一个直线方程,检查其他像素是否满足该方程.然而上述方法的复杂度很大. 对于一个有 N 个边缘像素的图像，复杂度为 $O(N^2)$.**霍夫变换的直线检测**:找到组成直线的像素集合:第一步是将边缘点转换到一个新的空间;以坐标为 (x_i,y_i) 的边缘点为例，通过该点的直线可以表示为: $y_i=a*x_i+b$.对直线族 $y_i=a*x_i+b$，其中 (x_i,y_i) 是常数，(a,b). 以 (a,b) 为变量生成一个新的空间.即，点 (x_i,y_i) 可以转换为 (a,b) 空间的一条直线: $b=-x_i*a+y_i$.而另一个边缘点 (x_2,y_2) 会转换为空间中的另一条直线.(x,y) 空间中的共线点转换为 (a,b) 空间中相交于单个点 (a',b') 的直线. 可以通过在 (a,b) 空间中找到这样的交点 (a',b') 来检测直线.在原空间中得到的直线方程是: $y=a'*x+b'$.将 (a,b) 空间量化为单元格，可以更有效地找到 (a,b) 空间中的交点。具体来说，不再将一个 (x,y) 空间的点转换为显式的直线，而是用转换后的直线对离散单元格进行“**投票**”。假定获得超过一定票数的单元格对应于 (x,y) 空间中的直线.每个 (x,y) 边缘点，对其相应的 (a,b) 空间的直线方程覆盖的单元格进行投票. 找到票数大于阈值的单元格.算法复杂度：正比于边缘点的数量;正比于单元格的数量.**使用极坐标形式表示直线**: $y=a*x+b$，极坐标表示形式: $x*\cos\theta+y*\sin\theta=\rho$. 与 (a,b) 空间不同，(x,y) 空间中的直线在 (ρ,θ) 空间并不是直线。
RANdom SAmple Consensus 随机抽样一致算法:为了避免“外点”（outliers，离群点）的影响，我们希望寻找“内点”（inliers），并且只使用它们来拟合直线;如果选择一个离群值来拟合当前的直线，那么得到 的直线将不会得到其它点的太多支持.重复 k 次迭代: 1)随机选择一组种子点进行模型估计 (例如，一组边缘点); 2)从种子组中计算模型参数 3)计算点到模型的距离，找到模型的内点 4)如果内点数目足够大，在内点上重新计算模型的最小二乘估计.
需要迭代多少次?需要多少次采样? 记 w 是内点 (来自直线的点)的比例; n 为确定模型的点的数目 (对于直线来说是 2); k 为采样次数.一次采样的 n 个点正确的概率: w^n;一次采样的 n 个点失败的概率: $1-w^n$;所有 k 次采样都失败的概率: $(1-w^n)^k$;k 次采样中至少有一次正确的概率: $1-(1-w^n)^k$.我们应选择足够高的 k，使其低于期望的失败率.
改善 RANSAC 估计:RANSAC 从 n 个点的最小样本中计算出它的最佳估计，并使用这个估计将所有数据点划分为内点和外点;可以通过在所有内点的估计来改进这个初始估计 (例如使用标准的最小二乘法）;但这可能会改变内点，所以要用重新分类内点/外点的方式交替拟合.优点:适用于各种模型拟合问题的通用方法;容易实现，容易计算其失败概率.缺点:在计算成本不剧烈增加的情况下，只能处理有限比例的外点（离群值）;而许多实际问题都有很高的异常值比率(有时选择性地 选择随机子集会有所帮助).投票策略，如霍夫变换，可以处理高比例的离群值。

立体视觉



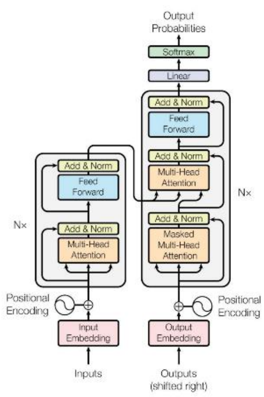
视差: $d=r-l=D-F$.单眼观看时，看上去是随机点；双眼观看时，可以看到三维结构。人类的双眼融合与视网膜没有直接联系，必须涉及中枢神经系统（例如 V2）。高层场景理解对产生立体视觉来说不是必需的
对极几何:



我们如何将第一幅图像中的一个点与第二幅图像中的一个点相匹配？限制我们的搜索空间？关键思想：极线约束. x 的潜在匹配点必须位于相应的直线 l' 上; x' 的潜在匹配点必须位于相应的直线 l 上. 关键概念: 基线: 连接两个相机中心的直线; 极点:基线与图像平面的交点;另一个相机中心的投影. 极平面:包含基线和 X 的平面;极线：极平面和图像平面的交线（成对出现）

Transformer

图像数据的主要特点: Non-Sequential; Continouns; Redundancy(相同或相似的模式);Multi-Scale(同一个目标可能会以不同的尺度出现);Semantic Hierarchy(语义层次结构):不同区域可能会对应不同的物体或场景，因此需要考虑不同语义层次的特征. Natural Signal: 图像通常来源于真实世界. 文本数据: Sequential, Discrete(考虑词语的语义和上下文); Information Density:文本中的不同词语携带不同数量的信息;Long-term Dependency; Recursive(文本可以很容易地表示为递归结构);Polysemy(多义性).
Attention: $X_o=\mathrm{Attention}(Q,K,V)=\mathrm{softmax}\left(\frac{M+QK^T}{\sqrt{d_k}}\right)V$. 参数量为$n^2L^2+n^2Ln$.接下来要做的事情是拿每一个 q 对每个 k 做 attention。attention 是吃两个向量，output 一个分数，告诉你这两个向量有多匹配，至于怎么吃这两个向量，则有各种各样的方法，这里我们采用 Scaled Dot-Product Attention，q^1和k^i做点积，然后除以$\sqrt{d}$。接下来$\alpha_{1,i}$通过一个 softmax layer 得到$\hat{\alpha}_{1,i}$然后拿 $\hat{\alpha}_{1,i}$分别和每一个v^i相乘。即$\hat{\alpha}_{1,1}$乘上v^1，$\hat{\alpha}_{1,2}$乘上v^2，....，最后相加起来。等于v^1到v^4拿$\hat{\alpha}_{1,i}$做加权求和，得到b^1。$b^2=\sum_i\hat{\alpha}_{2,i}v^i$
Self-Attention 的 Complexity per Layer $O(n^2\cdot d)$, Sequential Operations $O(1)$, Maximum Path Length $O(1)$.对比 Recurrent 分别为$O(n\cdot d^2)$，$O(n)$，$O(n)$. Convolutional 为$O(k\cdot n\cdot d^2)$，$O(1)$，$O\left(\log_k(n)\right)$.



Encoder：input 通过一个 input embedding layer 变成一个向量，然后加上位置 encoding 向量.然后进入灰色的 block，这个 block 会重复多次.第一层是 Multi-Head Attention，input 一个 sequence，输出另一个 sequence. 第二层是 Add&Norm.把 Multi-Head Attention 的 input a 和 output b 加起来得到 b’（Add）;把 b’再做 Layer Norm，上图右上方所示为 Batch Norm（行，n 个样本的同一个维度标准化），和 Layer Norm（列，1 个样本的 n 个维度标准化）。一般 Layer Norm 会搭配 RNN 使用，所以这里也使用 Layer Norm;第三层是 Feed Forward，会把 sequence 的每个 b’向量进行处理.第四层是另一个 Add&Norm/
Position encoding 我们希望把 input sequence 的顺序考虑进 self-attention layer 里去。$PE_t=\left[\sin(w_0t),\sin(w_1t)\dots,\sin(w_{i-1}t),\dots,\sin\left(w_{d_{\text{model}}-1}t\right)\right]$，其中 $w_i=\frac{1}{10000^{i/(d_{\text{model}}-1)}}$.在把$x^i$变成$a^i$后，还要加上一个$e^i$(要跟$a^i$的维度相同)
CNN 的 inductive bias 应该是 locality 和 spatial invariance，即空间相近的 grid elements 有联系而远的没有，和空间不变性（kernel 权重共享）Transformer 的 inductive bias 在于 sequentiality 和 time invariance，即序列顺序上的 timesteps 有联系，和时间变换的不变性（权重共享）CNN 的 inductive bias 在于是 locality 和 spatial invariance，即空间相近的 grid elements 有联系而远的没有，和空间不变性（kernel 权重共享）

光流

光流是图像中亮度模式的明显运动.注意:视动可以由没有任何实际运动的光照变化产生;例：球体绕固定光源匀速旋转 vs. 光源绕静止的球体旋转.目标:从光流中恢复出图像中每个像素的运动.如何估计从像素点 $I(x,y,t)$ 到像素点 $I(x,y,t+1)$ 的运动?求解像素点对应问题: 对于给定的像素点 $I(x,y,t)$，在紧邻点中找具有相同颜色的像素点 $I(x,y,t+1)$.
核心假设:颜色一致性(亮度恒定 brightness constancy): 像素点 $I(x,y,t)$ 与像素点 $I(x,y,t+1)$ 的颜色是一致的. 对于灰度图像来说，就是亮度的一致性;小运动(细微运动 small motion): 像素点移动距离较小.空间相干: 相邻点的运动模式相似. 详细解释: 小运动: 图像块的运动随时间是逐渐变化的.空间相干(空间相干性 spatial coherence):场景中的相邻点通常属于同一个表面，因此通常有着相似的运动;因为场景中的相邻点也会投影到图像中的相邻点，因此我们认为图像流是空间相干的.颜色一致:一小块区域中的图像测量（如亮度）是保持不变的，尽管它们的位置可能发生了变化 $I(x,y,t)=I(x+u(x,y),y+v(x,y),t+1)$.如何来决定对应性?块匹配: 方差和 SSD (Sum Squared Differences) $E(u,v)=\sum_{x,y}w(x,y)[I(x+u,y+v)-I(x,y)]^2$.
对灰度图像来说，光流约束:亮度一致性约束: $I(x,y,t)=I(x+u,y+v,t+1)$. 小运动约束: (u 和 v 不超过 1 个像素，或者是光滑的)，I 的泰勒展开为:$I(x+u,y+v)\approx I(x,y)+\frac{\partial I}{\partial x}u+\frac{\partial I}{\partial y}v$.将亮度恒定方程右侧使用泰勒级数线性展开:$I(x+u(x,y),y+v(x,y),t+1)\approx I(x,y,t)+I_x\cdot u+I_y\cdot v+I_t$，则$I_x\cdot u+I_y\cdot v+I_t\approx 0$,$\nabla I\cdot[u\quad v]^T+I_t\approx 0$.能否用以下方程恢复出图像在每个像素点的运动 (u,v)? $\nabla I\cdot[u\quad v]^T+I_t=0$. 问题: 无法测量垂直于梯度（平行于边缘）的流分量
Lucas-Kanade method: $\nabla I(p_i)\cdot[u\quad v]^T+I_t(p_i)=0$，认为相邻的一些点光流一致，列写出总方程，利用最小二乘求解: $\begin{bmatrix}\sum I_xI_x&\sum I_xI_y\\\sum I_xI_y&\sum I_yI_y\end{bmatrix}\begin{bmatrix}u\\v\end{bmatrix}=-\begin{bmatrix}\sum I_xI_t\\\sum I_yI_t\end{bmatrix}$. LK 方法问题：一个点可能与近邻的移动不一致;不具有亮度一致性;运动较大（超过一个像素）:非线性：迭代优化；局部最优：由粗到精计算估计.迭代法：$\vec{v}\leftarrow\vec{v}_{\text{previous}}-\frac{I_t}{I_x}$.
Horn-Schunk 光流法：流 (flow) 表述为应当最小化的全局能量函数;$E=\iint\left[\left(I_xu+I_yv+I_t\right)^2+\alpha^2(\|\nabla u\|^2+\|\nabla v\|^2)\right]\mathrm{d}x\,\mathrm{d}y$;函数的第一部分是亮度一致 (brightness consistency),第二部分是平滑约束，确保像素间的变化是很小的, α 是一个正则化常数，大 α 会得到更平滑的流