机器视觉

考试要点

- [ ] 第一次ppt内容
- [ ] MTF曲线
- [ ] 可分离迭代滤波算法
- [ ] 直线圆检测方法
- [ ] 基于灰度值的模板匹配
- [ ] 广义霍夫变换

第一次ppt内容

一、机器视觉基本概念（重点）

（一）基本目的（任务，能做什么，优点）

提升产量，提高利润，减少缺陷，Track Trace and Control

（二）如何做到

"计量位测码"

1、Measure测量

2、Count计数

3、Decode解码

4、Location定位

5、Inspection检测(缺陷检测)

（三）机器视觉系统基本组成

基于PC的机器视觉

物体，光源，镜头，相机(CCD,CMOS)，PC(算法+通信)

基于Smart Camera

物体，智能相机

长焦：视野窄，物体大

短焦：视野范围广，物体小

MTF曲线

3、MTF值：反应镜头分辨率能力和对比度能力❤️

（1）公式

$$F_{MTF} = \frac{g_1 - g_2}{255}$$

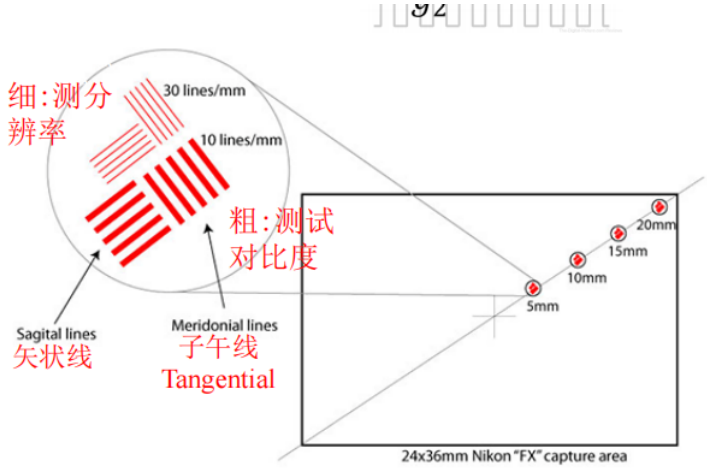
（2）线对对MTF值的影响：频率对MTF的影响

线对越密集MTF一般来说越小

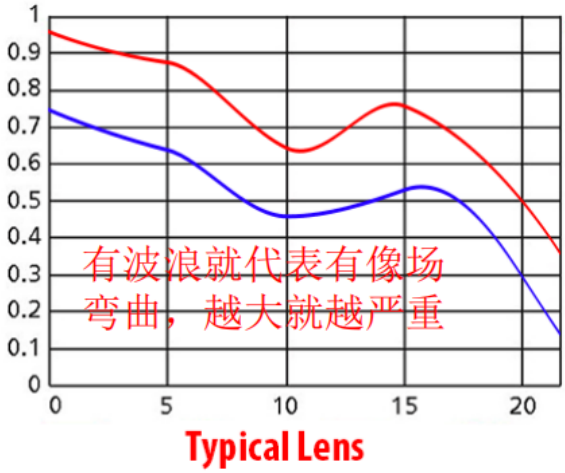
（3）分辨率&对比度测试

粗线用来测对比度，细线用来测分辨率，一般来说MTF>0.5 是可以接受的范围

MTF线弯曲：有场弯曲现象

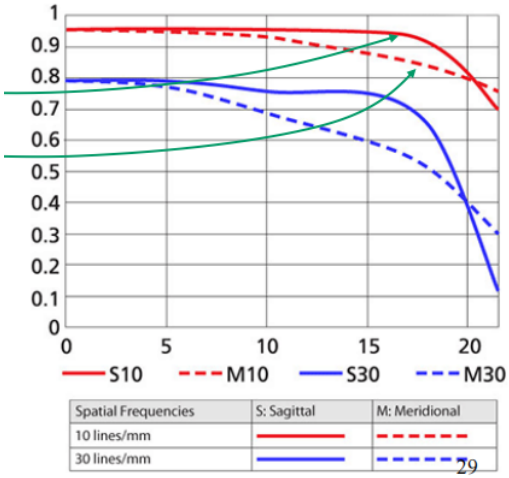


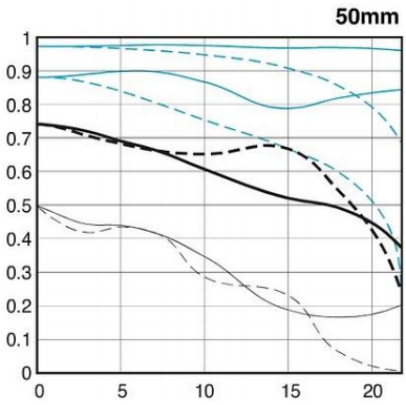
值大于0.9就代表镜头非常优秀，0.7-0.9是优秀，0.5-0.7就是普通，低于0.5就算差了。



对矢状线和子午线进行测量：

虚线实线越接近，代表镜头的色散和色差控制的很好，越背离，表示越严重。





1. 任何线条都是越高越好，下降的趋势越慢越好。
2. 粗线的位置越高，说明该镜头的反差和对比越高。
3. 细线的位置越高，说明该镜头的锐利度越高。
4. 黑线的位置越高，证明该镜头在全开光圈时表现越好。
5. 蓝线的位置越高，证明该镜头在光圈收缩到8的时候表现越好。
6. 蓝线如果和黑线很接近，说明这支镜头表现很出色，只有比较少的镜头有这种表现。
7. 如果蓝线的位置都比较低，说明这支镜头相当差。
8. 0.6以下一般画面就很糟糕了，0.6-0.8画面还可以，0.8-0.9可以算良，0.9以上算非常优秀了，只有优秀镜头在适合的状态下才可能达到，但是具体的情况还是要看各位自己判断了。
9. 实线和虚线越接近，表示这支镜头的焦外成像越柔和自然，反之，差得越远，焦外成像就越差，差太多就会斑斑驳驳一块一块的很难看。

> 图中10线/毫米的曲线越接近1（最大值），镜头的成像对比度就越好。30线/毫米的曲线越接近1，镜头分辨力就越高。
>
> 虚、实两条曲线越接近，说明镜头越能够在如实表现被摄体的同时，更易拍出美丽虚化

> 佳能（中国） - EF镜头 - EF镜头的基础知识 - 解读MTF曲线图 (canon.com.cn)
>
> 佳能，感动常在

可分离迭代滤波算法

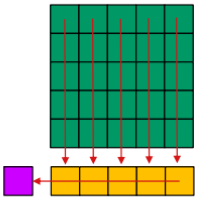
关注Lec7 P47、P51的例题!

（1）可分离滤波器->分离法：kernel分成列向量和行向量，然后先用行向量卷积再用列向量卷积

- Image Smoothing
  - Runtime Complexity of Filters

$$g_{r,c} = \frac{1}{(2n+1)(2m+1)} \sum_{i=-n}^{n} \sum_{j=-m}^{m} \hat{g}_{r-i,c-j}$$

$$g_{r,c} = \frac{1}{(2n+1)(2m+1)} \sum_{i=-n}^{n} \left(\sum_{j=-m}^{m} \hat{g}_{r-i,c-j} \right)$$



Double sum in mean filter (3-14) of complexity $\mathcal{O}(nm)$ is replaced by two sums of total complexity $\mathcal{O}(n+m)$. Consequently, the complexity drops from $\mathcal{O}(whmn)$ to $\mathcal{O}(wh(m+n))$, now only 6,758,400 additions are required.

37,171,200

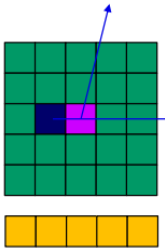
Whenever a filter calculation allows a decomposition into separate row and column sums, the filter is called separable.

（2）可迭代滤波器->迭代法：

- Image Smoothing
  - Runtime Complexity of Filters— recursive filter
    - Let the result of the column sum in Eq.(3.15) be denoted by $t_{r,c}$, then

$$g_{r,c} = \frac{1}{(2n+1)(2m+1)} \sum_{i=-n}^{n} \underbrace{\left(\sum_{j=-m}^{m} \hat{g}_{r-i,c-j} \right)}_{t_{r,c}}$$

$$t_{r,c} = \sum_{j=-m}^{m} \hat{g}_{r,c-j} = t_{r,c-1} + \hat{g}_{r,c+m} - \hat{g}_{r,c-m-1}$$



The sum at position $(r,c)^T$ can be computed based on the already computed sum at position $(r,c-1)^T$ with just two additions.

Whenever a filter can be implemented with this kind of updating scheme based on previously computed values, it is called a recursive filter.

直线圆检测方法

Hough变换

区域特征提取

（一）矩

零阶矩：连通域的面积

$$m_{p,q} = \Sigma r^p c^q$$

一阶矩（归一化的矩）：连通域的亚像素精度中心点

$$n_{p,q} = \frac{\Sigma r^p c^q}{A}$$

中心矩（归一化）：

$$\mu_{p,q} = \frac{1}{A}\sum_{(r,c)^T \in R}(r - n_{1,0})^p (c - n_{0,1})^q$$

二阶中心矩：跟椭圆的长轴短轴(r_1, r_2)，旋转角(θ)有关

> 通过$\frac{r_1}{r_2}$判断是圆还是椭圆

（五）圆形度：更多考虑boundary

$$C = \frac{P^2}{4\pi A}$$

P是区域的周长，A是区域的面积

$$\text{Distance} = \frac{1}{A}\sum ||p - p\_i||$$

$$\text{Sigma}^2 = \frac{1}{A}\sum (||p - p\_i|| - \text{Distance})^2$$

$$\text{Roundness} = 1 - \frac{\text{Sigma}}{\text{Distance}}$$

$$\text{Sides} = 1.4111\left(\frac{\text{Distance}}{\text{Sigma}}\right)^{0.4724}$$

设p为区域中心点（质点），p_i为轮廓上全部像素点，A为轮廓面积. Distance为轮廓上像素点到中心的平均距离，Sigma为轮廓像素点到中心的距离与平均距离的偏差，Roundness则表示平均值与标准差之间的关系.

（六）圆形性

与五合用：先计算圆形性，再计算region的圆形度

Shape factor for the circularity (similarity to a circle) of a region.

$$C = \min(1, C') \quad C' = \frac{A}{(\pi d_{max}^2)} \quad d_{max} = \max(|p - p_i|)$$

d_{max}是区域中心到边界的最远距离

基于灰度值的模板匹配

一、基于灰色值的模板匹配

$$s(r,c) = \mathbf{s}\{t(u,v), f(r+u, c+v); (u,v) \in T$$

当模板和图片完全一样时，相似度测量值为0；不同，相似度测量值大于1。

（一）绝对差异（SAD）和平方差异（SSD）

$$\text{SAD}(r,c) = \frac{1}{n}\sum_{(u,v)\in T} |t(u,v) - f(r+u, c+v)|$$

$$\text{SSD}(r,c) = \frac{1}{n}\sum_{(u,v)\in T} \big(t(u,v) - f(r+u, c+v)\big)^2$$

当光照变化时，上述指标会受到较大的影响，亮度变化时(整体灰度变)即使是相同的形状仍然会返回很大的SAD/SSD误差

==越接近0质量越高==

（二）归一化交叉关联（NCC）

$$\mathrm{NCC}(r,c)=\frac{1}{n}\sum_{(u,v)\in T}\frac{t(u,v)-m_t}{\sqrt{s_t^2}}\cdot\frac{f(r+u,c+v)-m_f(r,c)}{\sqrt{s_f^2(r,c)}}$$

其中，m_t是模板的灰度平均值，s_t^2是模板的方差

$$m_t=\frac{1}{n}\sum_{(u,v)\in T}t(u,v)$$

$$s_t^2=\frac{1}{n}\sum_{(u,v)\in T}(t(u,v)-m_t)^2$$

$m_f(r,c)$和$s_f^2(r,c)$是与图像卷积区域上的均值和方差

$$m_f=\frac{1}{n}\sum_{(u,v)\in T}f(r+u,c+v)$$

$$s_f^2=\frac{1}{n}\sum_{(u,v)\in T}(f(r+u,c+v)-m_f(r,c))^2$$

NCC取值范围为：$-1\le NCC\le 1$

如果NCC值为正负一，则匹配图像的灰度值可以视为模板的线性变换。

（三）提高NCC模板匹配效率的方法

根据SAD值设置停止条件

$$\mathrm{SAD'}(r,c)=\mathrm{SAD}_j'(r,c)+\sum_{i=j+1}^{n}|t(u_i,v_i)-f(r+u_i,c+v_i)|\le nt_s$$

逐行计算SAD（乘以像素点数n），设定差异边界threshold，先计算前j项（防止噪声影响），之后加和，如果超过nt_s则停止匹配。

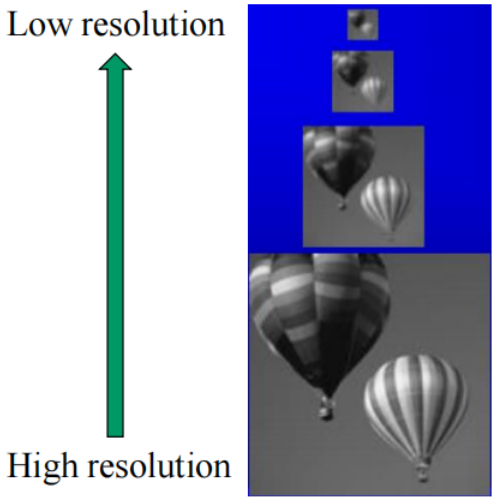
离线计算m_t和$\sqrt{s_t^2}$

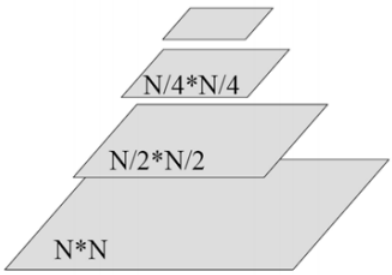
二、图像金字塔

构建匹配模板大小的$\frac{1}{2}$、$\frac{1}{4}$、、、

（一）减小复杂度的原理

先在小Scale图像上模板匹配，得到大致的ROI，传到下一层金字塔得到ROI，进一步模板匹配，由此类推





Space required for pyramids:

$$N^2 + \frac{1}{4}N^2 + \frac{1}{16}N^2 + \frac{1}{64}N^2 = \frac{85}{64}N^2$$

（二）构建图像金字塔的原理

原图不断下采样构成金字塔，越上层的金字塔scale感受野越大，scale越小

1、隔行抽取采样：会丢失一部分信息

2、滤波后采样（不断重复）：高斯核考虑了kernel内邻域像素的信息，随着上采样滤波kernel变为两倍

3、每增加一层层数，图像点和模板点的数量减小4倍，所以每层金字塔的加速速度为16倍。

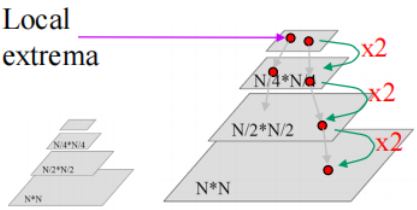
4、层数：最顶层的应该仍然具有一些区分度

（三）分层算法

步骤1：计算模板上的图像金字塔，并使用适当数量的级别搜索图像。

步骤2：在最高的金字塔级别上执行一个完全的匹配。

步骤3：通过将找到的匹配项的坐标乘以2，将较高级别中的匹配项投影到较低的级别。



"backup"

（三）算法备注

在更高的金字塔级别上，我们需要对匹配阈值更加宽松（宽松的），以确保找到所有潜在的匹配。

SAD和SSD相似度度量，我们需要使用略高的阈值。

NCC相似性度量，我们需要在较高的金字塔水平上使用略低的阈值

> *”最顶层阈值参数宽容一些。“*

广义霍夫变换

> ***回顾一下Hough变换***
>
> **霍夫变换就是对于原图上的每一个直线都在参数空间画一条线，最终找出参数空间变换线比较密集的地方在对应回到xy 空间**，就是直线的位置和方向了

> [霍夫变换——神奇的特征提取算法 - 知乎 (zhihu.com)](#)
>
> [9.霍夫变换：圆——介绍、用Hough检测圆、圆的Hough变换1使用hough变换寻找本图中的圆形 Tom Boom的博客-CSDN博客](#)
>
> 理解Hough变换中的Voting思维

（一）边缘点信息

与中心点连线，得到：

1、一个由中心点指向边缘点的向量

2、向量与水平方向的夹角θ

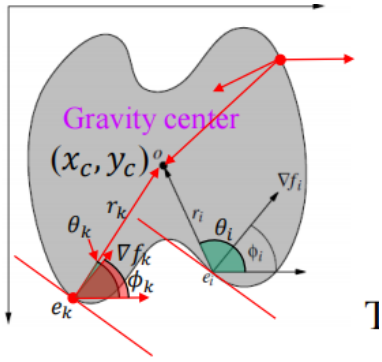
3、梯度方向

（二）R-table：基于模板构建

– At each boundary point, compute the edge location vector: $r = o - e$ and θ

–Store these vectors in a table indexed by gradient orientation (edge location) ϕ

Index	Edge direction		r
0	0	ϕ_0	$\boldsymbol{r}_0^0(r_0,\theta_0), \boldsymbol{r}_1^0(r_1,\theta_1), \cdots$
1	$\Delta\phi$	ϕ_1	$\boldsymbol{r}_0^1(r_0,\theta_0), \boldsymbol{r}_1^1(r_1,\theta_1), \cdots$
2	$2\Delta\phi$	ϕ_2	$\boldsymbol{r}_0^2(r_0,\theta_0), \boldsymbol{r}_1^2(r_1,\theta_1), \cdots$
⋮	⋮		⋮ r_k^i
n	$n\Delta\phi$	ϕ_n	$\boldsymbol{r}_0^n(r_0,\theta_0), \boldsymbol{r}_2^n(r_1,\theta_1), \cdots$



Assuming translation is the only transformation here, i.e., orientation and scale are fixed

（三）算法流程

对test image上的边缘上的每一个点，计算它的Edge direction（梯度方向），在上述R-table内寻找可能的几个(r,θ)数对，计算中心(x_c, y_c)，然后投票

得票数最高的(x_c, y_c)就是所求的中心

$$\begin{cases} x_c = x_i + r_k^i \cos\theta_k^i \\ y_c = y_i + r_k^i \sin\theta_k^i \end{cases}$$

（四）旋转和缩放

投票表变成4维(x_c, y_c, S, α)

$$\begin{cases} x_c = x_i + r_k^i \cos(\theta_k^i + \alpha) \\ y_c = y_i + r_k^i \sin(\theta_k^i + \alpha) \end{cases}$$