CSDN公开课

计算机视觉原理及实战：
3. 图像特征提取与运动估计

主讲人: *Roland*

哈尔滨工业大学

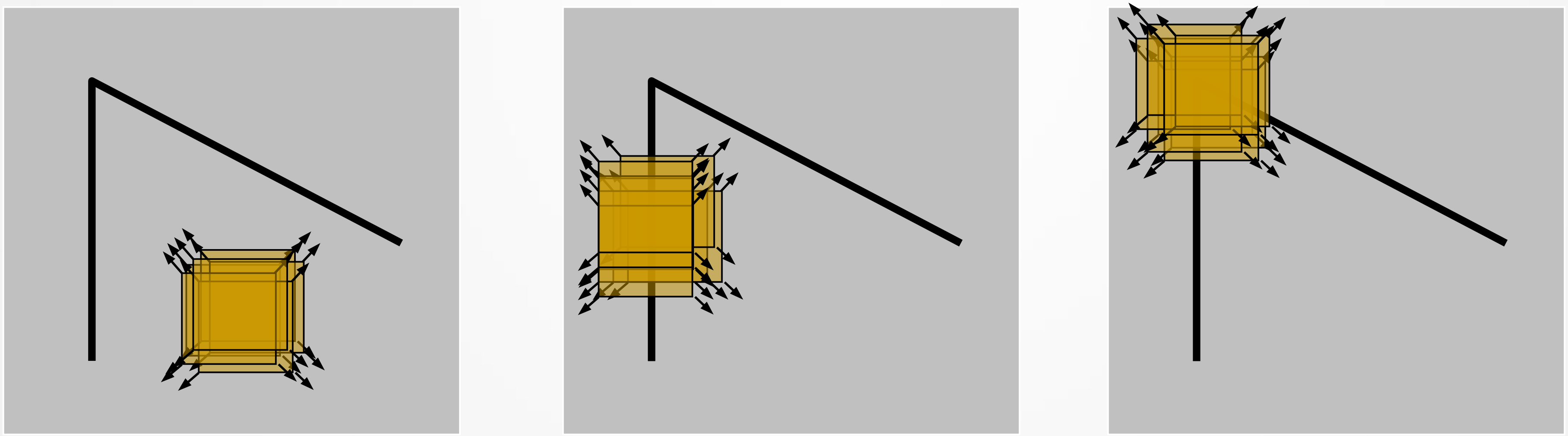
目录
Contents

角点检测

- Harris算子原理
- OpenCV示例

1.1 Harris算子原理

- 在灰度变化平缓区域，窗口内像素灰度积分近似保持不变
- 在边缘区域，边缘方向：灰度积分近似不变，其余任意方向：剧烈变化；
- 在角点处，任意方向均剧烈变化



1.1 Harris算子原理

- 定义灰度积分变化：

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

$$\sum\left[I(x+u,y+v)-I(x,y)\right]^2$$

$$\approx\sum\left[I(x,y)+uI_x+vI_y-I(x,y)\right]^2 \quad \text{First order approx}$$

$$=\sum u^2I_x^2+2uvI_xI_y+v^2I_y^2$$

$$=\sum\begin{bmatrix} u & v\end{bmatrix}\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2\end{bmatrix}\begin{bmatrix} u \\ v\end{bmatrix} \quad \text{Rewrite as matrix equation}$$

$$=\begin{bmatrix} u & v\end{bmatrix}\left(\sum\begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2\end{bmatrix}\right)\begin{bmatrix} u \\ v\end{bmatrix}$$

- 定义灰度积分变化:

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

- 如u, v很小，则有:

$$E(u,v)\cong[u,v]M\begin{bmatrix}u\\v\end{bmatrix}$$

其中

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

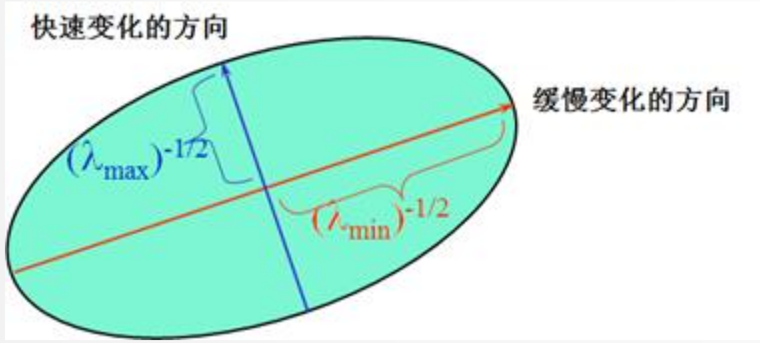
1.1 Harris算子原理

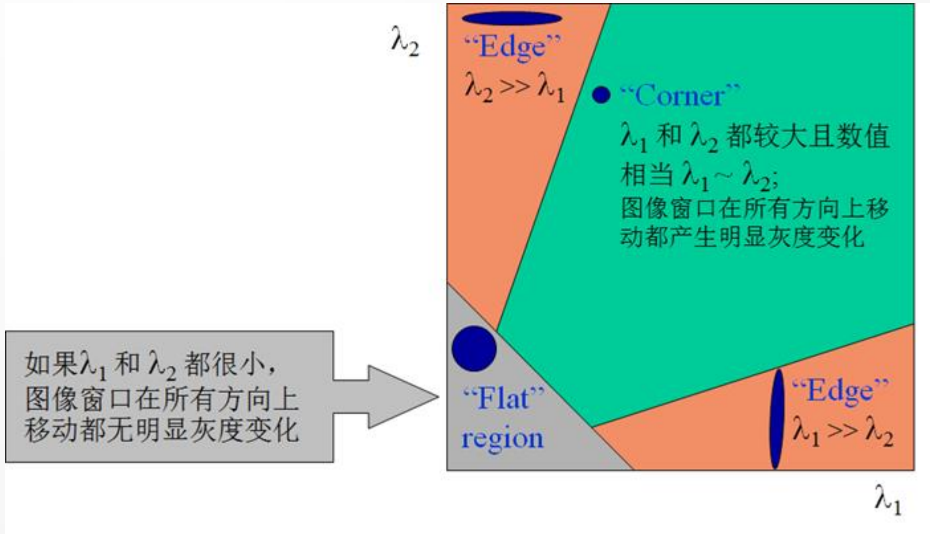
- 令 $E(u,v)=\text{const}$ ，则前面方程表示一个椭圆
- $\lambda_1^{-1/2}$ 和 $\lambda_2^{-1/2}$ 是椭圆的长短轴

当λ_1, λ_2都比较小时，点(x,y)处于灰度变化平缓区域；

当$\lambda_1 >> \lambda_2$或者$\lambda_1 \ll \lambda_2$时，点(x,y)为边界像素；

当λ_1，λ_2都比较大，且近似相等时，点(x,y)为角点；





- 使用角点响应函数：

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

$$\operatorname{trace} M = \lambda_1 + \lambda_2 \qquad \det M = \lambda_1 \lambda_2$$

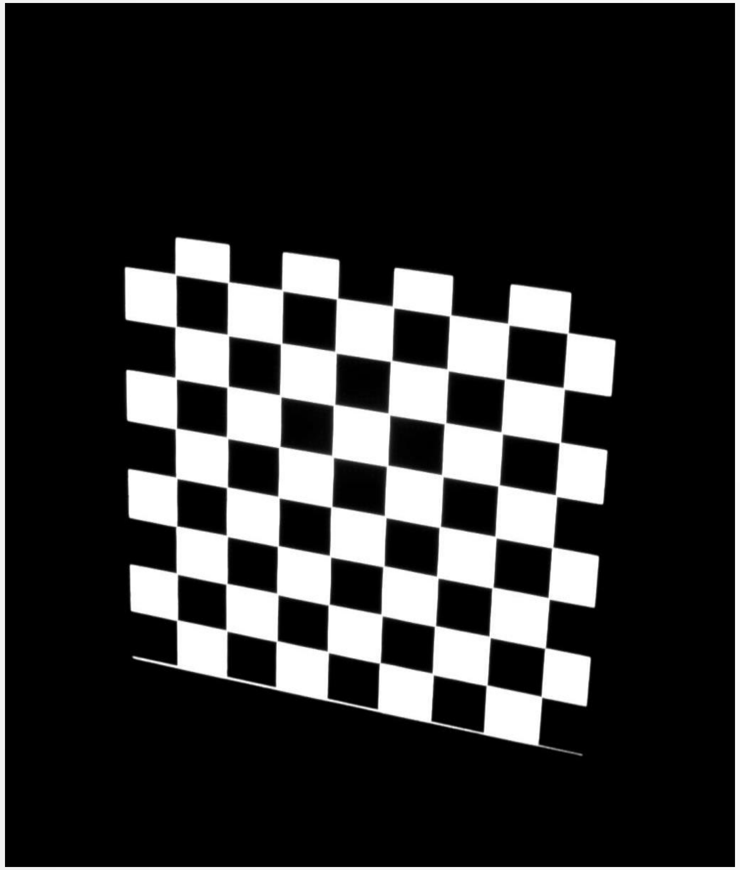
- 当R接近于零时，处于灰度变化平缓区域；
- 当$R<0$时，点为边界像素；
- 当$R>0$时，点为角点。

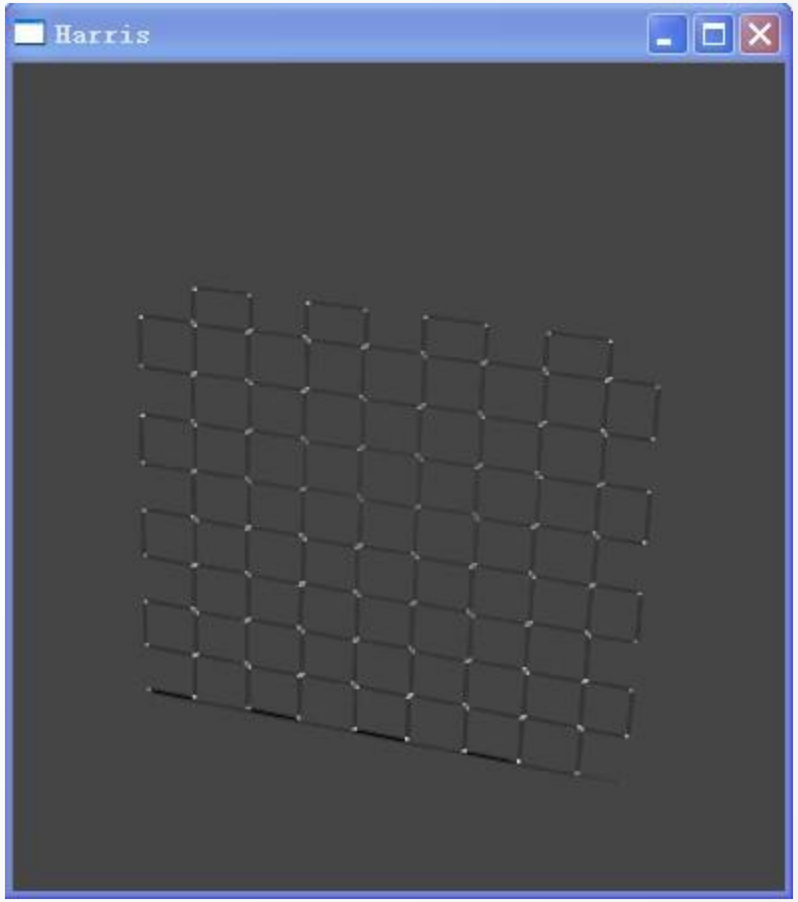
相关函数

```
void cornerHarris(InputArray src, OutputArray dst,
    int blockSize, int ksize, double k, int borderType=BORDER_DEFAULT );
```

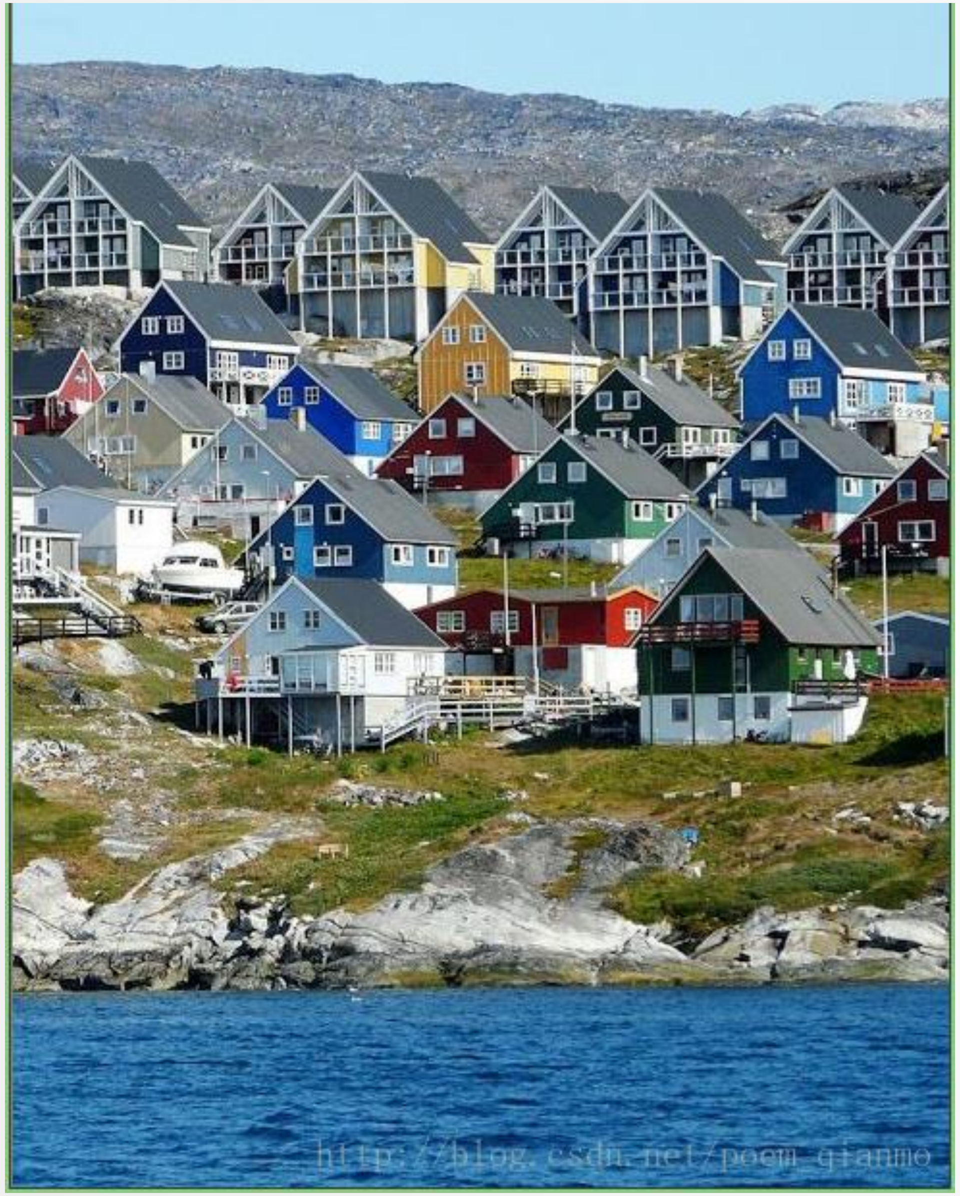
- *src*，输入图像，即源图像，填Mat类的对象即可，且需为单通道8位或者浮点型图像。
- *dst*，函数调用后的运算结果存在这里，即这个参数用于存放Harris角点检测的输出结果，和源图片有一样的尺寸和类型。
- *blockSize*，表示邻域的大小，更多的详细信息在cornerEigenValsAndVecs中有讲到。
- *ksize*，表示Sobel()算子的孔径大小。
- *k*，Harris参数。
- *borderType*，图像像素的边界模式，注意它有默认值BORDER_DEFAULT。更详细的解释，参考borderInterpolate函数。

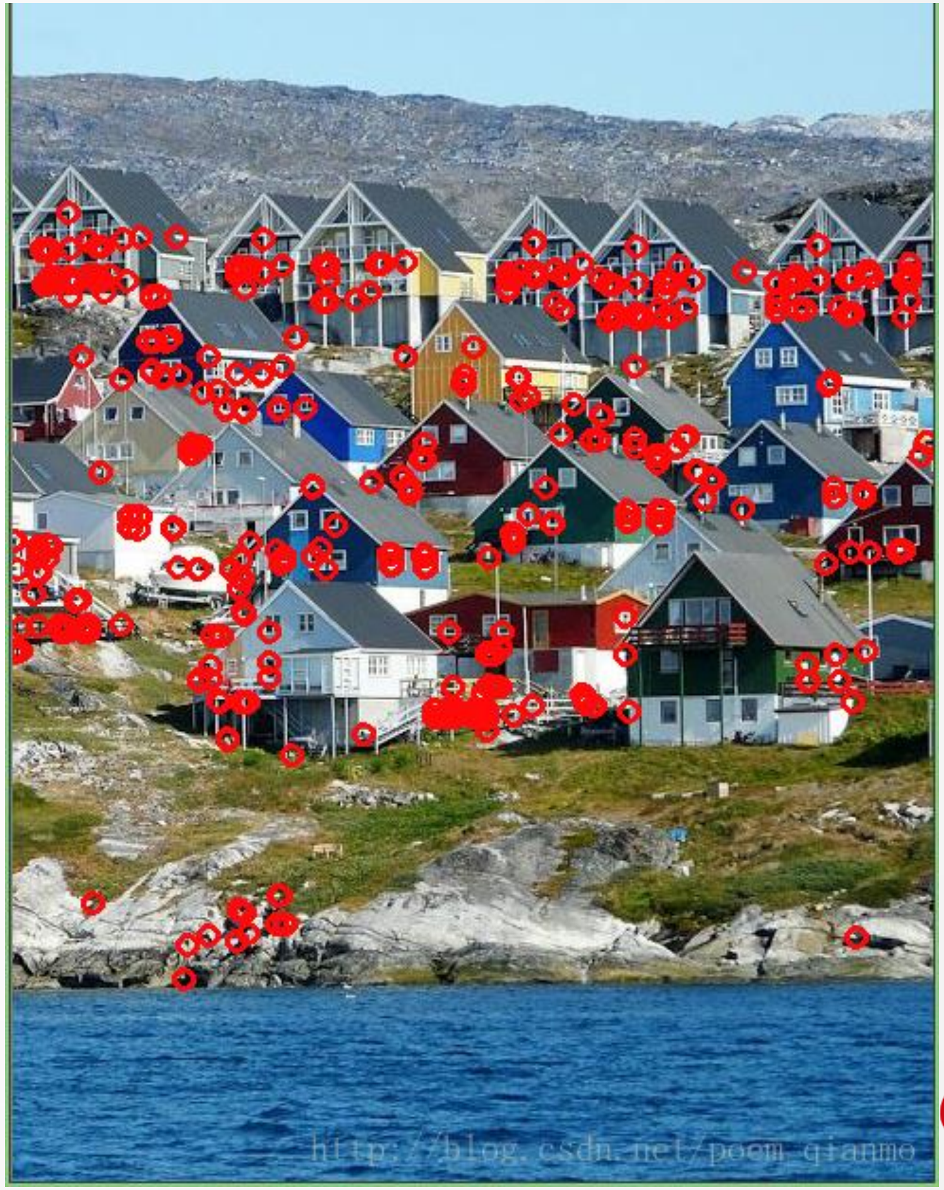
1.2 OpenCV实例





1.2 OpenCV实例


http://blog.csdn.net/poem_qianmo


http://blog.csdn.net/poem_qianmo

背景建模

- 背景建模原理
- 处理实例

相对运动的基本方式

- 相机静止，目标运动——背景提取(减除)
- 相机运动，目标静止——光流估计(全局运动)
- 相机和目标均运动——光流估计

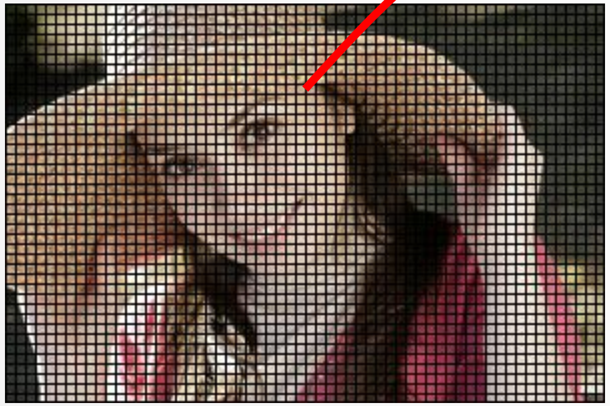
帧差法运动目标检测

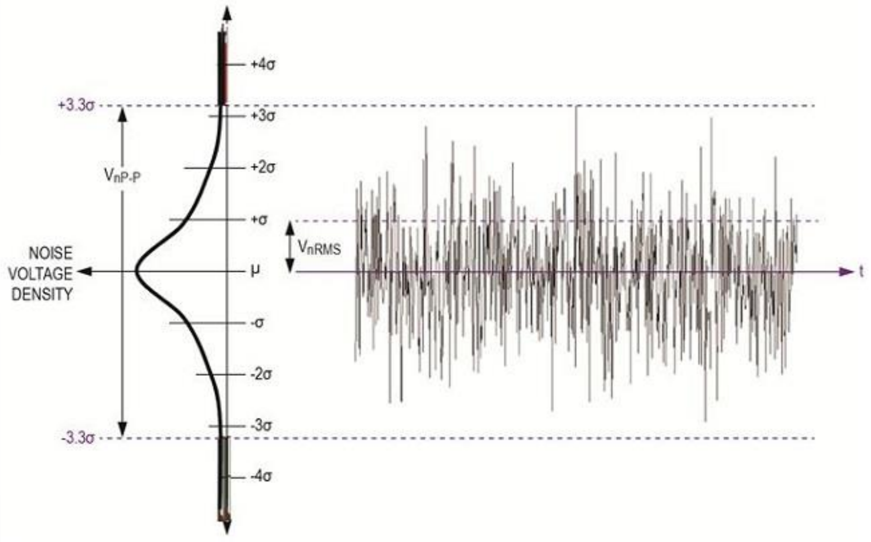
$$D(x,y)=\begin{cases}1;\ 如果 & I(x,y,t)-I(x,y,t-1)>T\\ 0;\ 如果 & 其它\end{cases}$$

- $D(x,y)$: 帧差
- $I(x,y,t)$: 当前帧(t时刻)图像
- $I(x,y,t)$: 上一帧(t-1时刻)图像
- T: 像素灰度差阈值

高斯背景

- 像素灰度值随时间变化符合高斯分布





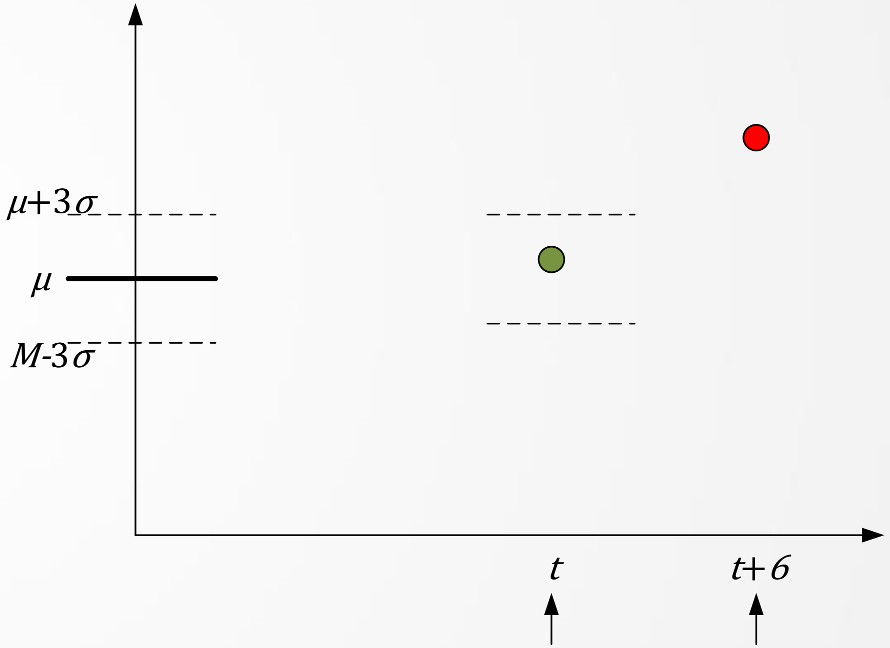
$$I(x,y) \sim \mathcal{N}\left(\mu,\sigma^2\right)$$

$I(30,50)$: 78, 80, 82, 79, 81, 83, 76, ……

如何检测当前图像中某像素属于前景？

$I(x,y,t)-\mu>3\sigma$, 则为前景

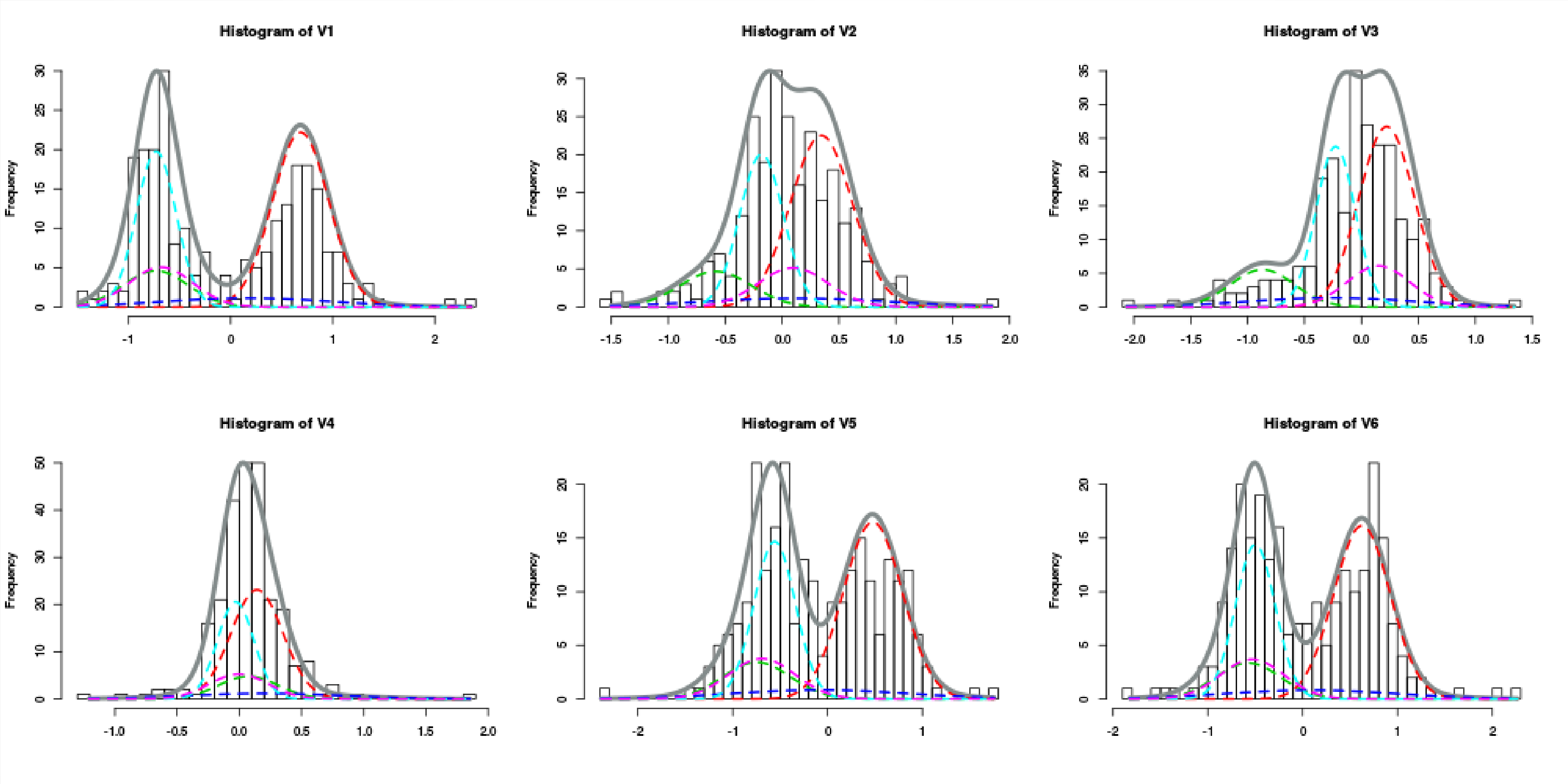
否则 , 背景



2.1 背景建模原理

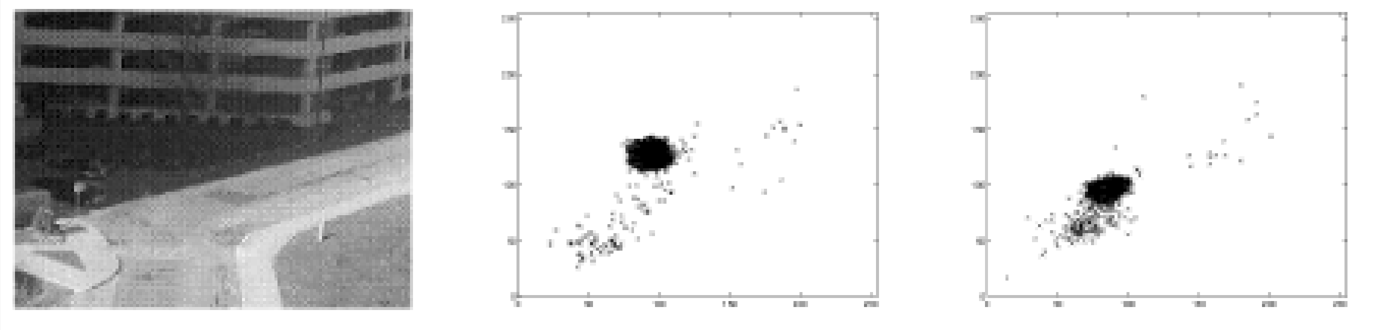
➤ 混合高斯模型

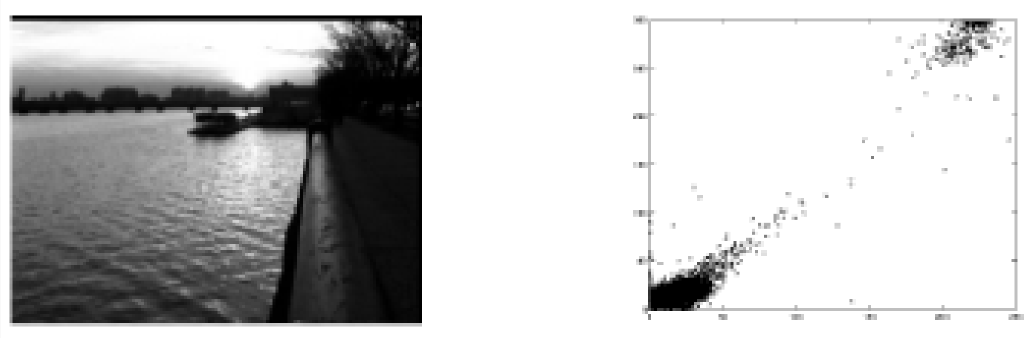
任何一种分布都可以看做是多个高斯分布的组合



➢ 混合高斯模型

任何一种分布都可以看做是多个高斯分布的组合





- 像素灰度(随时间)的概率密度函数：

$$p(I)=\sum_{q=1}^{Q} w_q \mathcal{N}\left(I;\mu_q,\sigma_q^2\right)$$

$$G\left(I;\mu_q,\sigma_q\right)=\frac{1}{\sqrt{2\pi}\sigma_q}e^{-\frac{\left(I-\mu_q\right)^2}{2\sigma_q^2}}$$

$$p(\mathbf{x})=\sum_{q=1}^{Q} p\left(\mathbf{x}\middle|\omega_q\right)P\left(\omega_q\right)$$

- 任务：在线计算 w_q,μ_q,σ_q

混合高斯背景建模步骤

- 模型初始化 将采到的第一帧图像的每个象素的灰度值作为均值，再赋以较大的方差。初值Q=1，w=1.0。
- 模型学习 将当前帧的对应点象素的灰度值与已有的Q个高斯模型作比较，若满足 $|x_k-\mu_{q,k}|<2.5\sigma_{q,k}$ ，则按上页方式调整第q个高斯模型的参数和权重；否则转入(3)：
- 增加/替换高斯分量 若不满足条件，且$q<Q$，则增加一个新分量；若$q=Q$，则替换
- 判断背景 $B=\arg\min_b(\sum_{q=1}^{b}w_q>T)$
- 判断前景

混合高斯模型迭代计算原理

迭代计算：

$$w_q(k+1)=(1-\alpha)w_q(k)+\alpha M_q(k+1)$$

$$\mu_q(k+1)=(1-\rho)\mu_q(k)+\rho I(k+1)$$

$$\sigma_q^2(k+1)=(1-\rho)\sigma_q^2(k)+\rho\left(I(k+1)-\mu_q(k+1)\right)^2$$

$$\rho=\alpha G\left(I(k+1);\mu_q,\sigma_q\right)$$

- $M_q(k)$为二值化函数，仅当像素值匹配第q类时取1，其余为0
- 类别数取值不大于5

➢ OpenCV实例

```
resize(source, image, Size(source.cols/2, source.rows/2), INTER_LINEAR);

if (foreGround.empty())
    foreGround.create(image.size(), image.type());

pBgModel->apply(image, fgMask);

GaussianBlur(fgMask, fgMask, Size(5, 5), 0);
threshold(fgMask, fgMask, 10, 255, THRESH_BINARY);

foreGround = Scalar::all(0);
image.copyTo(foreGround, fgMask);

pBgModel->getBackgroundImage(backGround);

// 显示原始图像及背景，前景
imshow("Source", source);
imshow("Background", backGround);
imshow("ForeGround", foreGround);
```

2.2 背景建模处理实例

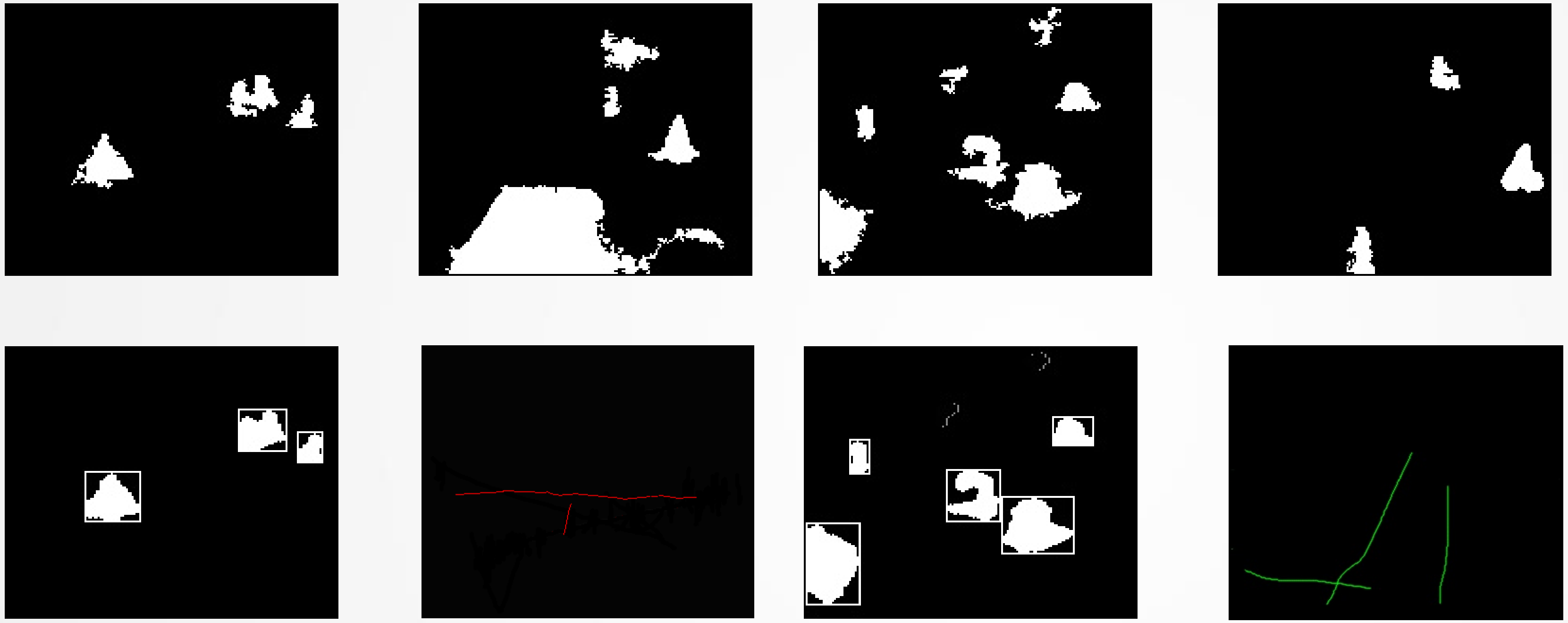







分别是原视频序列中的第281，477，1072，1399帧图像

2.2 背景建模处理实例

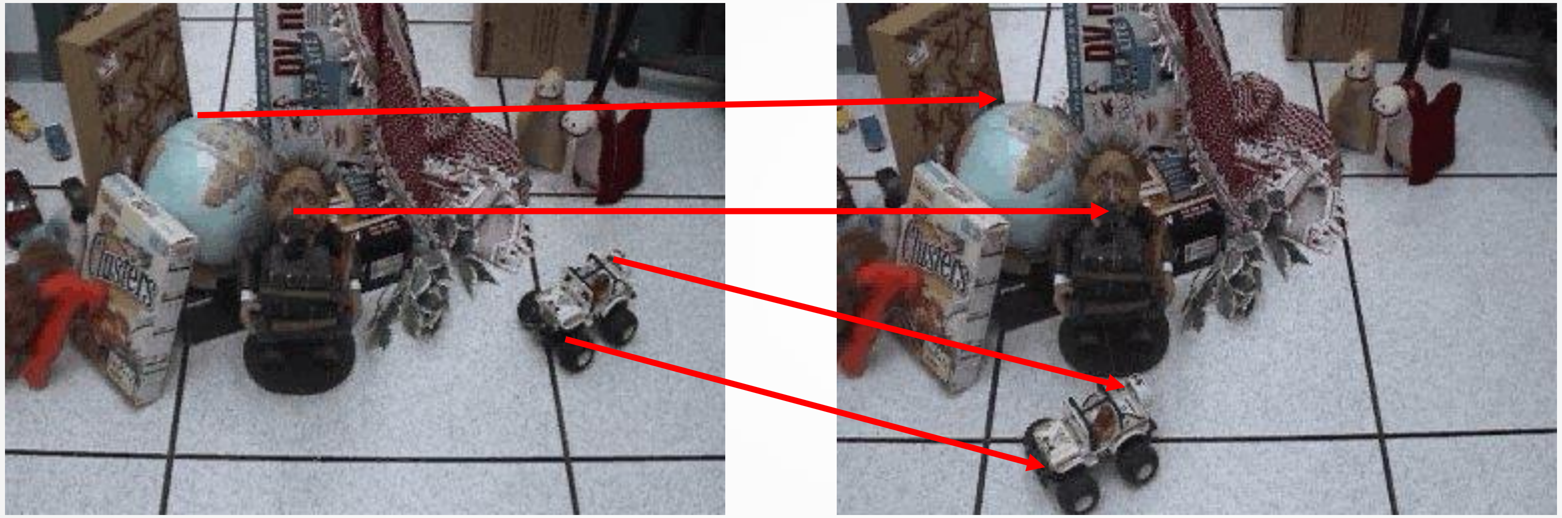


提取的前景图像和轨迹跟踪结果

光流估计

- 光流估计基本模型
- Lucas-Kanade方法
- 光流估计示例

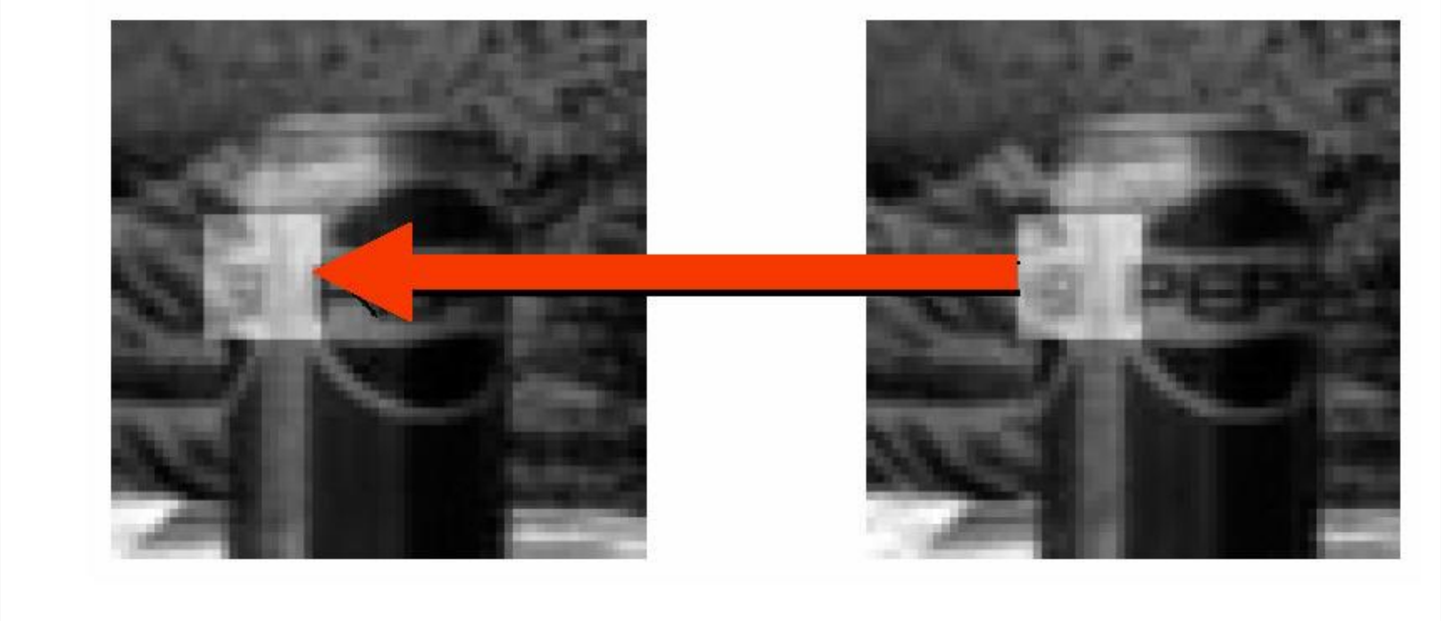
- 解决什么问题？



- 第一帧图像中的点移到了第二帧图像中的什么位置？

3.1 光流估计基本模型

- 恒定亮度假设(BCM)



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

在每一像素(x,y)处，有：

$$I(x+\Delta x, y+\Delta y, t+1)$$

$$= I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}$$

因此有：

$$I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t} = I(x,y,t)$$

$$\Rightarrow$$

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t} = 0 \Rightarrow I_x\Delta x + I_y\Delta y = -I_t$$

- 任务：求$(u,v) = (\Delta x,\ \Delta y)$
- 困难：1个方程两个未知数

- 一个小方格里的所有像素位移相同

$$\begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix}$$

即$A\mathbf{u} = b$

$$A = \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \end{bmatrix};$$

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix};\ b = \begin{bmatrix} I_{t1} \\ I_{t2} \\ \vdots \end{bmatrix}$$

3.2 Lucas-Kanade方法

- 最优化问题(超定方程求解)

$$\min \|A\mathbf{u} - b\|$$

- 最小二乘解：

$$\mathbf{u} = (A^{T}A)^{-1}A^{T}b$$

- 区域像素只有2个时，就是2元1次方程组求解！
多个像素，比如3*3时，则是求上述最小二乘解

- 思路：在一个小的图像邻域内速度近似一致
- 约束：$E(\Delta x, \Delta y) = \sum_i \ w_i^2 (I_{xi}\, \Delta x + I_{yi}\, \Delta y + \ I_{ti})^2$
  $\min E(\Delta x, \Delta y)$
- 对应 $\begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_N \end{bmatrix} \begin{bmatrix} I_{x1} & I_{y1} \\ I_{x2} & I_{y2} \\ \vdots & \vdots \end{bmatrix} A\mathbf{u} = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & w_N \end{bmatrix} b$
- 类似前述求解，可得$\mathbf{u} = (A^{\mathrm{T}}W^2A)^{-1}A^{\mathrm{T}}W^2b$

- 可信度判断：矩阵求逆是否能实现？

$$A^TA = \begin{bmatrix} \sum I_xI_x & \sum I_xI_y \\ \sum I_xI_y & \sum I_yI_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y] = \sum \nabla I (\nabla I)^T$$

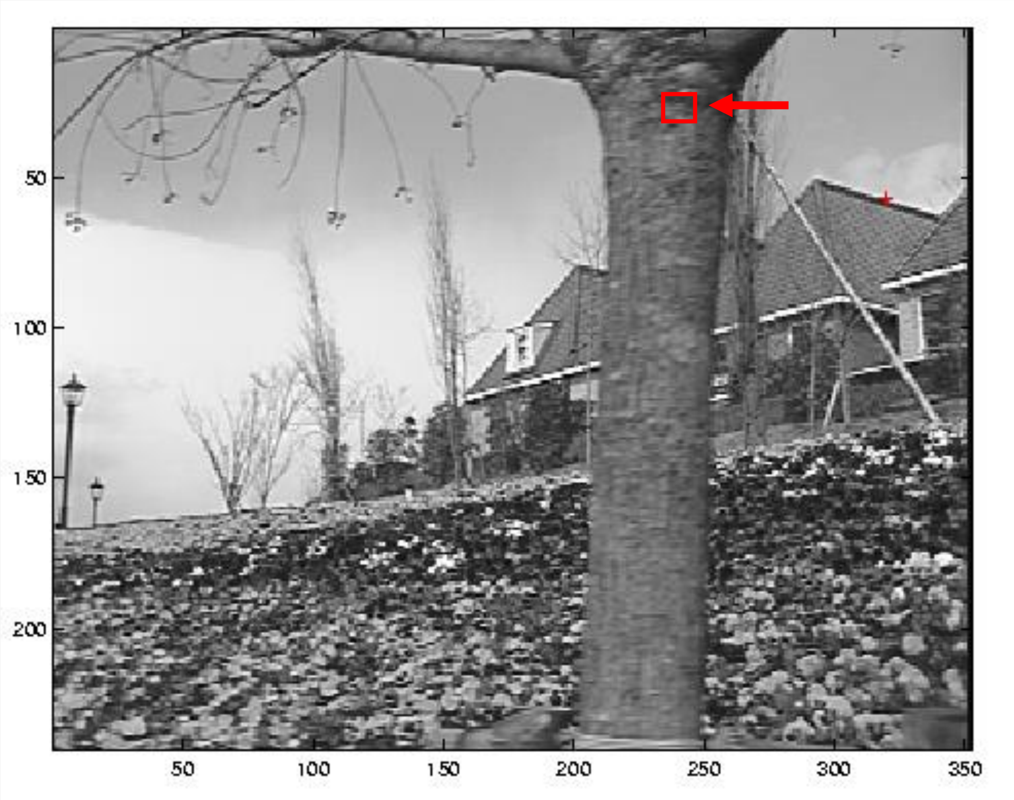
- 通过特征值判断是否计算可信

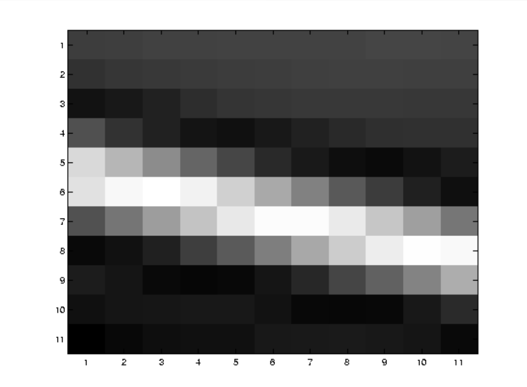
➢ OpenCV相关函数

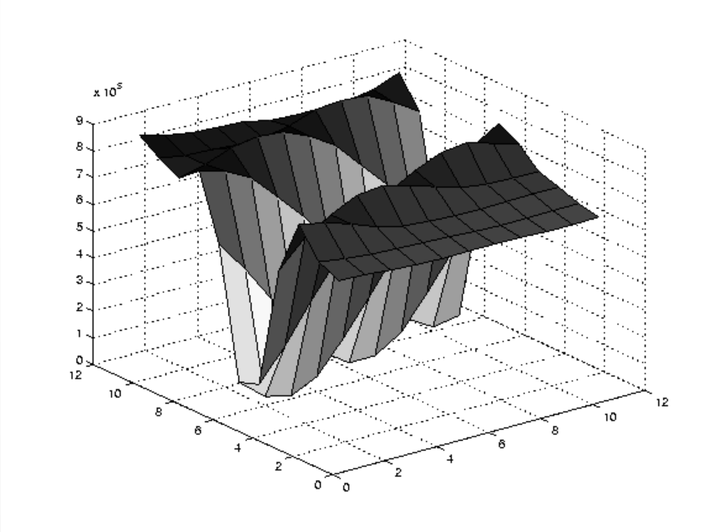
```
void calcOpticalFlowPyrLK(InputArray prevImg, InputArray nextImg,
InputArray prevPts, InputOutputArray nextPts, OutputArray status,
OutputArray err, Size winSize=Size(21,21), int maxLevel=3, TermCriteria
criteria=TermCriteria(TermCriteria::COUNT+TermCriteria::EPS, 30, 0.01),
int flags=0, double minEigThreshold=1e-4 )
```

- *prevImg*，第一帧图像。
- *nextImg*，第二帧图像。
- *prevPts*，第一帧图像中的所有特征点向量。
- *nextPts*，第二帧图像中的所有特征点向量。
- *status*，输出状态向量；如果相应点光流被发现，向量的每个元素被设置为1，否则，被置为0。

- 边缘(对比Harris算子)





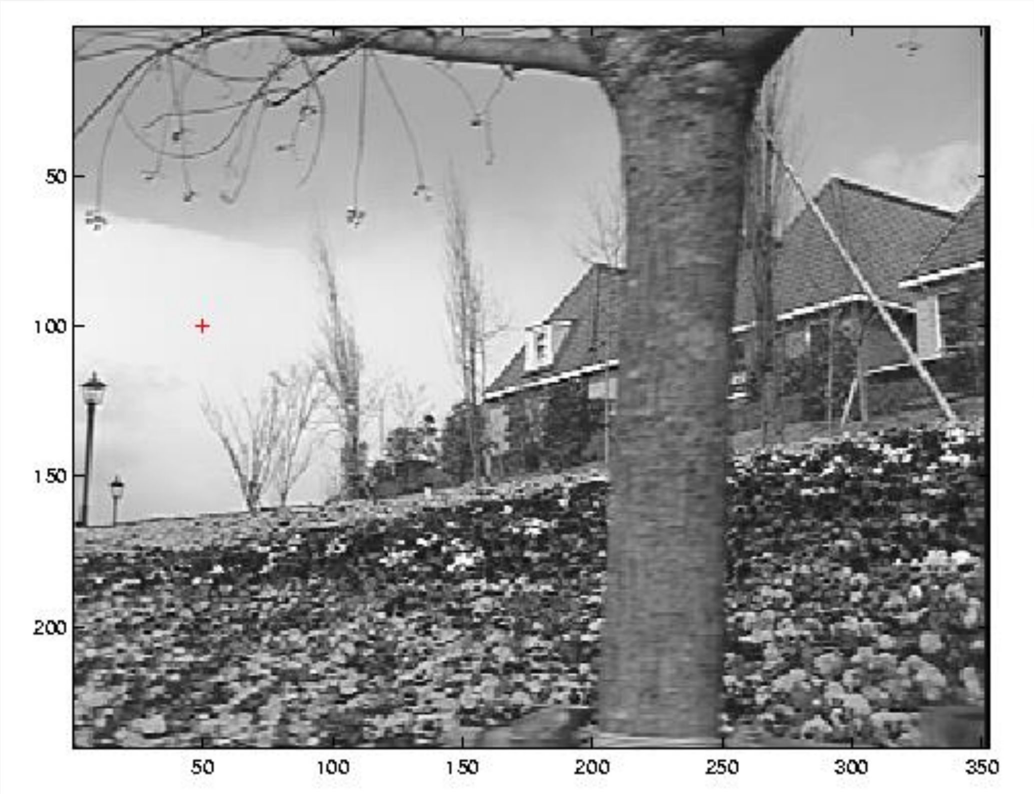


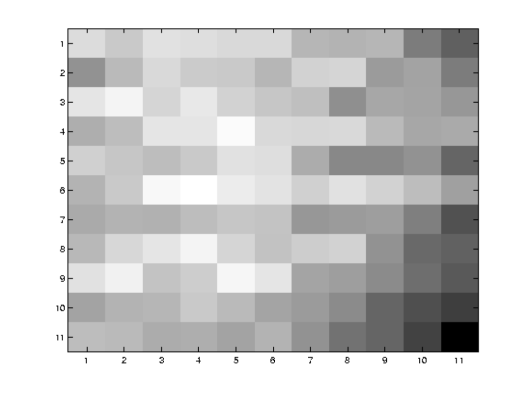
$\sum \nabla I (\nabla I)^T$ - 沿某一方向灰度剧烈变化

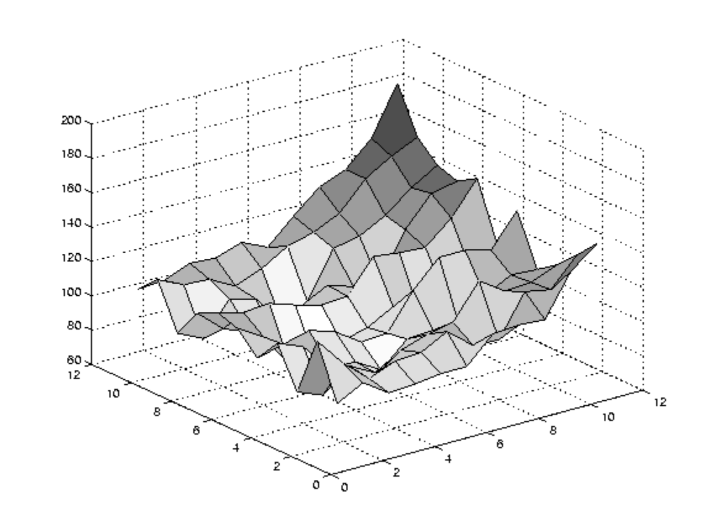
- 大λ_1，小 λ_2

3.2 Lucas-Kanade方法

➤ 低纹理区域(对比Harris算子)

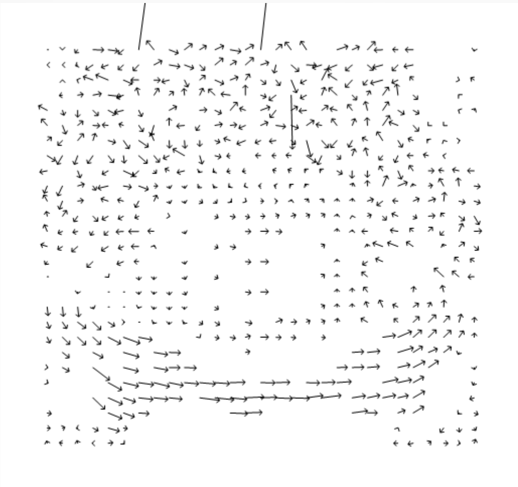


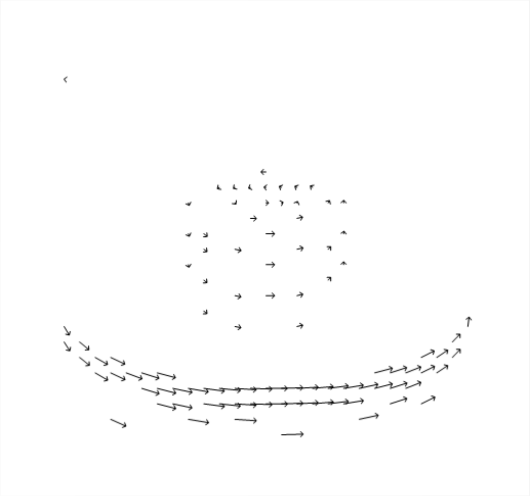


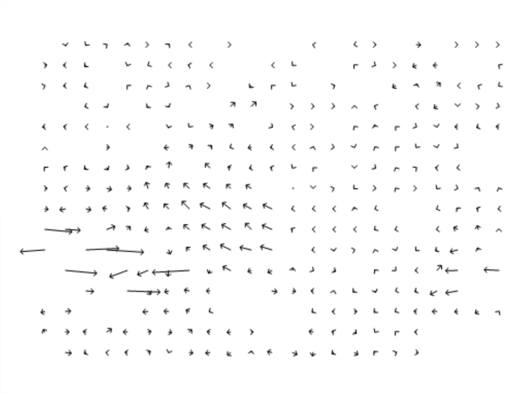


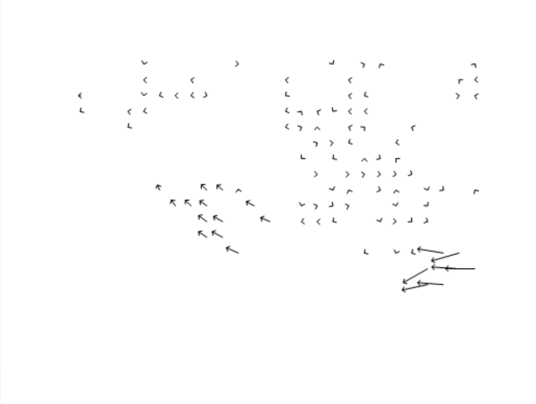
$\sum \nabla I (\nabla I)^T$ - 小λ_1，小 λ_2

3.3 光流估计示例

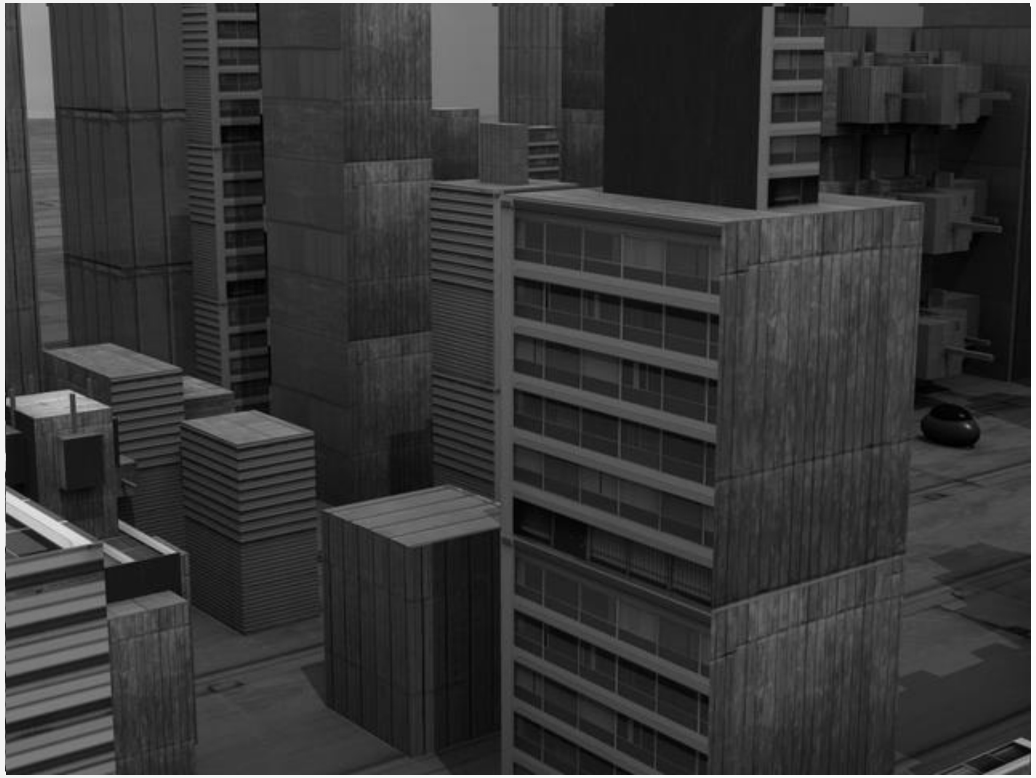


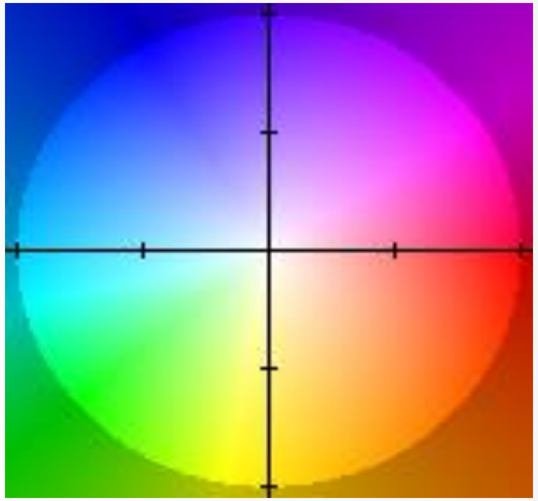






3.3 光流估计示例





总结

4.1 总结

- 特征提取：角点是典型特征，Harris算子是常用角点检测算子
- 背景建模：摄像机静止时的有效检测手段，
  运动目标=当前帧-背景
- 光流估计：基于恒定亮度假设，L-K方法在角点检测基础上，得到每一点的运动向量

感谢各位的聆听！

Roland