



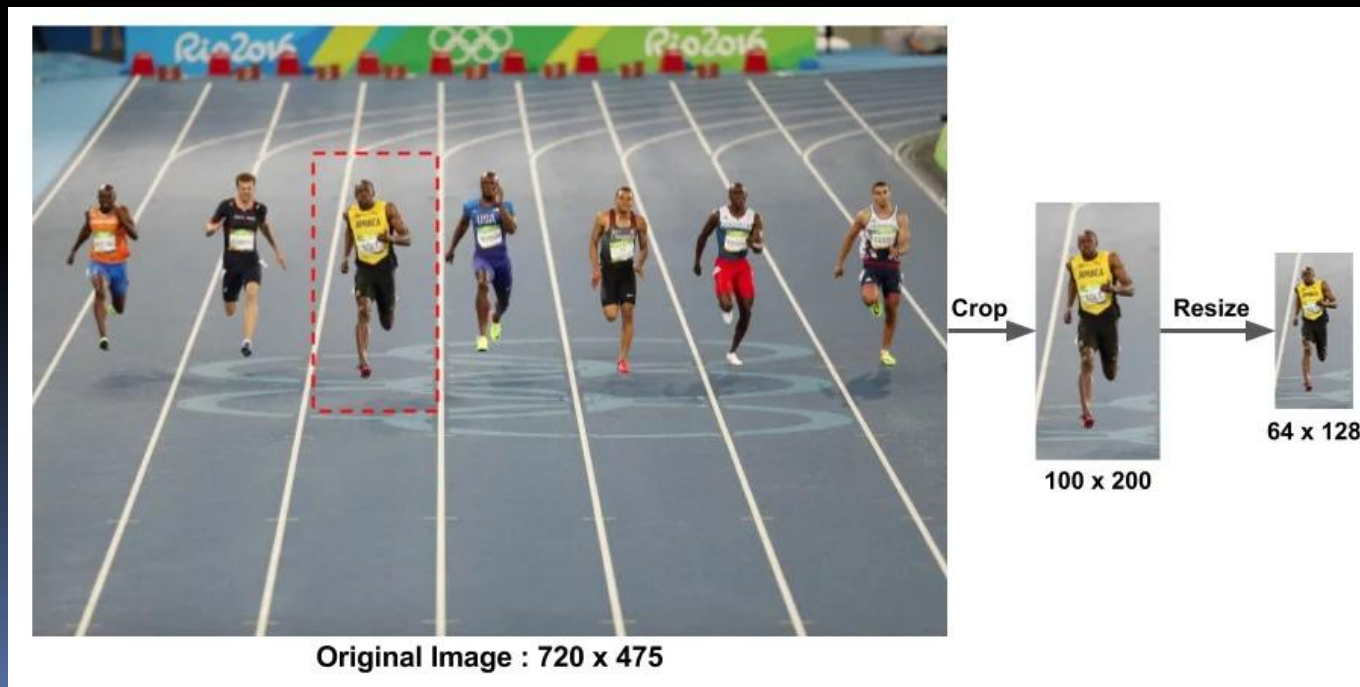
See book Chapter 4.1



IMAGE FEATURES (图像特征)

HOG 梯度方向直方图

- Histogram of Oriented Gradients
- Width-Height-Channel \longrightarrow vector
 - $64 \times 128 \times 3 \longrightarrow 3780$ in HOG





步骤

- 灰度图Gamma校正（不必须）
 - 梯度计算
 - 8x8 Cell梯度直方图
 - 16x16 Block归一化
 - 计算HOG特征描述
- 

梯度计算

$$\text{垂直方向 } y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad \text{水平方向 } x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

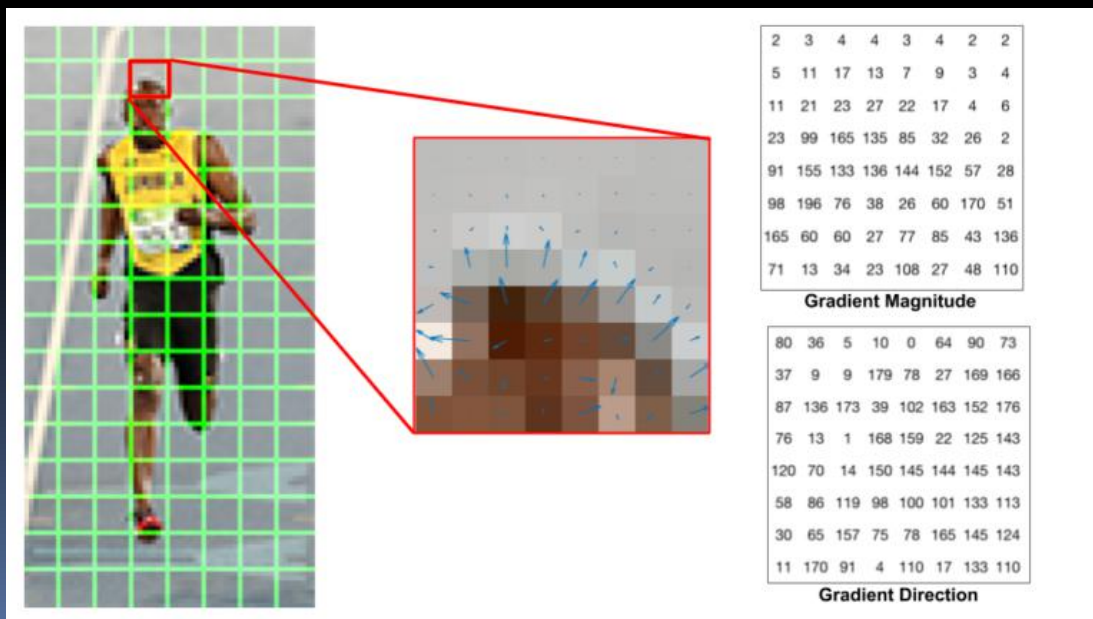
$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$



8x8 Cell 梯度直方图

- 一个cell
 - 8-8-3个数值
 - 8-8-2个梯度值（方向、大小）
 - 9 bins (0, 20, ..., 160) 直方图



| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 80 | 36 | 5 | 10 | 0 | 64 | 90 | 73 |
| 37 | 9 | 9 | 179 | 78 | 27 | 169 | 166 |
| 87 | 136 | 173 | 39 | 102 | 163 | 152 | 176 |
| 76 | 13 | 1 | 168 | 159 | 22 | 125 | 143 |
| 120 | 70 | 14 | 150 | 145 | 144 | 145 | 143 |
| 58 | 86 | 119 | 98 | 100 | 101 | 133 | 113 |
| 30 | 65 | 157 | 75 | 78 | 165 | 145 | 124 |
| 11 | 170 | 91 | 4 | 110 | 17 | 133 | 110 |

Gradient Direction

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 2 | 3 | 4 | 4 | 3 | 4 | 2 | 2 |
| 5 | 11 | 17 | 13 | 7 | 9 | 3 | 4 |
| 11 | 21 | 23 | 27 | 22 | 17 | 4 | 6 |
| 23 | 99 | 165 | 135 | 85 | 32 | 26 | 2 |
| 91 | 155 | 133 | 136 | 144 | 152 | 57 | 28 |
| 98 | 196 | 76 | 38 | 26 | 60 | 170 | 51 |
| 165 | 60 | 60 | 27 | 77 | 85 | 43 | 136 |
| 71 | 13 | 34 | 23 | 108 | 27 | 48 | 110 |

Gradient Magnitude



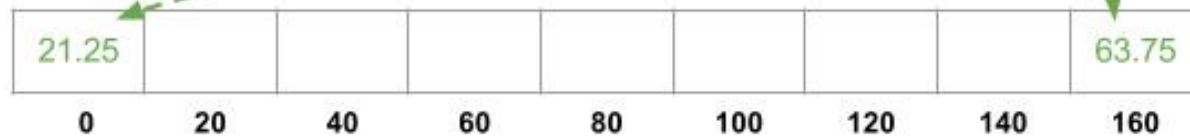
Histogram of Gradients

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 80 | 36 | 5 | 10 | 0 | 64 | 90 | 73 |
| 37 | 9 | 9 | 179 | 78 | 27 | 169 | 166 |
| 87 | 136 | 173 | 39 | 102 | 163 | 152 | 176 |
| 76 | 13 | 1 | 168 | 159 | 22 | 125 | 143 |
| 120 | 70 | 14 | 150 | 145 | 144 | 145 | 143 |
| 58 | 86 | 119 | 98 | 100 | 101 | 133 | 113 |
| 30 | 65 | 157 | 75 | 78 | 165 | 145 | 124 |
| 11 | 170 | 91 | 4 | 110 | 17 | 133 | 110 |

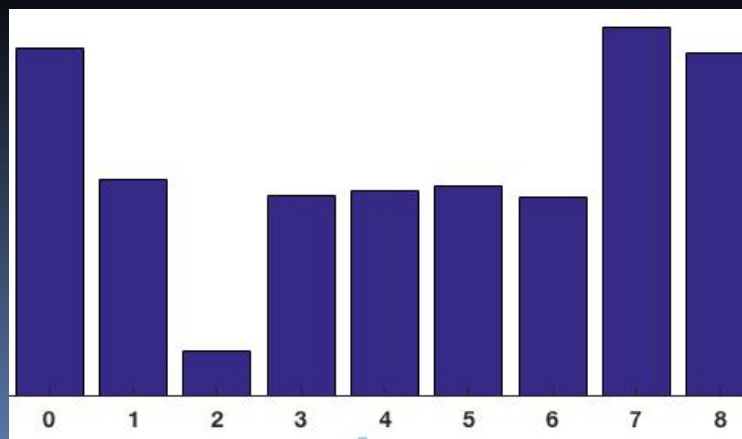
Gradient Direction

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 2 | 3 | 4 | 4 | 3 | 4 | 2 | 2 |
| 5 | 11 | 17 | 13 | 7 | 9 | 3 | 4 |
| 11 | 21 | 23 | 27 | 22 | 17 | 4 | 6 |
| 23 | 99 | 165 | 135 | 85 | 32 | 26 | 2 |
| 91 | 155 | 133 | 136 | 144 | 152 | 57 | 28 |
| 98 | 196 | 76 | 38 | 26 | 60 | 170 | 51 |
| 165 | 60 | 60 | 27 | 77 | 85 | 43 | 136 |
| 71 | 13 | 34 | 23 | 108 | 27 | 48 | 110 |

Gradient Magnitude

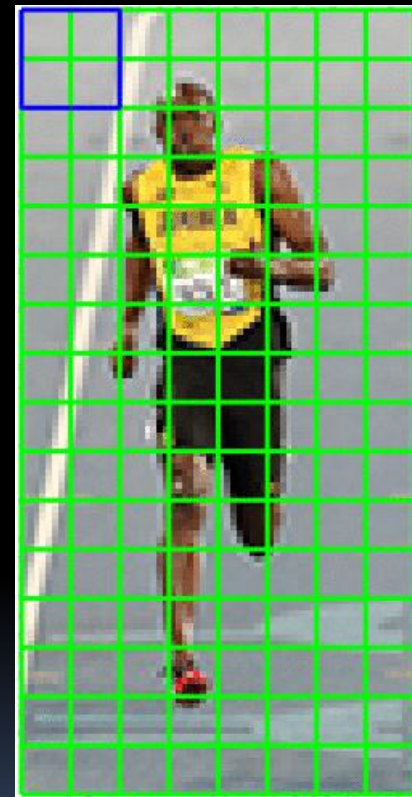


Histogram of Gradients



16x16 Block归一化

- 一个Block包含四个8x8 cell
 - $4 \times 9 = 36$ 个数值
 - 向量归一化



计算HOG特征描述

- $7 \times 15 = 105$ 个Block
- $105 \times 36 = 3780$
- Hog特征描述：一个维度为3780的向量

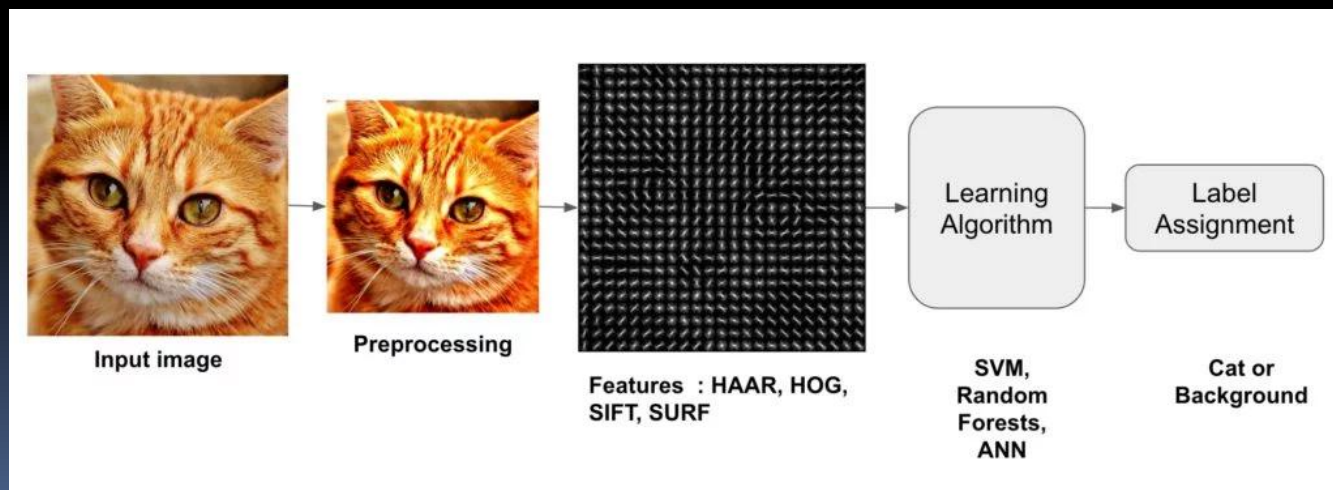
HOG的可视化效果

- 显示每个Cell最重要的梯度方向和大小



应用

- 利用HOG 实现手写数字图像分类
- Classification
 - 计算机能够分析出每类图像对应HOG的特点
 - 怎么找到特点？



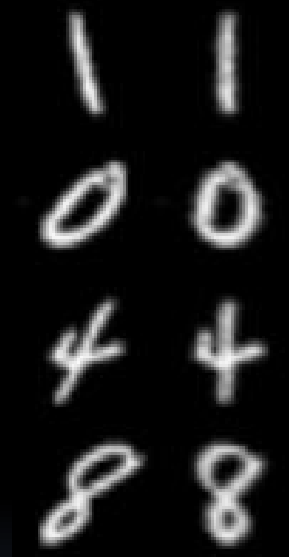
5000 images



预处理

- 图像对齐

在处理前提供更标准规范的输入，往往对提高准确性起到非常关键的作用（甚至往往影响到问题的可解性）



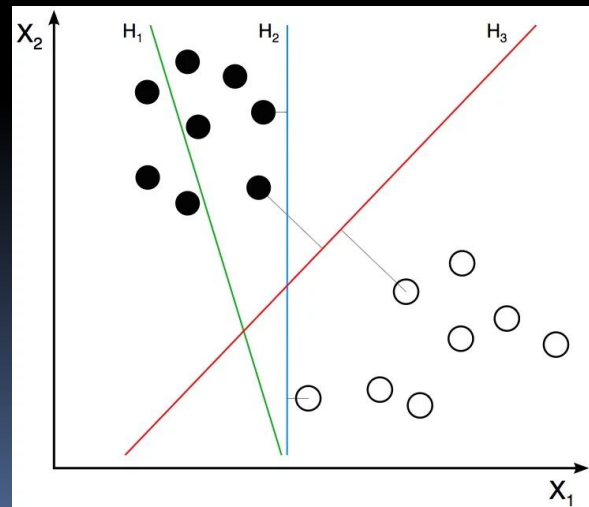
计算图像的HOG特征描述

- 使用了不同的参数
 - HOG特征描述为81维向量

```
HOGDescriptor hog(  
    Size(20, 20), //winSize  
    Size(8, 8), //blocksize  
    Size(4, 4), //blockStride,  
    Size(8, 8), //cellSize,  
    9, //nbins,  
    1, //derivAper,  
    -1, //winSigma,  
    HOGDescriptor::L2Hys, //histogramNormType,  
    0.2, //L2HysThresh,  
    false, //gamma correction,  
    HOGDescriptor::DEFAULT_NLEVELS, //nlevels=64  
    false);  
  
void CreateTrainTestHOG(vector<vector<float>> &trainHOG, vector<ve  
  
    for (int y = 0; y < deskewedtrainCells.size(); y++) {  
        vector<float> descriptors;  
        hog.compute(deskewedtrainCells[y], descriptors);  
        trainHOG.push_back(descriptors);  
    }  
  
    for (int y = 0; y < deskewedtestCells.size(); y++) {  
        vector<float> descriptors;  
        hog.compute(deskewedtestCells[y], descriptors);  
        testHOG.push_back(descriptors);  
    }  
}
```


分类模型训练

- SVM (Support Vector Machines) 支持向量机
 - 每个HOG描述是高维空间中的一个点
 - 简化二分类:
 - 已知若干点属于分别两个类别，能否找在高维空间找到一个超平面，能够有效地将这些点按类别分开，并且尽量距离两个类别的点集都尽量远？
 - 多分类：二分类的扩展





LIBSVM

- LIBSVM是台湾大学林智仁(Lin Chih-Jen)教授等开发设计的一个简单、易于使用和快速有效的SVM模式识别与回归的软件包
 - OpenCV中包含实现
- 


```
Ptr<SVM> svmInit(float C, float gamma)
```

```
{
```

```
    Ptr<SVM> svm = SVM::create();
```

```
    svm->setGamma(gamma);
```

```
    svm->setC(C);
```

```
    svm->setKernel(SVM::RBF);
```

```
    svm->setType(SVM::C_SVC);
```

```
    return svm;
```

```
}
```

```
void svmTrain(Ptr<SVM> svm, Mat &trainMat, vector<int> &trainLabels)
```

```
{
```

```
    Ptr<TrainData> td = TrainData::create(trainMat, ROW_SAMPLE, trainLabels);
```

```
    svm->train(td);
```

```
    svm->save("ClassifierModel.yml");
```

```
}
```

```
void svmPredict(Ptr<SVM> svm, Mat &testResponse, Mat &testMat)
```

```
{
```

```
    svm->predict(testMat, testResponse);
```

```
}
```

思考?

- 在当前基础上，如何实现手写数字的检测？
- 使用SIFT特征完成手写数字分类，可能性和思路？
- 编程作业
 - 使用OpenCV，利用HOG完成 行人/汽车 分类、检测