

# 电类工程导论C 实验报告

——SIFT算法

## 目录页

1. 引言
2. 实验环境
3. 实验过程
  - 3.1. 获取不同尺度的图像
  - 3.2. 计算特征点（角点）
  - 3.3. **SIFT** 描述子计算的预备工作
  - 3.4. 计算 **SIFT** 描述子
  - 3.5. 比较两个描述子
  - 3.6. 可视化
4. 成果展示
5. 总结
6. 参考
7. 源代码

# 1. 引言

在这次的实验里，我主要完成了自己实现简易SIFT算法。主要步骤有利用角点检测、计算主梯度方向，坐标系转换，SIFT 描述子计算，SIFT算子比较，结果连线显示等步骤。

## 2. 实验环境

Mac OS X 10.11.2 + opencv 2.4.12 + numpy1.10.1 + Python 2.7.10  
(64bit)

## 3. 实验过程

### 3.1. 获取不同尺度的图像

首先对于不同尺度的图像，我们要进行归一，这一步里，我利用了cv2.resize函数来实现图像的放缩，但是具体的比例还需要自己通过实验试出来。

代码如下：

```
def getResizedImg(ori, scale):  
    x = int(ori.shape[0]*scale)  
    y = int(ori.shape[1]*scale)  
    return cv2.resize(ori, (x, y))
```

## 3.2.计算特征点（角点）

在确定了尺度之后，我们要对图像进行特征点的选取，这一步也叫角点检测，这里主要利用的 Harris 的角点检测方法，函数是利用了 cv2 自带的 `goodFeaturesToTrack` 函数，这个函数可以返回规定数目的最好的特征点。不过返回格式有点奇怪，不太好用，需要转换成一个关于点的 `list` 形式才可以方便的进行调用和使用，此部分代码如下：

```
#获得特征点
goodFeatureTemp = cv2.goodFeaturesToTrack(img,100,0.01,5)
goodFeatures = []
#转换为 list
for gf in goodFeatureTemp:
    goodFeatures.append((int(gf[0][0]),int(gf[0][1])))
```

## 3.3.SIFT 描述子计算的预备工作

### 3.3.1.计算每个特征点的主梯度方向

计算主梯度方向的原理其实就是直方图投票的原理，我们把一个圆周360度划分为36个块，然后计算每个块里的总梯度强度，找到最强的那一个块，然后取中值作为主方向。

代码如下，im 表示图片矩阵，f 是一个特征点。

```
#计算主梯度方向
def getMainDirection(im,f):
    Hist = [0]*36
    fx = int(f[0]); fy=int(f[1]);
    R = 8
    for x in range(fx-R,fx+R+1):
        for y in range(fy-R,fy+R+1):
            if not isValid(im,x,y):
                continue
            Theta = getTheta(im,x,y)
            M = getIntension(im,x,y)
            ind = Theta/10
            if(ind==36): ind = 0;
            Hist[ind] += M

    maxTheta = 0
    for ind in range(36):
        if(Hist[ind]>maxTheta):#比当前的主方向的强度大
            maxTheta = ind*10+5 #取中值代表主方向
    return maxTheta
```

计算过程中涉及到对某一个点求梯度强度和梯度方向的过程，一开始我想先对所有的点都进行计算，后来发现这样很慢，而且浪费了时间，因为真正需要计算的只有特征点和特征点邻域内的点，这个数量远远小于所有的点的个数。

所以我是每次单独地去计算强度和方向。

代码如下，注意 hypot函数和 atan2函数的利用，还要记得最后要+pi 因为返回值是-pi 到 pi 的，不方便计算。

#获取梯度强度

```
def getIntension(im,x,y):  
    x = int(x); y = int(y);  
    dx = int(im[x+1,y])-int(im[x-1,y])  
    dy = int(im[x,y+1])-int(im[x,y-1])  
    return math.hypot(dx,dy)
```

#获取梯度方向

```
def getTheta(im,x,y):  
    x = int(x); y = int(y);  
    dx = int(im[x+1,y])-int(im[x-1,y])  
    dy = int(im[x,y+1])-int(im[x,y-1])  
    return int(math.atan2(dy,dx)*180/math.pi+180)
```

### 3.3.2.根据主梯度方向进行坐标系变换

根据公式：

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \times \begin{pmatrix} x \\ y \end{pmatrix}$$

我们可以得到转换坐标的函数如下：

```
def convertX(x,y,x0,sinDir,cosDir):  
    return int(x * cosDir - y*sinDir + x0)  
  
def convertY(x,y,y0,sinDir,cosDir):  
    return int(x * sinDir + y*cosDir + y0)
```

### 3.3.3.根据新的坐标系，描述特征点邻域

接下来要对关键点邻域的梯度方向和梯度强度进行记忆性存储，方便后面计算描述符的时候可以加快效率。

```
for f in features:
    fx = int(f[0]); fy=int(f[1]);
    if(not isValid(im,fx,fy)):
        continue
    mainDirection = getMainDirection(im,f)
    dir = mainDirection*math.pi/180.0
    sinDir = math.sin(dir)
    cosDir = math.cos(dir)

    #坐标变换
    for x in range(-8,9):
        for y in range(-8,9):
            rx = convertX(x,y,fx,sinDir,cosDir)
            rx1 = convertX(x-1,y,fx,sinDir,cosDir)
            rx2 = convertX(x+1,y,fx,sinDir,cosDir)

            ry = convertY(x,y,fy,sinDir,cosDir)
            ry1 = convertY(x,y-1,fy,sinDir,cosDir)
            ry2 = convertY(x,y+1,fy,sinDir,cosDir)

            Inten[x][y] = math.hypot((im[rx1,ry]-im[rx2,ry]),
                                     (im[rx,ry1]-im[rx,ry2]))
            Theta[x][y] = math.atan2(im[rx1][ry]-im[rx2][ry],
                                      im[rx][ry1]-im[rx][ry2])
                                      + math.pi + dir

            while Theta[x][y] >= math.pi * 2:
                Theta[x][y] -= math.pi * 2
            while Theta[x][y] < 0:
                Theta[x][y] += 2*math.pi
```

这里的 Inten记录了邻域的梯度强度，Theta 记录了邻域的梯度方向。

### 3.4.计算 SIFT 描述子

有了之前的铺垫，描述子的计算就非常简洁易懂了，我们只需要对邻域的4\*4\*16个点记性累计统计即可算出一个128维的描述子，我们这里用 Hist 来记录描述子。

代码如下：

```
#计算描述子
```

```
Hist=[]
```

```
for i in range(-2,2):
    for j in range(-2,2):
        tmpHist = [0]*8
        for m in range(4):
            for n in range(4):
                xx = i * 4 + m
                yy = j * 4 + n
                tmpHist[int(Theta[xx][yy]/(math.pi/4))] += Inten[xx][yy]

        Hist+=tmpHist
```

这是我一开始的代码，后来发现自己忘了利用插值的方法来提高准确性。所以后来又写了如下函数来实现插值：

```
#计算插值
```

```
def insertValue(im,x,y,dir):
    x0 = int(x); y0 = int(y);
    res = (
        getTheta(im,x0,y0)      *(x0+1-x) *(y0+1-y) +
        getTheta(im,x0+1,y0)    *(x-x0)    *(y0+1-y) +
        getTheta(im,x0,y0+1)    *(x0+1-x) *(y-y0)    +
        getTheta(im,x0+1,y0+1)  *(x-x0)    *(y-y0)
    )
    res -= dir
    while res<0: res += 360;
    return res
```

### 3.5.比较两个描述子

接下来就是查询的过程了，我们输入了两个图片之后，可以先分别对他们提取 SIFT 描述子集合，然后对集合进行双重遍历来算出相似度，然后判断相似度是否超过阈值来判断是否要输出这个点对。这里为了提高效率，也可以使用kd 树的数据结构来进行搜索。

根据公式：

$$d(R_i, S_i) = \sqrt{\sum_{j=1}^{128} (r_{ij} - s_{ij})^2}$$

可以写出如下代码：

```
v_tar, f_t = getSIFTbyURL(url1)
v_i, f_i = getSIFTbyURL(url2)

res_t = []
res_i = []

for t, vt in enumerate(v_tar):
    for i, vi in enumerate(v_i):
        s = 0
        for ind in range(128):
            s += vt[ind]*vi[ind]
        if(s>0.77):
            res_t.append(f_t[t])
            res_i.append(f_i[i])
```



### 3.6.可视化

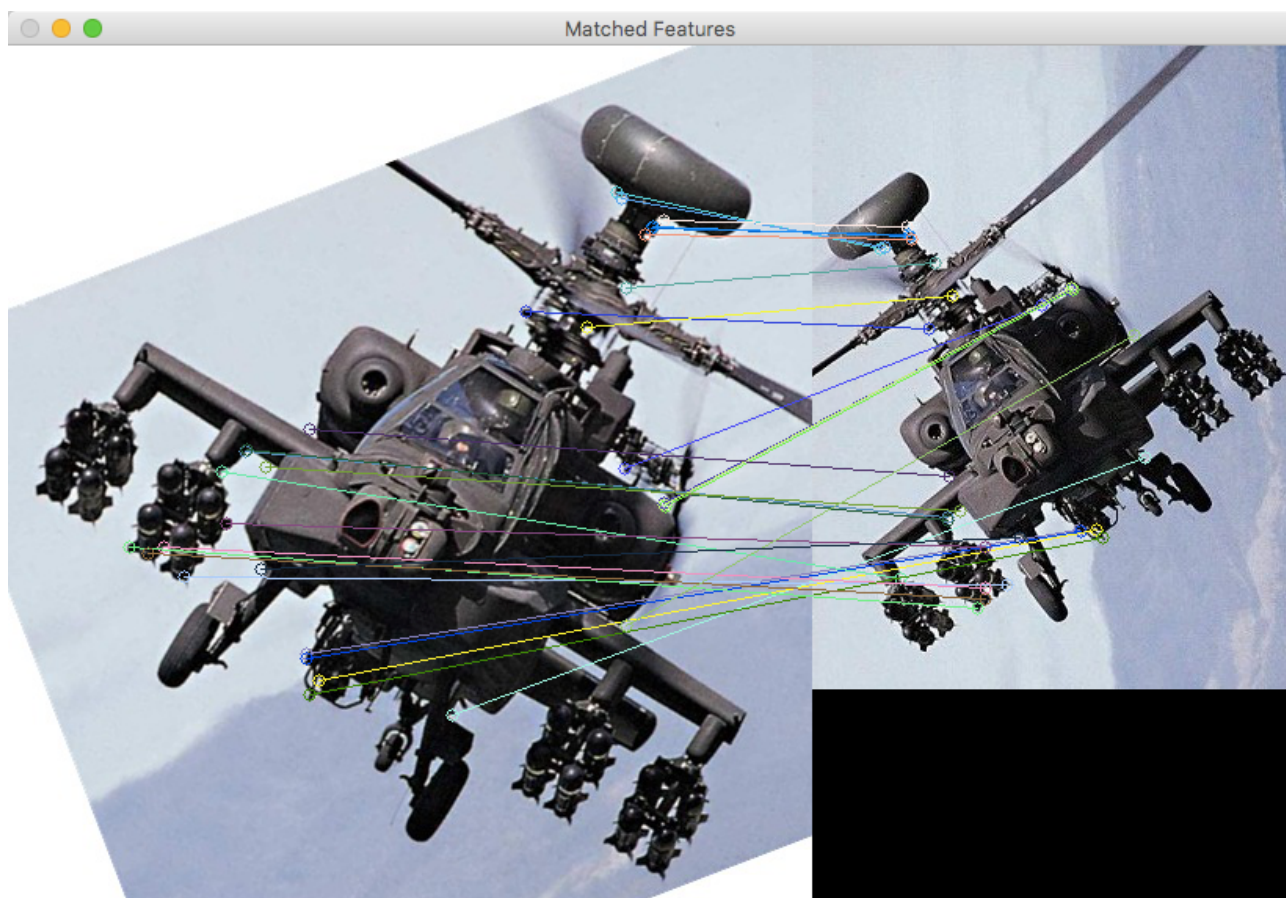
然后我们要对两个 res 的 list 进行可视化，这里我单独写了一个 drawMatches 来实现这个过程。

为了实现关键点的可视化过程，我创建了一个新的图像矩阵，首先存储这两个图片的彩色图像，然后根据关键点画出线即可，这里线的颜色，是我随机生成的，这样做的目的是为了可以精准的对比关键点的位置。代码如下：

```
def drawMatches(img1, kp1, img2, kp2):
    rows1 = img1.shape[0]
    cols1 = img1.shape[1]
    rows2 = img2.shape[0]
    cols2 = img2.shape[1]
    out = np.zeros((max([rows1, rows2]), cols1+cols2, 3), dtype='uint8')
    out[:rows1, :cols1, :] = np.dstack([img1])
    out[:rows2, cols1:cols1+cols2, :] = np.dstack([img2])

    for i in range(len(kp1)):
        (x1, y1) = kp1[i]
        (x2, y2) = kp2[i]
        import random
        b=random.randint(0,255)
        g=random.randint(0,255)
        r=random.randint(0,255)
        color = (b,g,r)
        cv2.circle(out, (int(x1),int(y1)), 4, color, 1)
        cv2.circle(out, (int(x2)+cols1,int(y2)), 4, color, 1)
        cv2.line(out, (int(x1),int(y1)),
                  (int(x2)+cols1,int(y2)), color, 1)
    cv2.imshow('Matched Features', out)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

## 4. 成果展示



（为了精准我特意减少了线条，否则会看不清）

其他的图像，都没有任何地特征匹配，如果放低了阈值的话，确实有会一些，但是很少很少。





## 5. 总结

在这次实验中我更加深入的学习了 SIFT 算法的原理和流程，对整体有了一个新的了解，非常感谢这个机会。我会继续在课余时间研究如何把 KD 树结合到查询里来，提高查询速率，并且在算子计算中进行优化。

## 6. 参考

[http://opencv-python-tutroals.readthedocs.org/en/latest/py\\_tutorials/py\\_feature2d/py\\_matcher/py\\_matcher.html](http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_feature2d/py_matcher/py_matcher.html)  
<http://blog.jobbole.com/84227/>

## 7. 代码

```
1 #coding=utf8
2 import cv2
3 import numpy as np
4 import sys
5 import math
6
7 #获取梯度强度
8 def getIntension(im,x,y):
9     x = int(x); y = int(y);
10    dx = int(im[x+1,y])-int(im[x-1,y])
11    dy = int(im[x,y+1])-int(im[x,y-1])
12    return math.hypot(dx,dy)
13
14 #获取梯度方向
15 def getTheta(im,x,y):
16     x = int(x); y = int(y);
17     dx = int(im[x+1,y])-int(im[x-1,y])
18     dy = int(im[x,y+1])-int(im[x,y-1])
19     return int(math.atan2(dy,dx)*180/math.pi+180)
20
21 #判断是否越界
22 def isValid(im,x,y):
23     return (x>=16) and (x<=(im.shape[0]-16)) and
24           (y>=16) and (y<=(im.shape[1]-16))
25
26 #计算主梯度方向
27 def getMainDirection(im,f):
28     Hist = [0]*36
29     fx = int(f[0]); fy=int(f[1]);
30     R = 8
31     for x in range(fx-R,fx+R+1):
32         for y in range(fy-R,fy+R+1):
33             if not isValid(im,x,y):
34                 continue
35             Theta = getTheta(im,x,y)
36             M = getIntension(im,x,y)
37             ind = Theta/10
38             if(ind==36): ind = 0;
39             Hist[ind] += M
```



```

39     maxTheta = 0
40     for ind in range(36):
41         if(Hist[ind]>maxTheta):#比当前的主方向的强度大
42             maxTheta = ind*10+5 #取中值代表主方向
43     return maxTheta
44
45 #计算插值
46 def insertValue(im,x,y,dir):
47     x0 = int(x); y0 = int(y);
48     res = (
49         getTheta(im,x0,y0)      *(x0+1-x) *(y0+1-y) +
50         getTheta(im,x0+1,y0)    *(x-x0)    *(y0+1-y) +
51         getTheta(im,x0,y0+1)    *(x0+1-x) *(y-y0)    +
52         getTheta(im,x0+1,y0+1) *(x-x0)    *(y-y0)
53     )
54     res -= dir
55     while res<0: res += 360;
56     return res
57
58 def convertX(x,y,x0,sinDir,cosDir):
59     return int(x * cosDir - y*sinDir + x0)
60
61 def convertY(x,y,y0,sinDir,cosDir):
62     return int(x * sinDir + y*cosDir + y0)
63
64 def getSIFTVector(im,features):
65     siftVec = []
66
67     #定义存储单元
68     Inten=[0]*16 # 存储梯度强度
69     Theta=[0]*16 # 存储梯度方向
70     for i in range(16):
71         Inten[i]=[0]*16
72         Theta[i]=[0]*16
73     final={} #128维sift 向量
74
75     for f in features:
76         fx = int(f[0]); fy=int(f[1]);
77         if(not isValid(im,fx,fy)):
78             continue
79         mainDirection = getMainDirection(im,f)
80         dir = mainDirection*math.pi/180.0
81         sinDir = math.sin(dir)
82         cosDir = math.cos(dir)
83
84         #坐标变换
85         for x in range(-8,9):
86             for y in range(-8,9):
87                 rx = convertX(x,y,fx,sinDir,cosDir)

```

```

88         rx1 = convertX(x-1,y,fx,sinDir,cosDir)
89         rx2 = convertX(x+1,y,fx,sinDir,cosDir)
90
91         ry = convertY(x,y,fy,sinDir,cosDir)
92         ry1 = convertY(x,y-1,fy,sinDir,cosDir)
93         ry2 = convertY(x,y+1,fy,sinDir,cosDir)
94
95         Inten[x][y] = math.hypot((im[rx1,ry]-im[rx2,ry]),
96                                 (im[rx,ry1]-im[rx,ry2]))
97         Theta[x][y] = math.atan2(im[rx1][ry]-im[rx2][ry],
98                                 im[rx][ry1]-im[rx][ry2])
99                                 + math.pi + dir
100
101
102         while Theta[x][y] >= math.pi * 2:
103             Theta[x][y] -= math.pi * 2
104         while Theta[x][y] < 0:
105             Theta[x][y] += 2*math.pi
106
107     #计算描述子
108
109     Hist=[]
110
111     for i in range(-2,2):
112         for j in range(-2,2):
113             tmpHist = [0]*8
114             for m in range(4):
115                 for n in range(4):
116                     xx = i * 4 + m
117                     yy = j * 4 + n
118                     tmpHist[int(Theta[xx][yy]/(math.pi/4))]
119                     += Inten[xx][yy]
120
121             Hist+=tmpHist
122
123     sumValue = 0
124     #128维向量描述子
125
126     for i in range(128):
127         sumValue += Hist[i]**2
128     sumValue = math.sqrt(sumValue)
129     if(sumValue!=0):
130         for x in range(128):

```

```

130         for x in range(128):
131             Hist[x] /= float(sumValue)
132         siftVec.append(Hist)
133
134     return siftVec
135
136
137 def getResizedImg(ori,scale):
138     x = int(ori.shape[0]*scale)
139     y = int(ori.shape[1]*scale)
140     return cv2.resize(ori,(x,y))
141
142
143
144 def getSIFTbyURL(url,scale=1,mainDirection=0):
145     filename = url
146     img = cv2.imread(filename,0)
147     if(scale!=1):
148         img = getResizedImg(img,scale)
149
150     #获得特征点
151     goodFeatureTemp = cv2.goodFeaturesToTrack(img,100,0.01,5)
152     goodFeatures = []
153     #转换格式为 list
154     for gf in goodFeatureTemp:
155         goodFeatures.append((int(gf[0][0]),int(gf[0][1])))
156     vector = getSIFTVector(img,goodFeatures)
157
158     print goodFeatures
159     return vector,goodFeatures
160
161 def main():
162     url1 = 'target.jpg'
163     url2 = 'dataset/5.jpg'
164
165     v_tar,f_t = getSIFTbyURL(url1)
166     v_i,f_i = getSIFTbyURL(url2)
167
168     res_t = []
169     res_i = []
170
171     for t,vt in enumerate(v_tar):
172         for i,vi in enumerate(v_i):
173             s = 0
174             for ind in range(128):
175                 s += vt[ind]*vi[ind]
176             if(s>0.77):
177                 res_t.append(f_t[t])
178                 res_i.append(f_i[i])

```

```

179
180     img1 = cv2.imread(url1)
181     img2 = cv2.imread(url2)
182     drawMatches(img1,res_t,img2,res_i)
183     print res_t
184     print res_i
185
186 def drawMatches(img1, kp1, img2, kp2):
187     rows1 = img1.shape[0]
188     cols1 = img1.shape[1]
189     rows2 = img2.shape[0]
190     cols2 = img2.shape[1]
191     out = np.zeros((max([rows1,rows2]),cols1+cols2,3), dtype='uint8')
192     out[:rows1,:cols1,:] = np.dstack([img1])
193     out[:rows2,cols1:cols1+cols2,:] = np.dstack([img2])
194
195     for i in range(len(kp1)):
196         (x1,y1) = kp1[i]
197         (x2,y2) = kp2[i]
198         import random
199         b=random.randint(0,255)
200         g=random.randint(0,255)
201         r=random.randint(0,255)
202         color = (b,g,r)
203         cv2.circle(out, (int(x1),int(y1)), 4, color, 1)
204         cv2.circle(out, (int(x2)+cols1,int(y2)), 4, color, 1)
205         cv2.line(out, (int(x1),int(y1)),
206                  (int(x2)+cols1,int(y2)), color, 1)
207     cv2.imshow('Matched Features', out)
208     cv2.waitKey(0)
209     cv2.destroyAllWindows()
210
211 main()

```

非常感谢何老师和赵学姐的指导。

林禹臣

[yuchenlin@sjtu.edu.cn](mailto:yuchenlin@sjtu.edu.cn)

5140309507

2015.12.15