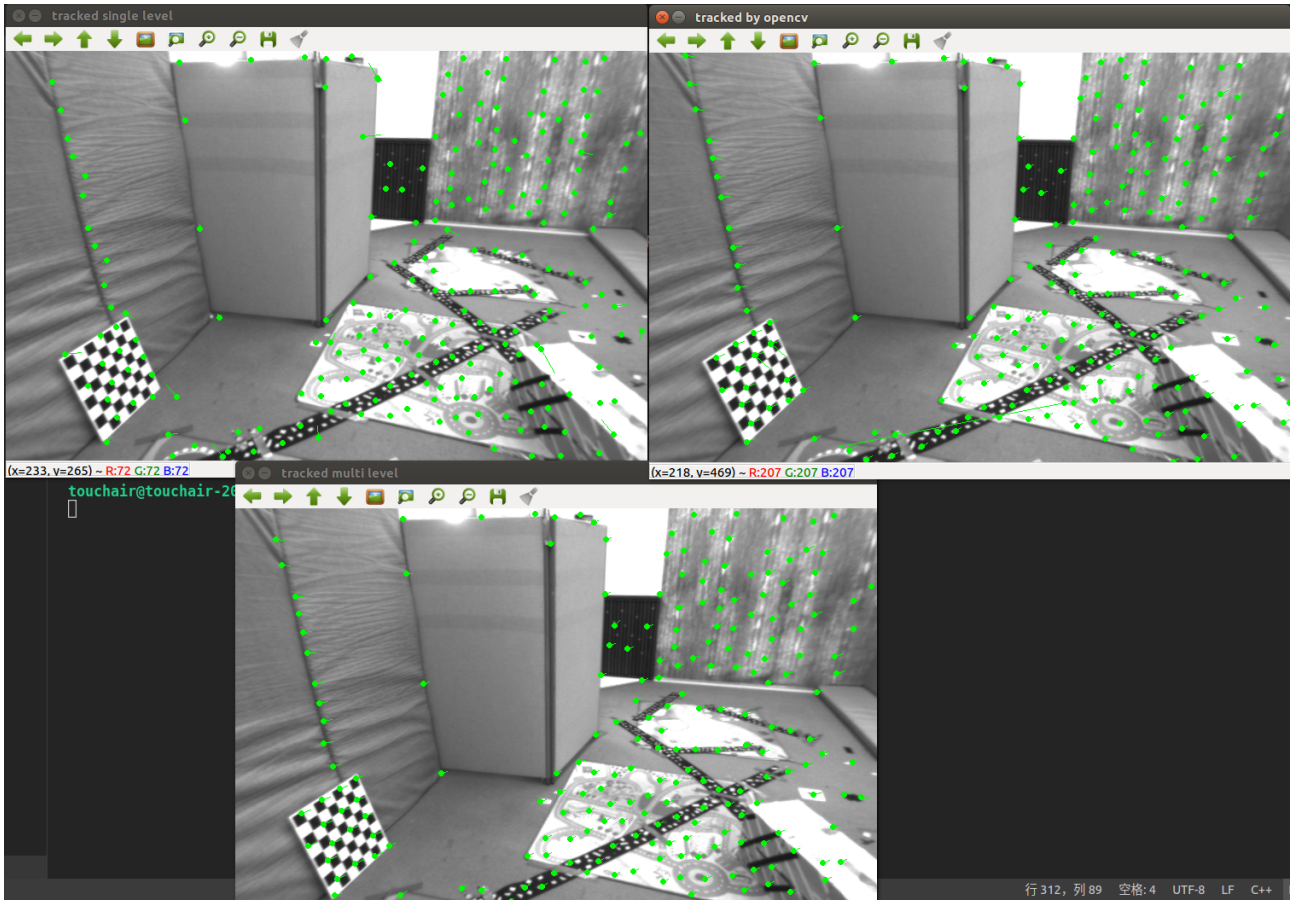
2.1.1 按照估计的是参数的叠加增量还是增量 Warp 将光流法分为叠加(additional)和组合(compositional)算法，按照 Warp 更新规则可以将光流法分为前向（forward）和逆向(inverse)两种算法

2.1.2 与 Lucas-Kanade 算法中那样简单地将迭代的更新 Δp 添加到当前估计的参数 p 不同，组合（compositional）算法中，对扭曲 W(x;Δp)的增量更新必须由 Warp 的当前估计组成 W(x;p)。

2.1.3 forward 方法和 inverse 方法在目标函数上不太一样，一个是把运动向量 p 都是跟着被匹配图像，但是向前方法中的迭代的微小量 Δp 使用 I 计算，反向方法中的 Δp 使用 T 计算，这样计算量便小了。

2.2



2.2.1 误差定义如下

$$e = I_1(x, y) - I_2(x + \Delta x, y + \Delta y)$$

2.2.2 误差对于自变量的导数定义如下

$$\frac{\partial e}{\partial \Delta x} = -\left(\frac{\partial I_2(x + \Delta x, y + \Delta y)}{\partial \Delta x}\right)$$

$$\frac{\partial e}{\partial \Delta y} = -\left(\frac{\partial I_2(x + \Delta x, y + \Delta y)}{\partial \Delta y}\right)$$

图像梯度一般也可以用中值差分：

dx(i,j) = [I(i+1,j) - I(i-1,j)]/2;
dy(i,j) = [I(i,j+1) – I(i,j-1)]/2;

对应的雅克比矩阵计算如下

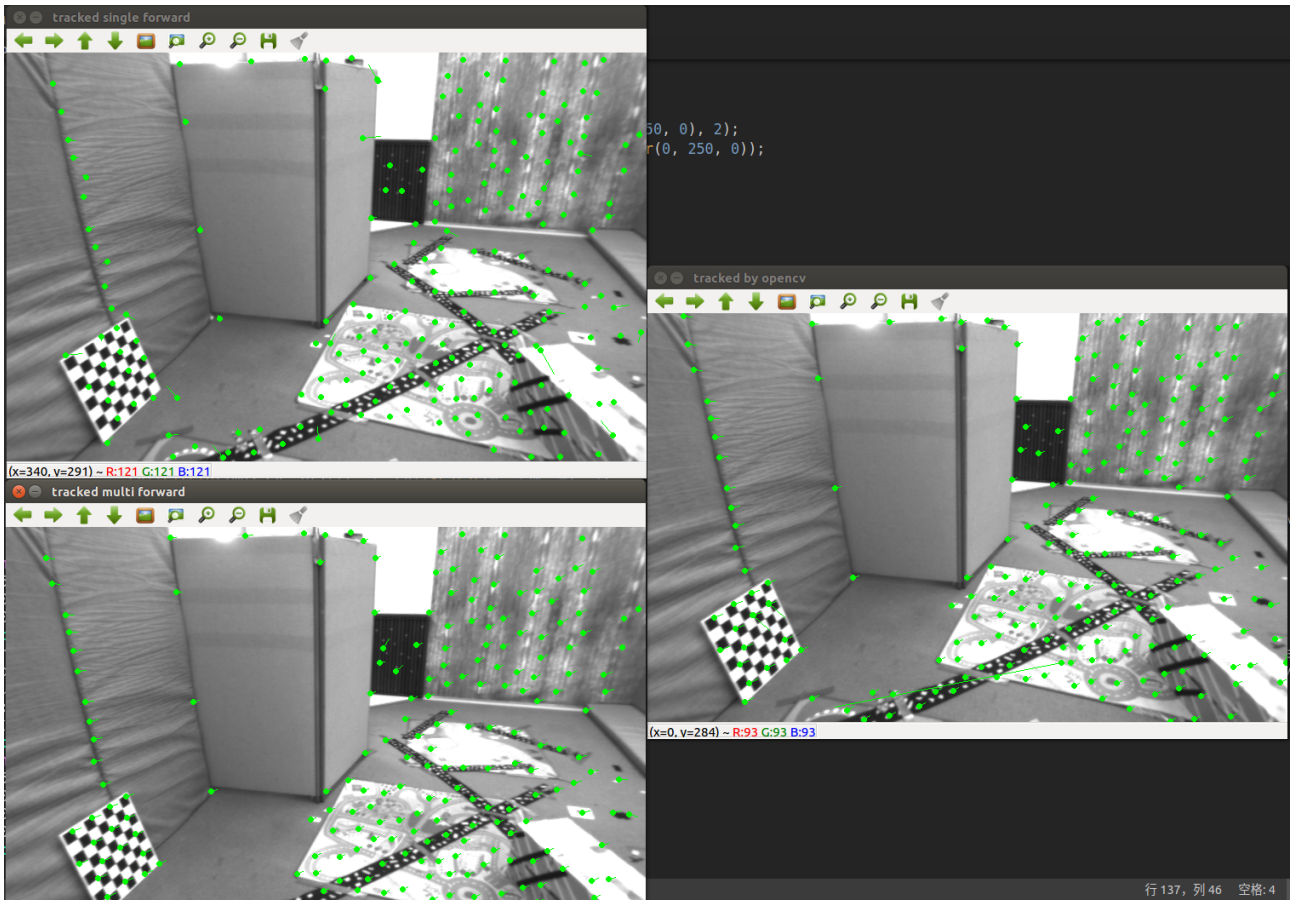
```
J(0) = -0.5 * (GetPixelValue(img2, kp.pt.x+x+1+dx, kp.pt.y+y+dy) - GetPixelValue(img2, kp.pt.x+x-1+dx, kp.pt.y+y+dy));
J(1) = -0.5 * (GetPixelValue(img2, kp.pt.x+x+dx, kp.pt.y+y+1+dy) - GetPixelValue(img2, kp.pt.x+x+dx, kp.pt.y+y-1+dy));
```

2.3 反向的光流法用 $I_1(xi,yi)$处的梯度，替换掉原本要计算的 $I_2(xi+\Delta xi, yi+\Delta yi)$

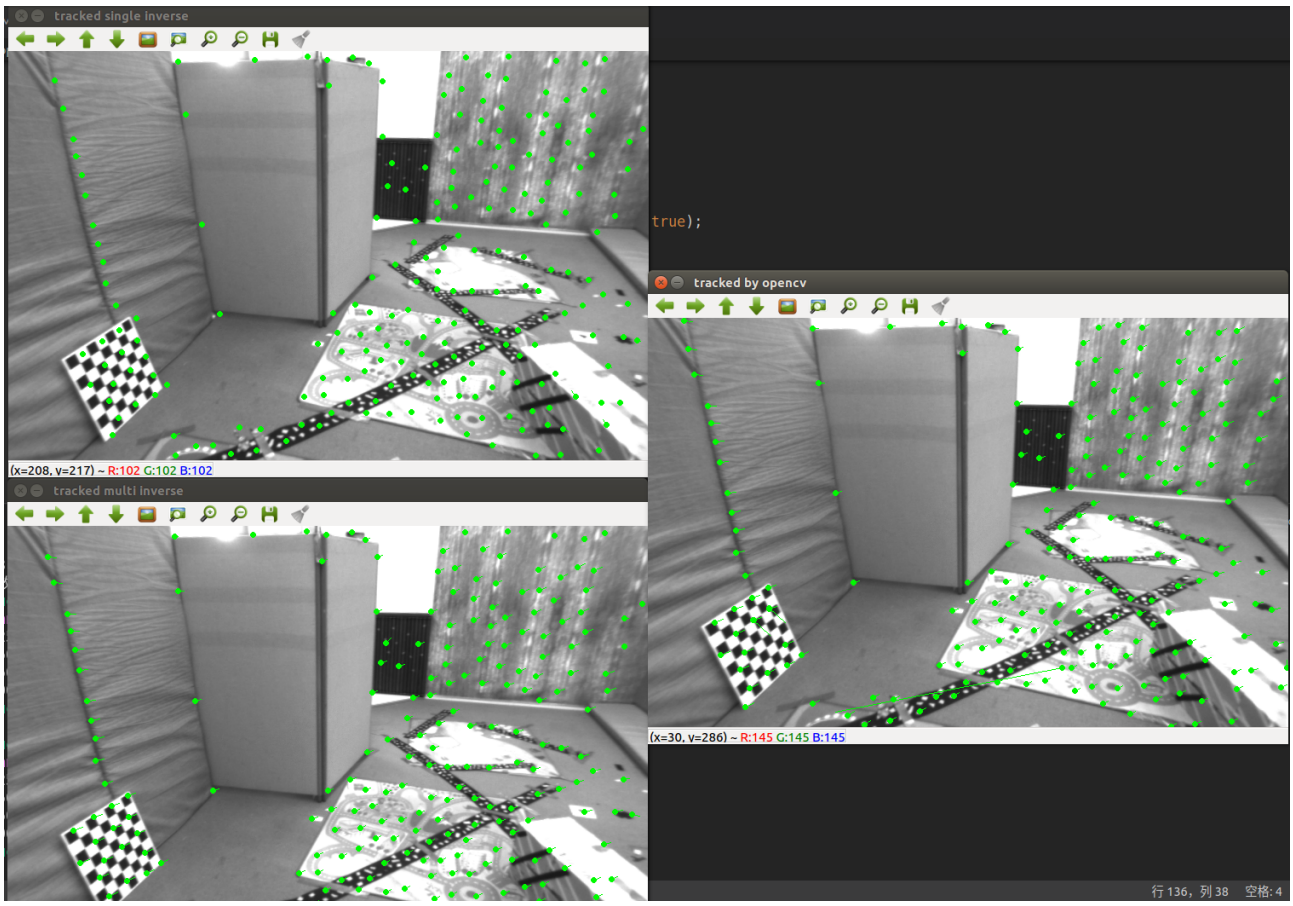
对应的雅克比矩阵计算如下

```
J(0) = -0.5 * (GetPixelValue(img1, kp.pt.x+x+1, kp.pt.y+y) - GetPixelValue(img1, kp.pt.x+x-1, kp.pt.y+y));
J(1) = -0.5 * (GetPixelValue(img1, kp.pt.x+x, kp.pt.y+y+1) - GetPixelValue(img1, kp.pt.x+x, kp.pt.y+y-1));
```

2.4 forward



inverse



从结果图看，多层光流与 OpenCV 效果相当，多层光流强于单层光流

2.4.1 coarse-to-fine 是从最粗糙的顶层金字塔开始向下迭代，不断细化估计的过程

2.4.2 特征点法的金字塔主要用于不同层级之间的匹配，以使得匹配对缩放不敏感。光流中金字塔 主要用于 coarse-to-fine 的估计

3.1.1 误差项如下

$$e = \boldsymbol{I}_1(\boldsymbol{p}_1) - \boldsymbol{I}_2(\boldsymbol{p}_2)$$

3.1.2 误差相对于自变量的维度是 1×6，求解如下

$$e(\delta T \oplus T) = I_1(p_1) - I_2(p_2 + u)$$

$$\approx I_1(p_1) - I_2(p_2) - \frac{\partial I_2}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\delta\xi$$

$$= e - \frac{\partial I_2}{\partial u}\frac{\partial u}{\partial q}\frac{\partial q}{\partial \delta\xi}\delta\xi$$

$$\boldsymbol{J} = -\frac{\partial \boldsymbol{I}_2}{\partial \boldsymbol{u}} \frac{\partial \boldsymbol{u}}{\partial \delta \boldsymbol{\xi}}. \quad \frac{\partial \boldsymbol{u}}{\partial \delta \boldsymbol{\xi}} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_x Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix}$$

3.1.3 窗口大小的选择和误差以及计算量有关系。窗口不能取太大，随着窗口的增大，两帧之间的关键点的计算量将会呈指数增长；窗口可以取单点。但是这样会导致像素梯度计算误差增大. 单个像素在两张图片里灰度不变的假设太过强烈，使用多个像素有助于提供鲁棒性。

3.1 单层直接法位姿估计如下

```
cost increased: 13036.6, 13036.6
good projection: 989
T21 =
   0.999991  0.00242132  0.00337215 -0.00184407
-0.00242871    0.999995  0.00218895   0.0026733
-0.00336684 -0.00219713    0.999992   -0.725126
          0           0           0           1
cost increased: 29782.5, 29782.5
good projection: 928
T21 =
   0.999972  0.00137282  0.00728922  0.00740815
-0.00140116    0.999991  0.00388442  -0.0013171
-0.00728382 -0.00389453    0.999966     -1.4707
          0           0           0           1
cost increased: 86163.3, 86153.7
good projection: 878
T21 =
    0.999912  0.000471702    0.0132948    -0.226133
-0.000544436     0.999985    0.0054678  -0.00180793
   -0.013292  -0.00547455     0.999897     -1.87834
           0           0           0           1
cost increased: 157690, 157619
good projection: 865
T21 =
   0.999858  0.00265099   0.0166496   -0.289507
 -0.0027373    0.999983   0.0051628   0.0221588
 -0.0166356 -0.00520765    0.999848    -2.02289
          0           0           0           1
cost increased: 167625, 167443
good projection: 759
T21 =
   0.999734  0.00162393   0.0230252    -0.40974
-0.00172887    0.999988  0.00453838   0.0629497
 -0.0230176 -0.00457698    0.999725    -2.96648
          0           0           0           1
```

3.2 多层直接法位姿估计如下

```
T21 =
   0.999991  0.00242145  0.00337204 -0.00183501
-0.00242884    0.999995  0.00218915  0.00266685
-0.00336673 -0.00219732    0.999992   -0.725144
          0           0           0           1
```

```
T21 =
   0.999972  0.00137282   0.0072892  0.00740898
-0.00140116    0.999991  0.00388443 -0.00131734
-0.00728381 -0.00389454    0.999966     -1.4707
          0           0           0           1
```

```
T21 =
   0.999937  0.00161533   0.0111188  0.00708388
-0.00167194    0.999986  0.00508421  0.00377634
 -0.0111104 -0.00510248    0.999925    -2.20888
          0           0           0           1
```

```
T21 =
    0.999874  0.000356565    0.0158908   0.00832184
-0.000447807     0.999983   0.00573866   0.00285176
  -0.0158884  -0.00574505     0.999857     -2.99611
           0           0            0            1
```

```
T21 =
   0.999803  0.00120152   0.0198237   0.0189205
-0.00133116    0.999978  0.00652743  -0.0102539
 -0.0198154 -0.00655254    0.999782    -3.79301
          0           0           0           1
```

可以看出多层直接法估计的位姿比单层直接法准确