optical flow walkthrough

This week we talked about Lucas-Kanade method to estimate the optical flow. Given two images I and J, we'd like to find a 2d displacement d = (u, v) such that

$||J(x+u, y+v) - I(x,y)||^2$

is minimized.

When d is small, we can linearize it with Taylor Expansion

$J(x+u, y+v) \approx J(x,y) + \frac{\partial J}{\partial x}u + \frac{\partial J}{\partial y}v$

$\frac{\partial J}{\partial x}$ is just the x gradient of J image, let's use J_x to stand for it. Then our goal can be rewritten as

$$||\begin{bmatrix} J_x & J_y \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} - (I(x,y) - J(x,y))||^2$$

Only one corresondence can't give us a lot information. If we assume over a small window (for instance, 5 by 5) all pixels have the same movement d = (u, v) and stack them together

$$||\begin{bmatrix} J_x(p_1) & J_y(p_1) \\ J_x(p_2) & J_y(p_2) \\ \vdots & \vdots \\ J_x(p_{25}) & J_y(p_{25}) \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} I(p_1) - J(p_1) \\ I(p_2) - J(p_2) \\ \vdots \\ I(p_{25}) - J(p_{25}) \end{bmatrix}||^2$$

This is in the form of least squares problem $||Ax - b||^2$. The close-form solution to this minimization problem is

$x = (A^TA)^{-1}A^Tb$

To get the solution x, we can also solve for the equation (known as the normal equation)

$(A^TA)x = A^Tb$

In our case, we end up solving the following equation:

$$\begin{bmatrix} \sum_i J_x(p_i)^2 & \sum_i J_x(p_i)J_y(p_i) \\ \sum_i J_y(p_i)J_x(p_i) & \sum_i J_y(p_i)^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i J_x(p_i)(I(p_i) - J(p_i)) \\ \sum_i J_y(p_i)(I(p_i) - J(p_i)) \end{bmatrix}$$

It is a 2 by 2 matrix times a 2 by 1 vector on the left hand side and a 2 by 1 vector on the right hand side. We can solve it easily with numpy function np.linalg.solve.

Sometimes for simplicity, we write $I_xu + I_yv = -I_t$

In this "equation", $I_t = J - I$, I_x, I_y are actually J_x, J_y and the equal size is an approximation. It is like $Ax = b$ but what we are actually doing is to minimize $||Ax - b||^2$.

In summary, the step to estimate optical flow is

1. compute J_x, J_y, the gradient of the second image

```
Jx, Jy = findGradient(img2)
```

2. compute $I_t = J - I$,

```
It = img2 - img1
```

3. compute the 2 by 2 matrix before (u, v) and 2 by 1 matrix on the RHS.

```
A = np.hstack((Jx.reshape(-1, 1), Jy.reshape(-1, 1)))

b = -It.reshape(-1, 1)
```

4. Solve the linear equation and get (u, v)

```
res = np.linalg.solve(A.T @ A, A.T @ b)
```

For question 2 in the homework 3, it is good enough to perform the estimaton once. In practice, we use the estimated (u,v) to move the seond image, and then start over again until the (u,v) is small enough as illustrated in the lecture.

Question 3 uses a global affine motion model where global means the model applies to the whole image.
$x + u = A11x + A12y + b1, y + v = A21x + A22y + b2$
Previously, we assume the motion is constant and is local to a small window.
$x + u = x + b1, y + v = y + b2$.

So now we need to derive an equation to estimate the paremeters for the new problem.
To do this, we can rewrite $u = (A11 - 1)x + A12y + b1, v = A21x + (A22 - 1)y + b2$.
Again we can linearize J
$J(x + u, y + v) \approx J(x, y) + \frac{\partial J(x,y)}{\partial x} . u + \frac{\partial J(x,y)}{\partial y} . v$
Use this new (u,v) in this above equation and then try to express the it in the form of $||Ax - b||^2$ where $x = [A_{11} - 1, A_{12}, A_{21}, A_{22} - 1, b_1, b_2]$ is a vector with the unknow affine model parameters. The closed form solution to this is, as already mentioned above, $x = (A^T A)^{-1} A^T b$.

For the affine motion model, the origin is at the center of the image.
In 3.1, you need to formulate the problem and find out what A matrix is. Since the motion model is global, the pooling window size is the entire image.
In 3.2, you need to solve the linear system by plugging in numbers and solve it. You are free to compute it with a program.

1D example:

0	0	1	1	1	0	0

img 1

0	1	1	1	1	1	0

img 2

Here we go through an 1D example to see how to solve parameters for affine motion model. Given two images img1 and img2, we assume (0, 0) lies at the center of the image. The x coordinates of the image look like

-3	-2	-1	0	1	2	3

x

First we compute I_x, the gradient in the x and $I_t = img2 - img1$.

0	1	0	0	0	1	0

It

0.13	0.10	0.05	0	-0.05	-0.10	-0.13

Ix (with Gaussian blurring)

With Taylor expansion as in the walkthrough, we have

$I_x u = -I_t$

With our affine model,

$x + u = ax + b$

The displacement is

$u = (a-1)x + b$

Thus we have

$I_x(a-1)x + bI_x = -I_t$

We can organize it as

$$[\, I_x x \quad I_x \,] \begin{bmatrix} a-1 \\ b \end{bmatrix} = -I_t$$

We have this constraint for each element in I_x, x and I_t. For example, the first element in $I_x (I_x[0,0])$ is 0.15, in x is -3, in I_t is 0. Thus the first constraint is

$$[\, 0.13 * -3 \quad 0.13 \,] \begin{bmatrix} a-1 \\ b \end{bmatrix} = 0.$$

All constraints form the following system of equations

$$\begin{bmatrix} -0.39 & 0.13 \\ -0.20 & 0.10 \\ -0.05 & 0.05 \\ 0 & 0 \\ -0.05 & -0.05 \\ -0.20 & -0.10 \\ -0.39 & -0.13 \end{bmatrix} \begin{bmatrix} a-1 \\ b \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ 0 \\ -1 \\ 0 \end{bmatrix}$$

To get the least squares solution, by viewing the above equations as Ax = b, now we solve for $A^T Ax = A^T b$.

$$A^T A = \begin{bmatrix} 0.40 & 0 \\ 0 & 0.06 \end{bmatrix}, A^T b = \begin{bmatrix} 0.40 \\ 0 \end{bmatrix}$$

The solution is

$$\begin{bmatrix} 1.00 \\ 0 \end{bmatrix}$$

Thus, our model of motion model is $x + u = 2.00 * x$

We can generalize it to 2D case. Consider the image 1 and image 2

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

img 1

0	0	0	0	0	0	0	0	0
0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0
0	0.5	1	1	1	1	1	0.5	0
0	0.5	1	1	1	1	1	0.5	0
0	0.5	1	1	1	1	1	0.5	0
0	0.5	1	1	1	1	1	0.5	0
0	0.5	1	1	1	1	1	0.5	0
0	0.25	0.5	0.5	0.5	0.5	0.5	0.25	0
0	0	0	0	0	0	0	0	0

img 2

The underlaying motion model is

$$\begin{bmatrix} x+u \\ y+v \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

We have x, y, I_x, I_y, I_t (I_x and I_y are graident of image 2 in x and y direction)

We have x, y, I_x, I_y, I_t (I_x and I_y are graident of image 2 in x and y direction)

-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4
-4	-3	-2	-1	0	1	2	3	4

x

-4	-4	-4	-4	-4	-4	-4	-4	-4
-3	-3	-3	-3	-3	-3	-3	-3	-3
-2	-2	-2	-2	-2	-2	-2	-2	-2
-1	-1	-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4

y

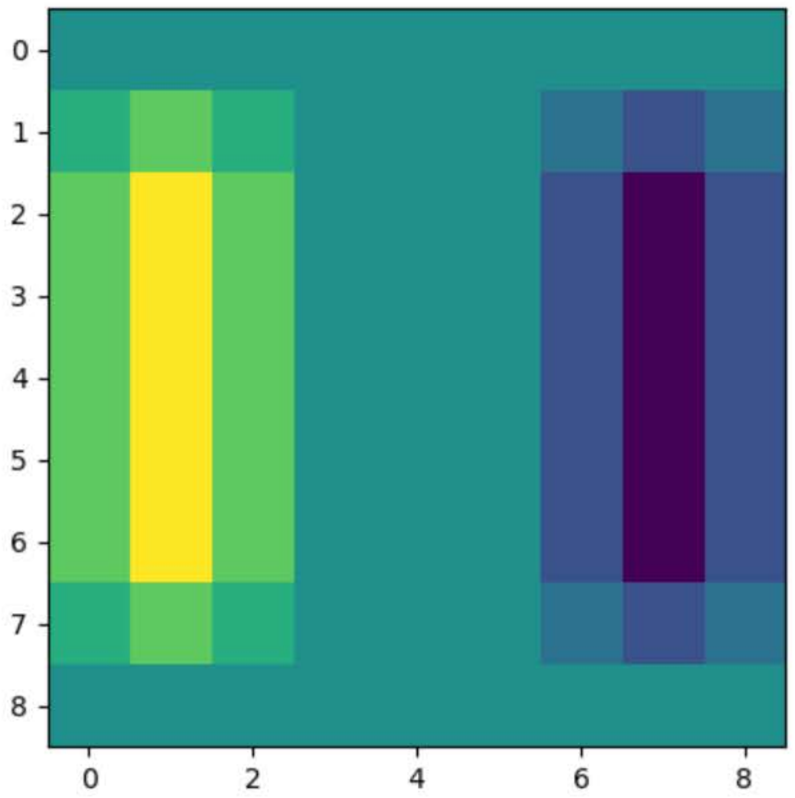
To get the I_x, I_y, besides the gradient kernel we also need to use a Gaussian Kernel with appropriate size to blur it as dicussed in the "Failure Modes of Large Displacement Optical Flow" and "Solution for Large Displacement Optical Flow Using Image Pyramid". Here we use a 11 by 11 Gaussian kernel.

If we use $I_x = img2 \otimes d_x$, following the same procedure we will get $A = \begin{bmatrix} 1.15 & 0 \\ 0 & 1.15 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. The movement is small because I_x, I_y don't spread large enough to compose the movement.

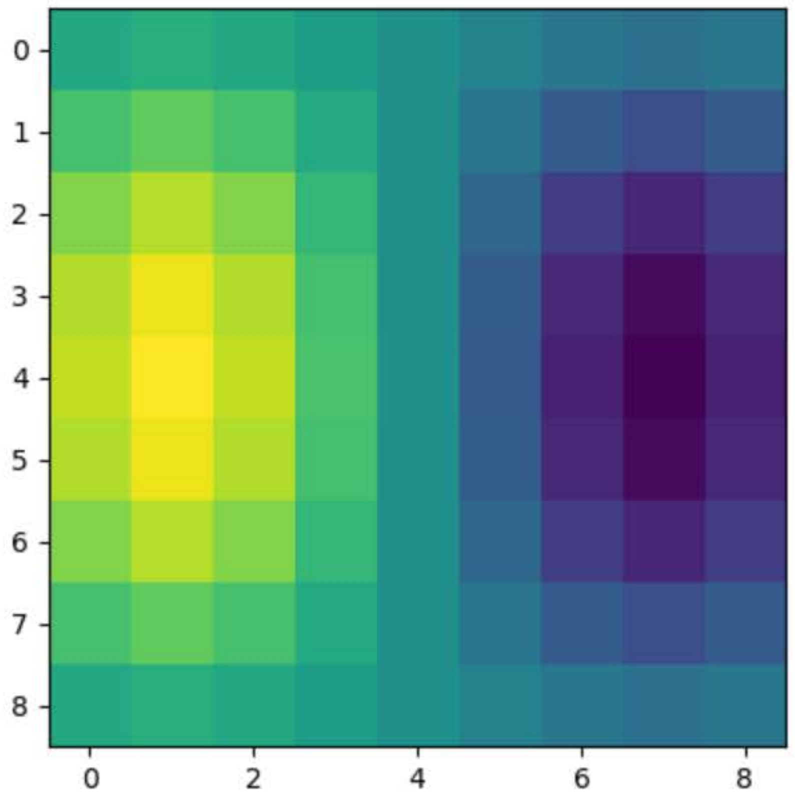
Below we are showing the I_x.

To get the I_x, I_y, besides the gradient kernel we also need to use a Gaussian Kernel with appropriate size to blur it as dicussed in the "Failure Modes of Large Displacement Optical Flow" and "Solution for Large Displacement Optical Flow Using Image Pyramid". Here we use a 11 by 11 Gaussian kernel.

If we use $I_x = img2 \otimes d_x$, following the same procedure we will get $A = \begin{bmatrix} 1.15 & 0 \\ 0 & 1.15 \end{bmatrix}, b = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$. The movement is small because I_x, I_y don't spread large enough to compose the movement. Below we are showing the I_x.



Instead, if we $I_x = img2 \otimes Gaussian \otimes d_x$, we can get a "wide spread" I_x, with this we can get a better estimation. With the Gaussian kernel, the I_x becomes



Similar to 1D case, the motion model $\begin{bmatrix} x+u \\ y+v \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ tells us
$u = (A_{11} - 1)x + A_{12}y + b_1,$
$v = A_{21}x + (A_{22} - 1)y + b_2$
We can plug them in the $I_x u + I_y v = -I_t$ and get

$$[\, I_x x \quad I_x y \quad I_x \quad I_y x \quad I_y y \quad I_y \,] \begin{bmatrix} A_{11} - 1 \\ A_{12} \\ b_1 \\ A_{21} \\ A_{22} - 1 \\ b_2 \end{bmatrix} = -I_t$$

For example, $x[0,0], y[0,0], I_x[0,0], I_y[0,0], I_t[0,0]$ are -4, -4, 0.378, 0.378, 0. Thus the constraint is

$$[\, 0.378 * -4 \quad 0.378 * -4 \quad 0.378 * -4 \quad 0.378 * -4 \quad 0.378 * -4 \quad 0.378 \,] \begin{bmatrix} A_{11} - 1 \\ A_{12} \\ b_1 \\ A_{21} \\ A_{22} - 1 \\ b_2 \end{bmatrix} = 0$$

This constraint holds for all 9 * 9 = 81 pixel locations.
We can view it as $||Ax - b||^2$ and A is 81 by 6.
To get the least squares estimate, we solve for $A^T Ax = A^T b$ and we have $A_{11} - 1 = 1.00, A_{12} = 0, b_1 = 0, A_{21} = 0, A_{22} - 1 = 1.00, b_2 = 0.$
So we have
$\begin{bmatrix} x+u \\ y+v \end{bmatrix} = \begin{bmatrix} 2.00 & 0 \\ 0 & 2.00 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$
which is close to the real model.

You could play around with this example with the code: opt_flow_affine.py
You can set the useGaussian variable at the top of the code to choose to use Gaussian blur when computing the gradient or not.