Hw3 Horn & Schunck OpticalFlow

R07941023 呂彥穎 光電所一年級

Method

1. Create the frame 2 from frame 1

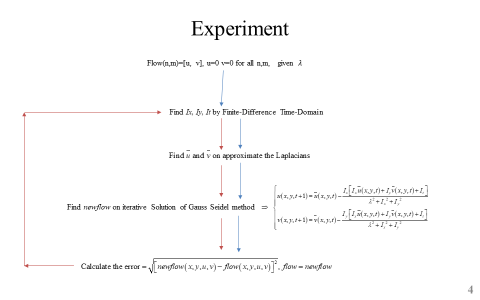
Frame1 (original frame)

Frame2 (shift one pixel to the right and downward)

Frame1 Frame2







The parameter:







Result:

=0.1



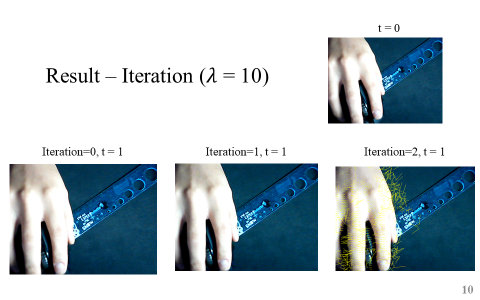
=1

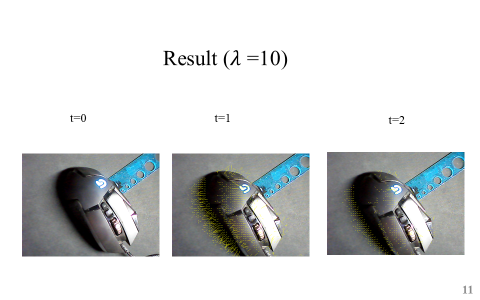


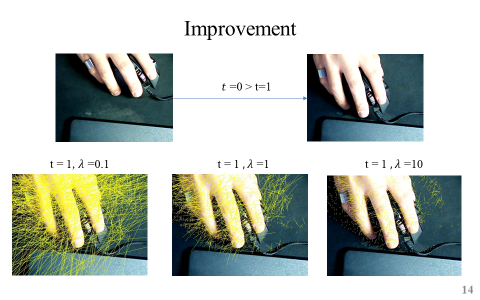
=10



Validation:







Code (python3)

import numpy as np  
import cv2  
import time  
  
def image(img1, img2):  
 step = 10  
  
 # Detect the image (gray or not)  
 try:  
 prevgray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)  
 gray = cv2.cvtColor(img2, cv2.COLOR\_BGR2GRAY)  
 except:  
 prevgray = img1  
 gray = img2  
  
 # https://docs.opencv.org/2.4/modules/video/doc/motion\_analysis\_and\_object\_tracking.html  
 ti = time.time()  
 # flow = cv2.calcOpticalFlowFarneback(prevgray, gray, None, 0.5, 3, 15, 3, 5, 1.2, 0) # Gunnar Farneback  
 flow = CalcOpticalFlowHS(prevgray, gray, flow\_lambda=0.1, iteration\_error=1e-2, convergence\_limit=10)  
 tf = time.time()  
 print('t = ', tf - ti, )  
 # Draw line  
 img = gray.copy()  
 h, w = gray.shape[:2]  
 y, x = np.mgrid[step / 2:h:step, step / 2:w:step].reshape(2, -1).astype(int)  
 fx, fy = flow[y, x].T  
 lines = np.vstack([x, y, x + fx, y + fy]).T.reshape(-1, 2, 2)  
 lines = np.int32(lines)  
 line = []  
 for l in lines:  
 if l[0][0]-l[1][0] > 3 or l[0][1]-l[1][1] > 3:  
 line.append(l)  
  
 cv2.polylines(img, line, 0, (0, 255, 255))  
 return img  
  
def CalcOpticalFlowHS(prevgray, gray, flow\_lambda, iteration\_error, convergence\_limit):  
  
 boundary\_pandding = 0  
 img\_h, img\_w = prevgray.shape[:2]  
 pooling\_matrix = np.zeros((img\_h+1, img\_w+1), np.uint8) + boundary\_pandding  
 pre\_matrix = pooling\_matrix.copy()  
 matrix = pooling\_matrix.copy()  
 pre\_matrix[0:img\_h, 0:img\_w] = prevgray  
 matrix[0:img\_h, 0:img\_w] = gray  
 # cv2.imshow('test', pre\_matrix)  
 # cv2.waitKey(0)  
  
 flow = np.zeros((img\_h, img\_w, 2))  
 newflow = np.zeros((img\_h, img\_w, 2))  
 error = iteration\_error\*2  
 count = 1  
 while (error > iteration\_error):  
 for i in range(img\_w):  
 for j in range(img\_h):  
 if i > 1 and j > 1 and i < img\_w-1 and j < img\_h-1:  
 Ex = 1 / 4 \* ((int(pre\_matrix[j, i+1]) + int(matrix[j, i+1]) + int(pre\_matrix[j+1, i+1]) + int(matrix[j+1, i+1])) - (int(pre\_matrix[j, i]) + int(matrix[j, i]) + int(pre\_matrix[j+1, i]) + int(matrix[j+1, i])))  
 Ey = 1 / 4 \* ((int(pre\_matrix[j+1, i]) + int(matrix[j+1, i]) + int(pre\_matrix[j+1, i+1]) + int(matrix[j+1, i+1])) - (int(pre\_matrix[j, i]) + int(matrix[j, i]) + int(pre\_matrix[j, i+1]) + int(matrix[j, i+1])))  
 Et = 1 / 4 \* ((int(matrix[j, i]) + int(matrix[j+1, i]) + int(matrix[j, i+1]) + int(matrix[j+1, i+1])) - (int(pre\_matrix[j, i]) + int(pre\_matrix[j+1, i]) + int(pre\_matrix[j, i+1]) + int(pre\_matrix[j+1, i+1])))  
 u0 = 1/6\*(flow[j, i-1][0] + flow[j+1, i][0] + flow[j, i+1][0] + flow[j-1, i][0]) + 1/12\*(flow[j-1, i-1][0] + flow[j+1, i+1][0] + flow[j-1, i+1][0] + flow[j+1, i-1][0])  
 v0 = 1/6\*(flow[j, i-1][1] + flow[j+1, i][1] + flow[j, i+1][1] + flow[j-1, i][1]) + 1/12\*(flow[j-1, i-1][1] + flow[j+1, i+1][1] + flow[j-1, i+1][1] + flow[j+1, i-1][1])  
 u = (u0 - Ex\*(Ex\*u0+Ey\*v0+Et)/(1+flow\_lambda\*(Ex\*\*2+Ey\*\*2)))  
 v = (v0 - Ey\*(Ex\*u0+Ey\*v0+Et)/(1+flow\_lambda\*(Ex\*\*2+Ey\*\*2)))  
 newflow[j, i] = [u, v]  
 error = (newflow-flow).reshape(-1)  
 error = np.sqrt(np.sum(np.power(error, 2)))  
 if count > convergence\_limit:  
 break  
 flow = newflow  
 print(count, error)  
 count += 1  
  
 return flow  
  
def make\_img2(img1, shiftw, shifth):  
 boundary\_pooling = 0  
 img\_h, img\_w = img1.shape[:2]  
 img2 = np.zeros((img\_h, img\_w), np.uint8) + boundary\_pooling  
 img2[shifth:img\_h, shiftw:img\_w] = img1[0:img\_h-shifth, 0:img\_w-shiftw]  
 return img2  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 img1 = cv2.imread("lena.bmp", cv2.IMREAD\_GRAYSCALE) # gray  
 img2 = make\_img2(img1, shiftw=1, shifth=1)  
 Optical\_img = image(img1, img2)  
 cv2.imshow('Optical\_img', Optical\_img)  
 cv2.imwrite('OpticalFlowHS.jpg', Optical\_img)  
 cv2.waitKey(0)