

Sec 8 (CV)

how to align images automatically?

① Feature based alignment

- Find few matching features in both images
- Compute alignment

② Direct (Pixel based) alignment

- Search for alignment where most pixels agree

Brute Force Search

① - Define image matching function

SSD, Normalize C-correlation

② - Search over all params within reasonable range

loop in all pixels in X, y axis in 2 images
Compare Image₁ pixel with image₂ pixel

Problems

- Big O is high $O(N^8)$
- Not Clear for starting value initiation and step

Solution

- use parallel search for set start value, step
- In special cases Big O = $O(N^4)$

Alternative

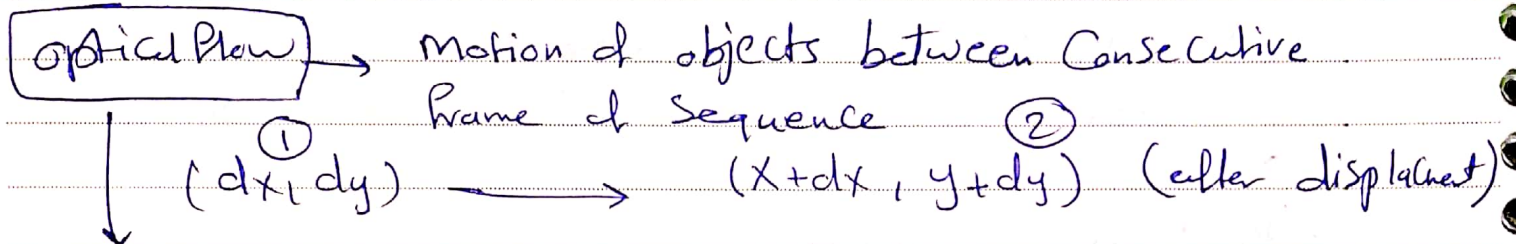
- use gradient decent on error function
- Motion Estimation

→ optical Flow

- get motion of each pixel
- get motion of entire image

Motion Estimation usages

- ① track object behavior
- ② Correct Camera jitter
- ⇒ ③ Align images (mosaics)
- ④ 3d reconstruction
- ⑤ Spatial effects

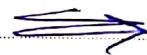


Caused by: relative movement between object and camera

- ① $I(x, y, t)$
intensity (space, time)
 - ② Apply Taylor series approximation
 - ③ divide to the change
- } Calculate of optical flow

Code

- ① Lucas Kanade method → track some point in video
- ② Find points to be tracked
- ③ CV2. `goodFeaturesToTrack()` → find points to track
- ④ Capture first frame and detect corner points
- ⑤ These points will be tracked using lucas kanade



art

lucas kanade / /

Function used

cv2.CalcOpticalFlowPyrLK (^①Previmg, ^②nextimg, ^③prevpts, ^④nextpts, [winsize, maxlevel, criteria])

Params

Previmg → First input image

nextimg → Second input image

prevpts → vector of 2d points → flow needed to be found

winsize → Size of search window at each pyramid lvl

maxlevel → 0 based maximal pyramid level number

0 → Pyramid are not used (Single level)

1 → 2 Pyramid levels are used

Criteria → specify the termination criteria of iterative search algorithm

Return

① nextpts
• output vector of 2d points
Contain
Calculated new position of input features in 2nd image

② status
Output status vector
each element of vector = 1
if flow of corresponding features are found

else = 0

③ err
output vector of errors
each element in vector = error
if cross corresponding feature wasn't found

Some Code Snippets

cv2.VideoCapture (video path) → عنه اجيب الفيديو


Params for Corner detection →

dict (① max Corners ② quality level ③ min distance
④ block size)

Params for Lucas Kanade

dict (① winSize (x,y) ② max level)

random → np.random.randint ()

Frames $\xrightarrow{\text{نحوه}}$ Gray 

cv2.goodFeatureToTrack (img , mask = none , Params of Corners)

Dense optical flow

optical flow vector for every pixel of Frame
slow speed, more accurate
used in

- ① video segmentation
- ② learning structure from motion

Dense optical flow has many implementation
but we will learn with
Franeback method

① approximate window of image by quadratic
Polynomial with help of polynomial expansion

② observing polynomial transform under state of
motion

③ Dense optical flow

Need magnitude, direction from 2d-Channel array
↓
المقدار والاتجاه → angle

Calc opticalFlow Frane back ()

Prev → First input image

next → Second input image

pyr-scale → image scale to build pyramid Scale < 1

levels → (= 1) → no extra layers on image

winSize → average window size

iteration → number of iterations

Poly-n → 5, 7

pixel-neighborhood

Polysigma \rightarrow standard deviation & gaussian deriv.

1.1 poly = 5

1.5 poly = 7

Flow
Flags \rightarrow Computed flow image