**Lab 4 – Video Enhancement**

***T1.*** *Create a denoised version of the 16th frame (i.e. noisy\_frames[15]) by naive temporal averaging of the pixels (you can use the function np.mean). Change the value of m (use 2, 4 and 6) and comment its impact on the results, identify the benefits and the pitfalls.*

Aquí hem calculat la forma sense soroll del frame 16, applicant el *naive temporal averaging*. Hem calculat el Peak Signal-to-Noise Ratio (PSNR) per veure-ho d’una manera més númerica. Per tant, podem veure que el *temporal averaging* produeix efectes no desitjats en els objectes que estan en moviment com l’efecte fantasma. Això ho podem apreciar a les imatges adjuntes que corresponen a *m = 2, 4 i 6*, on veiem que al augmentar *m,* l’efecte fantasma també augmenta, ja que la finestra és més gran i agafa major nombre de frames per aproximar el moviment.

Comparant-ho a partir del PSNR, sabem que quan el PSNR és major tenim una millor qualitat d’imatge. Per tant, s’afirma la norma suposició de que quan menor es *m* en aquest vídeo és millor, ja que té un PSNR major.

Per reduir l’efecte fantasma es podria fer calculant *l’optical flow* de la imatge i veure cap a on serà el moviment.

|  |  |  |
| --- | --- | --- |
| *m = 2*  *PSNR = 24.381* | *m = 4*  *PSNR = 23.599* | *m = 6*  *PSNR = 23.009* |

***OPTIONAL.*** *Denoise the whole video by temporal averaging.*

En aquest apartat hem tret el soroll de tot el video sencer. Ens hem basat en l’esquema següent per tal de visualitzar bé el rang a partir de *m*. Hem fet només per *m = 2*, ja que la resta de casos seria el mateix canviant *m*.



***T2.*** *As a first step towards this second naive method you will write the code that takes the 16th frame as target frame and computes the 2 previous and 2 posterior motion-compensated frames. Write the 5 frames as images and visualize them one after the other, you can also compute the difference of each frame with respect to the central (target) frame. Comment on the results.*

Tal i com hem comentat a l’apartat anterior, podem obtenir un millor resultat si compensem localment el moviment dels frames següents respecte el frame de referencia. Per això, hem estimat l’*optical flow* amb l’algoritme TV-L1 i compensem el moviment amb la funció *warp\_image\_flow* que teniem del lab 3.

A continuació podem veure el resultat obtingut fent una comparació amb els frames compensats (amb *optical flow*) i els no compensats. Podem apreciar que les imatges compensades son més similars a la imatge de referencia en respecte dels moviments dels objectes, ja que si ens fixem en les regions ocluides han reduït en respecte a les imatges no compensades.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

***T3.*** *Extend the previous code so as to denoise the frames 15th, 16th and 17th by temporal averaging in a temporal window of 2m+1 motion-compensated frames. Try different values of m and compare the result with respect to the first naive method.*

En aquest apartat, hem juntat els dos anteriors. Calculant primer l’*optical flow* i després aplicant el *temporal averaging*. Hem calculat el PSNR per cada frame ja sigui compensat o no i per cada m. Això ens ajuda a visualitzar el resultat d’una manera numèrica, veient els efectes.

L’esquema visual per aquest apartat seria el següent:

En les següents imatges podem vure el resultat visual de la imatge no compensada i la imatge comepnsada de moviment:

|  |  |
| --- | --- |

Taula dels resultats obtingut per cada *m:*

|  | PSNR m = 2 | PSNR m = 4 | PSNR m = 6 |
| --- | --- | --- | --- |
| frame 14 | 24.713 | 23.697 | 23.003 |
| frame 14 comp | 29.594 | 29.198 | 28.796 |
| frame 15 | 24.393 | 23.608 | 23.016 |
| frame 15 comp | 29.491 | 29.189 | 28.796 |
| frame 16 | 24.109 | 23.409 | 22.951 |
| frame 16 comp | 29.443 | 29.097 | 28.707 |

Mirant la taula podem comentar dues coses. Primer de tot el més obvi és que la forma compensada és molt més fidel al frame original que la no compensada.

La segona és que a mesura que augmentem el valor d’m, és a dir, apliquem l’algoritme tenint en compte més frames, el psnr disminueix. Això és degut a que el moviment de l’objecte és molt pronunciat i per tant en uns pocs frames ja canvia de posició notablement i per tant com més frames fem servir més apareix l’efecte fantasma, disminuint la qualitat de la imatge i el psnr.

***T4.*** *Compute the PSNR of the four denoised versions of frame 16th (obtained with the four different denoising methods). Compare the results and comment.*

| MULTI | SPACIAL | NAIVE | COMPENSATED |
| --- | --- | --- | --- |
| 26.8615 | 28.5013 | 23.6078 | 29.1889 |
|  |  |  |  |

El valor PSNR (Peak Signal-to-Noise Ratio) ens indica la qualitat de la tècnica de denoising, és a dir la fidelitat de la imatge denoised a la original. Com més alt és aquest valor, menys distorsió i soroll té la imatge.

Si mirem el PSNR d’un mateix frame amb diferents algoritmes de denoising podem veure que la que millor funciona és COMPENSATING. La segona millor és SPACIAL, la tercera MULTI i finalment la pitjor és NATIVE.

***T5.*** *Try and explore different values of the parameter h (h = 20, 10, 8). Apply the algorithm to both the noisy subvideo and the warped noisy subvideo. Comment and compare the results, both qualitatively and quantitatively.*

Analitzant la funció *cv2.fastNlMeansDenoisingMulti,* posant com exemple *cv2.fastNlMeansDenoisingMulti(noisy\_frames, t, T, None, h, 5, 21),* podem veure que son els diferents paràmetres. Veiem que la *t* és l'índex del frame, *T* correspon a *2m+1,* després *h* és el paràmetre que regula la força de filtre (podem relacionar-lo amb la variança), el valor de 5 és el patch per això ha de ser senar i, per últim, el valor 21, es la finestra per comparar els patches, per no agafar tota la imatge sencera i només els píxels veins.

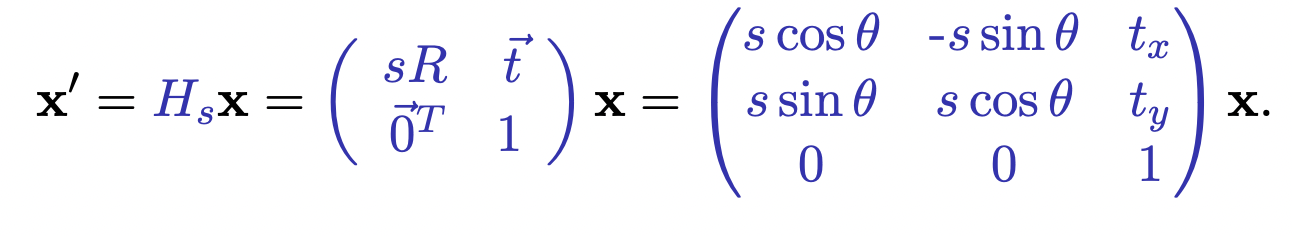
Per tant, variant el paràmetre *h* podem veure els resultats a la gràfica següent. Sabem que si el valor de *h* és més gran elimina perfectament el soroll, però també elimina els detalls de la imatge, i d’altra banda, quan més petit és el valor de *h* conserva els detalls, però també conserva el soroll.

Llavors, podem veure que en el cas no compensat, la *h* estaria més pròxima a *h = 20*, ja que es el valor més elevat dels tres casos (8,10,20). En el cas de les imatges compensades de moviment el valor més òptim d’*h* seria el 10, encara que s’hauria de mirar altres valors per ser exacte.

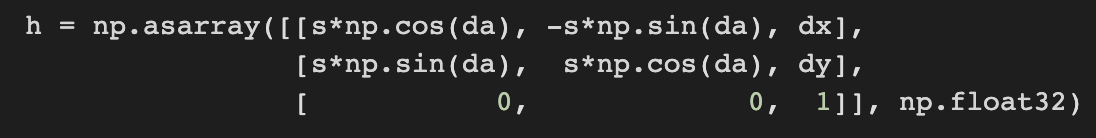
|  | *h = 8* | *h = 10* | *h = 20* |
| --- | --- | --- | --- |
| no compensated | 25.965 | 26.221 | 27.663 |
| compensated | 25.928 | 26.179 | 25.892 |

***T6.*** *Analyze the code above. Which type of 2D transformation are we applying to the frames? (Check the slides from lecture 9).*

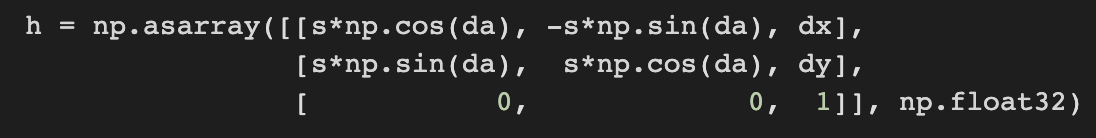
El tipus de transformació 2D que estem aplicant als frames és la transformació *Similarity*. Ho sabem perquè de teoría hem vist que la matriu *Similarity* té la forma següent:



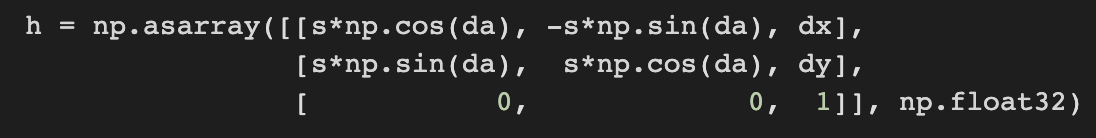
I la nostra matriu té una forma molt semblant, veie’m-ho:



En aquesta matriu tenim dues parts, primer la part de rotació + reescalació, que és la part que engloba els sinus i cosinus multiplicats per la constant *s,* seguint una distribució uniforme Z ∼ N(0, 0.01). Al codi veiem que *da* és el valor que rota la imatge, segueix una distribució normal i passa de graus a radians N ( 0, pi/180):



Per una altra banda, tenim la part de translació: *dx* i *dy*. Fent una translació de manera U(0,10):

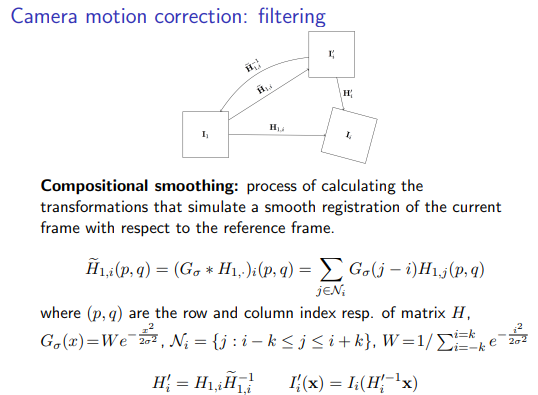


Aquesta transformació no modifica la relació de longituds, la relació de areas i els angles. Bàsicament rota, escala i mou el frame per estabilitzar el video.

Per tant, concluïm amb que la nostra matriu és de tipus *Similarity*.

***T7.*** *Complete the function below. The part that needs to be completed is the one that computes h, the stabilizing homography for every frame. In order to do the Gaussian smoothing you can use the function above.*

En aquest apartat hem completat el codi, seguint com exemple aquesta diapositiva de la teoria.



Primer calculem les transformacions que compensen les imatges tenint en compte el frame de referència. Aquestes transformacions les obtenim mediant el moviment relatiu entre frames successius. Un cop tenim aquestes transformacions, apliquem un low-pass filter com un filtre Gaussià per assegurar-nos de tenir transicions suaus.

Això ho fem per tots els parells de frames i sintetitzem un nou vídeo amb aquests frames estabilitzats.

***T8.*** *Complete the function below that displays the non-stabilized and the stabilized trajectories of the central pixel*

Per poder mostrar les trajectòries primer de tot creem les llistes on les guardarem amb els valors x i y.

Un cop les tenim les inicialitzem amb el pixel inicial i després per cada frame calculem la trajectòria mediant les transformacions calculades anteriorment i guardem aquest nous valors a les llistes. Amb la no estabilitzada per un costat i la estabilitzada per l’altre.

Finalment simplement fem dos plots visualitzant aquestes llistes: