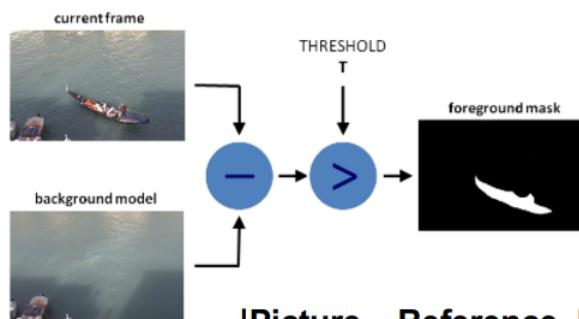


LAB 2 - Video Segmentation

TASK 1: Background Subtraction

- a. L'objectiu d'aquest exercici és detectar l'objecte que es mou a les diferents imatges i calcular la trajectòria que segueix. En aquest cas, es tracta d'un ocell que vola i volem veure el desplaçament que fa. Per fer-ho, implementem la funció `extract_object()` que detecta el fons de les imatges a partir de l'algorisme vist a classe:



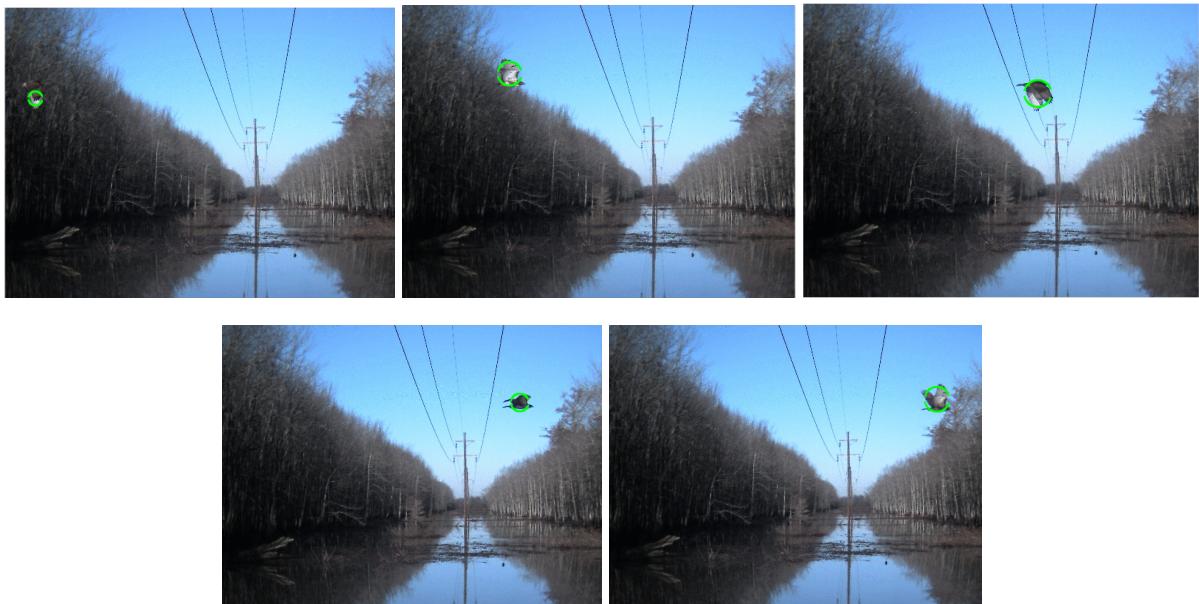
|Picture – Reference_Picture| > threshold

Algorisme d'extracció del fons

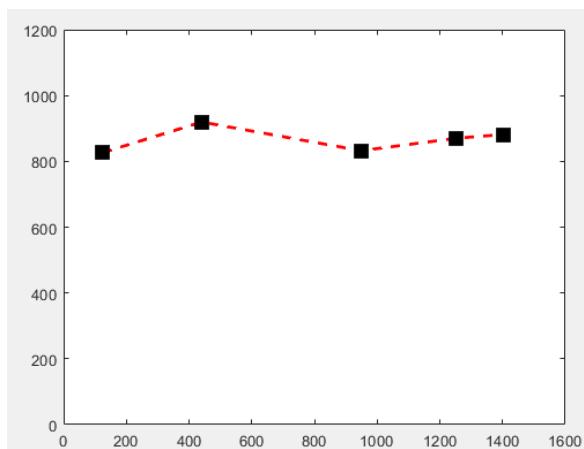
Això fa que, a partir d'una imatge, si li restem el seu fons i imposem que sigui major a un umbral (threshold) determinat, trobem on està situat l'objecte que buscavem.

Aquesta fórmula la implementem a la funció `extract_object()` fent una suma de l'absolut de la diferencia entre `input_image` i `image_template`) dels tres canals R,G,B de la imatge i així trobar l'objecte a cadascun dels frames.

Un cop aquesta funció està llena, anem al `main()`. El primer pas serà trobar el fons de la imatge sense ocell que necessitem per l'algorisme. Aquest fons l'aconseguim mirant quines imatges no tenen ocell, és a dir, en quines imatges es veu clarament el fons, i fent la mitjana aritmètica d'aquestes imatges. Quan tenim aquest fons lleny, ja el podem utilitzar a l'algorisme. Posem el threshold igual a 10 i executem la funció i veiem que ens detecta perfectament l'ocell a totes les imatges:

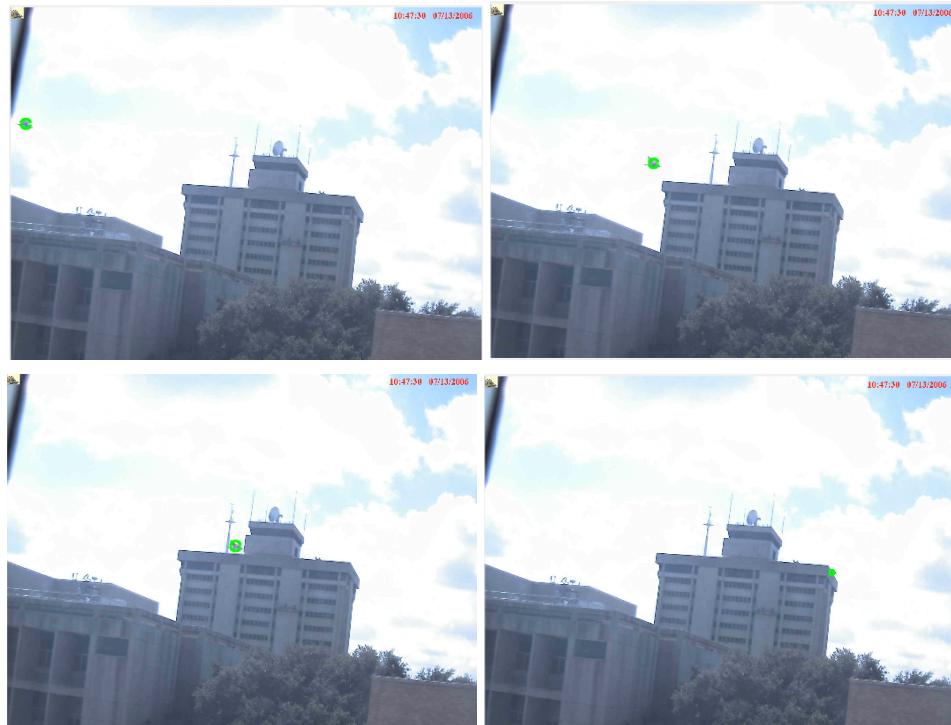


També veiem el recorregut de l'ocell, que és el següent:



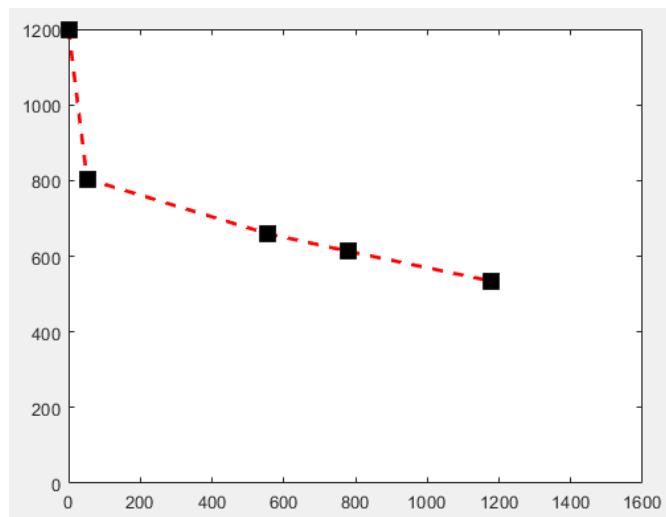
Trajectòria de l'ocell detectat per l'algorisme

- b. Ara farem el mateix però en comptes de tener unes imatges determinades, tenim un vídeo on un ocell vola sobre uns edificis. Hem de seguir un procés semblant a l'anterior, en aquest cas comencem llegint el video i fent la mediana dels frames per tal d'aconseguir una imatge aproximada del fons del video.
- Després, al *for* escollim en quins frames volem detectar l'ocell i executem. Veiem un exemple:



Detecció de l'ocell als frames del 15 al 18

Aquesta trajectòria es veuria així:



Trajectòria de l'ocell als frames del 15 al 18

Veiem que encaixa amb la selecció de l'ocell que fa l'algorisme.

TASK 2: Graph-Cut Algorithm

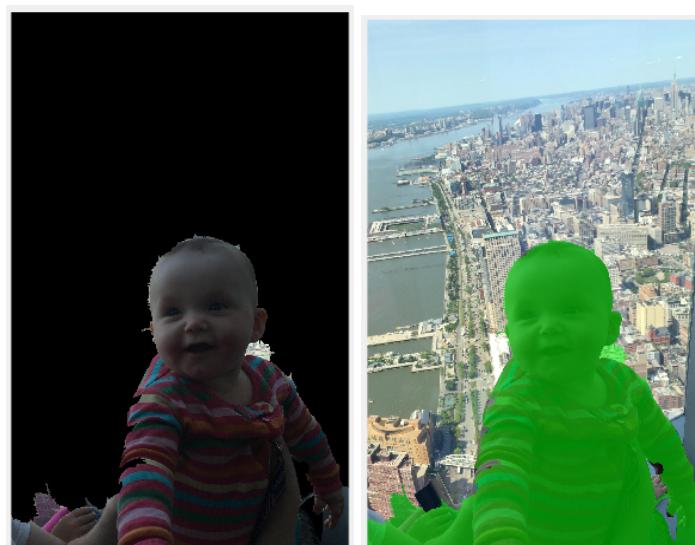
- a. Executem l'algorisme donat amb dues imatges diferents. Aquest algorisme el que fa és, demanar a l'usuari que, primer seleccioni el objecte (o objectes) principal de la imatge i després el fons. Un cop ho seleccionem, crea una imatge amb l'objecte principal sense fons. Veie'm-ho:

Seleccionem l'objecte principal (verd), per tal de que surti el més exacte possible intentem no agafar cap valor del fons, simplement l'objecte, i el fons (vermell) de la imatge, sense agafar cap valor de l'objecte.



Imatge baby y peppers passant pel procés de selecció d'àrees

Un cop o tenim seleccionat ens crea una imatge de l'objecte principal sense fons, d'una manera força exacte.



Imatge baby després de passar per l'algorisme



Imatge peppers després de passar per l'algorisme

Veiem que aquest procés separa força bé l'objecte del fons de la imatge sense tenir molta interacció, ja que només hem seleccionar dues areas i la resta ho fa sol.

- b. Ara utilitzarem l'aplicació *Image Segmenter* i l'opció de *Graph-cut* per extreure l'objecte principal de la imatge i comparar-ho amb l'algorisme anterior. Primer, seleccionem la imatge que volem analitzar i després hem de "pintar" a sobre de l'objecte i del fons per que pugui extreure l'objecte principal del fons,



Imatge peppers i baby en la selecció de foreground i background

Un cop les seleccionem, creem la màscara i es veu de la següent manera:



Màscara de les imatges peppers i baby

Observem que és un resultat bo ja que la màscara s'aproxima molt a l'objecte original. Si ho comparem amb l'algorisme anterior creiem que en aquest procés es necessita més interacció ja que has d'anar pintant les zones que no es seleccionen bé, en canvi, a l'altre algorisme només fas un parell de seleccions i crea la màscara perfectament.

- c. En aquest apartat, hem d'agafar les imatges de *camel* i *flamingo*, i utilitzar el mètode del primer apartat per tal de fer la segmentació.



Imatge *camel* y *flamingo* passant pel procés de selecció d'àrees

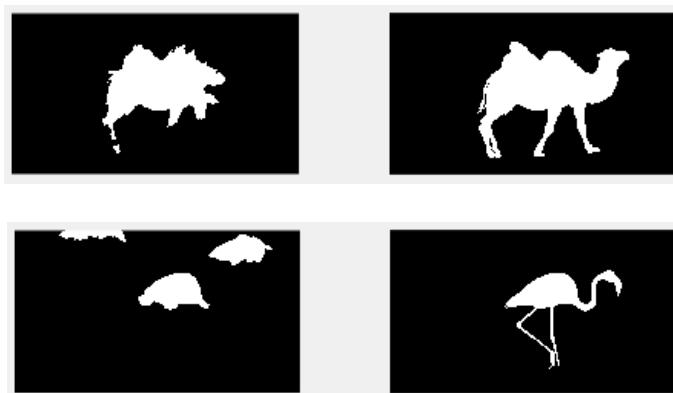
Després de fer la selecció, comparem la imatge de la segmentació que hem fet amb el *ground truth* de la imatge corresponent.

En el cas del *camel*, podem veure dos tipus d'error:

El immse = 3.6642e+03 que quan més elevat pitjor, i el psnr = 12.4911, quan més elevat millor resultat obtindrem.

En el cas del *camel*, tenim el immse = 3.7067e+03 que quan més elevat pitjor, i el psnr = 12.4410, quan més elevat millor resultat obtindrem.

Aquí podem veure la comparació amb el *ground truth*:



Comparació amb el *ground truth*

TASK 3: Subspace Clustering

En aquest apartat hem tingut més problemes, ja al començament ens demana que convertim de C a mex per tal de que matlab pugui llegir l'arxiu ho hem fet a partir de la comanda mex filename.

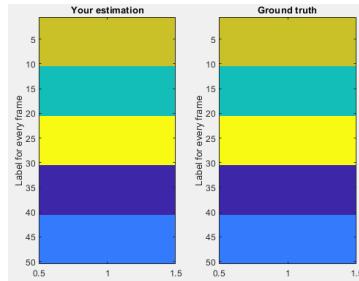
Després se'ns demana que realitzem filtres de segon i quart ordre, nosaltres a teoria hem vist tal que això:

$$\mathbf{R}_1 = \begin{bmatrix} -1 & & & \\ 1 & -1 & & \\ & 1 & -1 & \\ & & \ddots & \ddots \\ & & & 1 \end{bmatrix} \quad \mathbf{R}_4 = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & -16 & 16 & 16 \\ -1 & 16 & -30 & -16 \\ & -1 & 16 & -30 \\ & & 16 & 16 \\ & & & -1 \\ & & & 1 \\ & & & 0 \end{bmatrix}$$

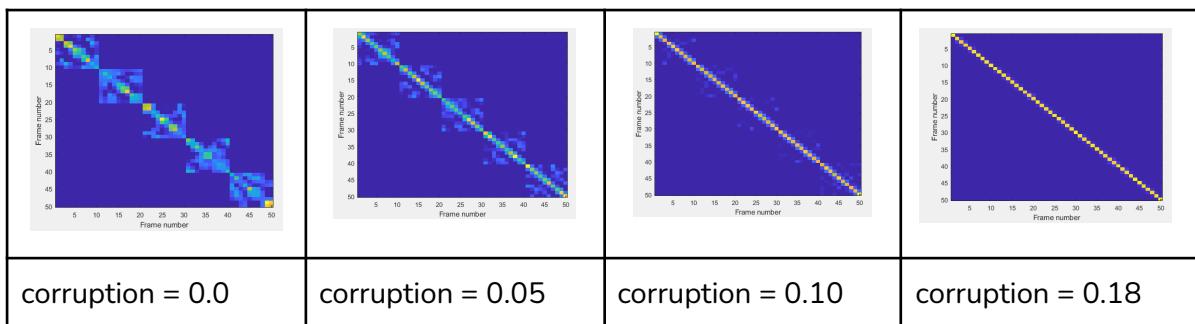
$$\mathbf{R}_2 = \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 2 & 2 & -1 \\ -1 & -1 & 2 & -1 \\ & \ddots & \ddots & \ddots \\ & & -1 & 1 \end{bmatrix}$$

per tant, hem realitzat la matriu R_2 i R_4 , a partir de la funció diag de Matlab.

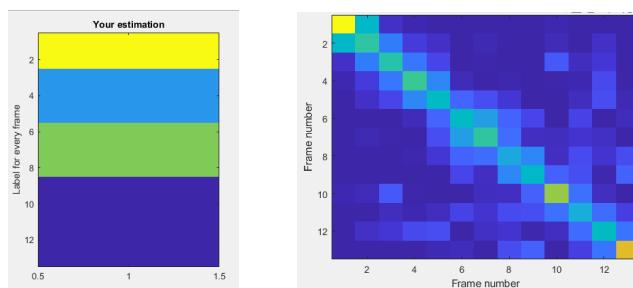
Hem aplicat diferents nivells de *corruption*, i així hem pogut observar com es comportava.



La nostra estimació i el ground truth



En l'altre cas, utilitzem les imatges proposades i li apliquem els diferents filtres.



La nostra estimació amb filtre = 2