Maud Buffier

Controllable Heliotrope Video

Lab6

Basic section

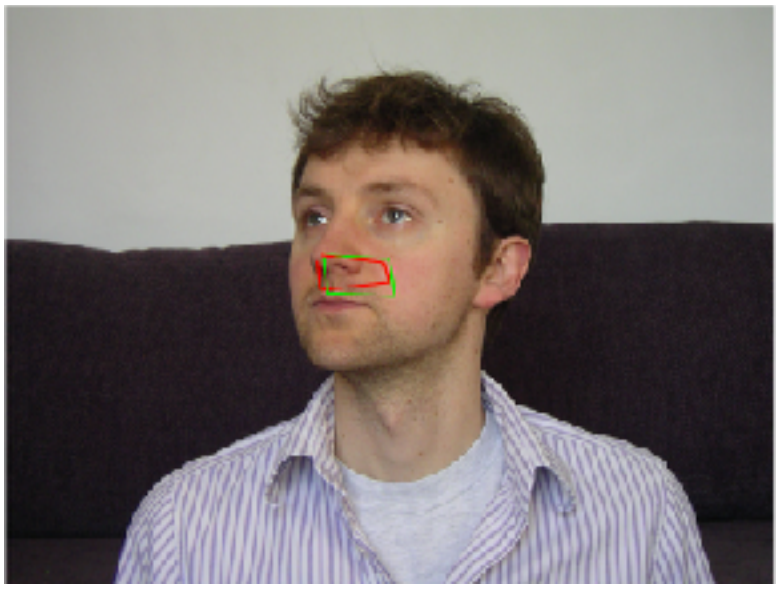
You can run my the basic section (extended directly with the first (distance using a trajectory) and the third items of the advanced section (interpolation between nodes) :

- Run lab6_adv('gjbLookAtTargets', 'gjbLookAtTarget', 0, 71, 'jpg')
- Then the program indicates at which step it is. Distance matrices are already computed and are only load each time. (we can see my implementation to compute the distance matrices at the bottom of the file)
- However, the flows was too heavy so I couldn't include it in my report
- Then, the program asks you to choose a starting frame
- This frame is displayed and you can chose a path for the nose
- You can select the path you desire by clicking on the image. However, you need to choose a trajectory coherent with the images provided because the result will depend on the presence or not of the asked position in the frames
- The program is computing the best path between frames using the algorithm describe in the lab
- After computation of the paths, an image is displayed showing the asked trajectory (in green) and the best trajectory possible with the images provided (in red)
- The program asks you if you want to interpolate using synthesized images (you need to answer 0 or 1)
- It also asks how many frames do you want to create between existing ones
- The algorithm synthesized the frames, and store the result both in a video (in resultVideo but not included in the report) and in a sequence of images (in resultImages)

Interesting points for the basic section :

- Look for directions and not positions

I decided to look for the « best direction » rather than « the best position » in the lab when computing my paths. That way, the directions asked by the user are respected in the best way possible and the nose movement is the best one. Indeed, if I would have taken the asked positions into account to compute my paths, it would give weird result if the previous position isn't reached properly by the system and the obtained movements could be very different from the asked one.

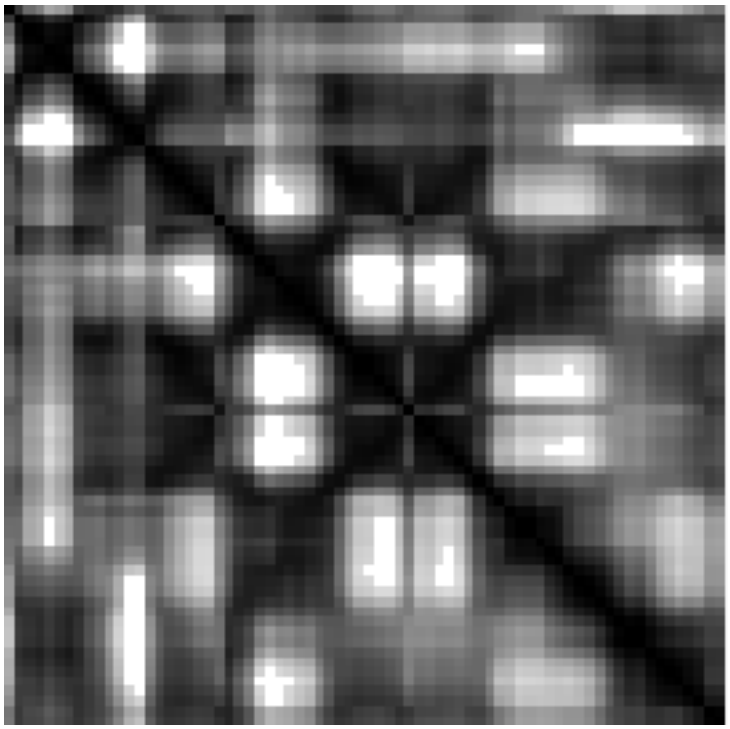


Here is an example of a best trajectory possible :

- Create a graph sufficiently connected but not too dense

In order to create a graph sufficiently connected (avoiding having 2 or 3 islands only connected by an edge for example) I computed a graph using all the distances in my distance matrix. I applied the «graphminspantree » function to obtain the minimum spanning tree over the whole graph to assure no isolated island. Then, I threshold my distance matrix to eliminate high distances not relevant for our application. I chose the threshold by experimenting the distance matrix I obtained. I chose one which kept sufficient connections between images. After created full matrices with those graphs, I combined them using the « or » operator. Hence, with this final matrix I computed an undirected graph using the « trill » matlab function.

Here is the distance matrix (already taking into account the trajectories) :



The lower the distance is, the darker it appears.

- Multiplying the flows

As the size of the flow images are reduced to 30%, I rescaled and multiplied them by 1/0.3 to obtain the movements (described in pixel) in the same scale as the images provided.

Maud Buffier

Advanced section

Updated distance matrix :

To update the distance matrix and taking into account the trajectory of the images I used the flows with respect to one reference frame (the first one). Indeed, to see if any two images are moving similarly, we need a fixed frame to compare the movement of the first image with respect to the fixed one and the movement of the second also with respect to the fixed one.
So, in the function where I computed the distance, I added a dimension in addition of the R, G, B channels using the flow matrices. Is the flows with respect to the fixed frame is similar, the 4th dimension of the distance will be low.
The distance becomes :

$$dist = sqrt(dist\_R^2 + dist\_G^2 + dist\_B^2 + dist\_Flow^2)$$

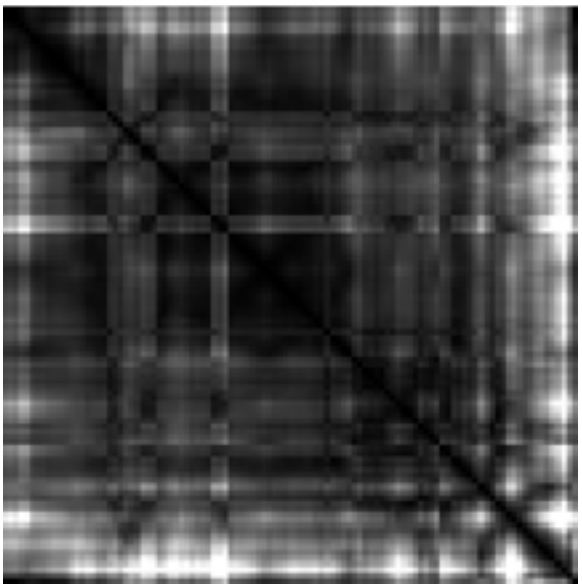
where dist_i is computed by taking the sum of the squared differences between the channels or the flows of the 2 images we are looking the distance between.

New data :

I first try to do something a bit creative using a smiley like that :

which took different expressions. However, as the flow were non zero at too few places, the matches were not good and it didn't work.
Hence, I gathered my own real data :

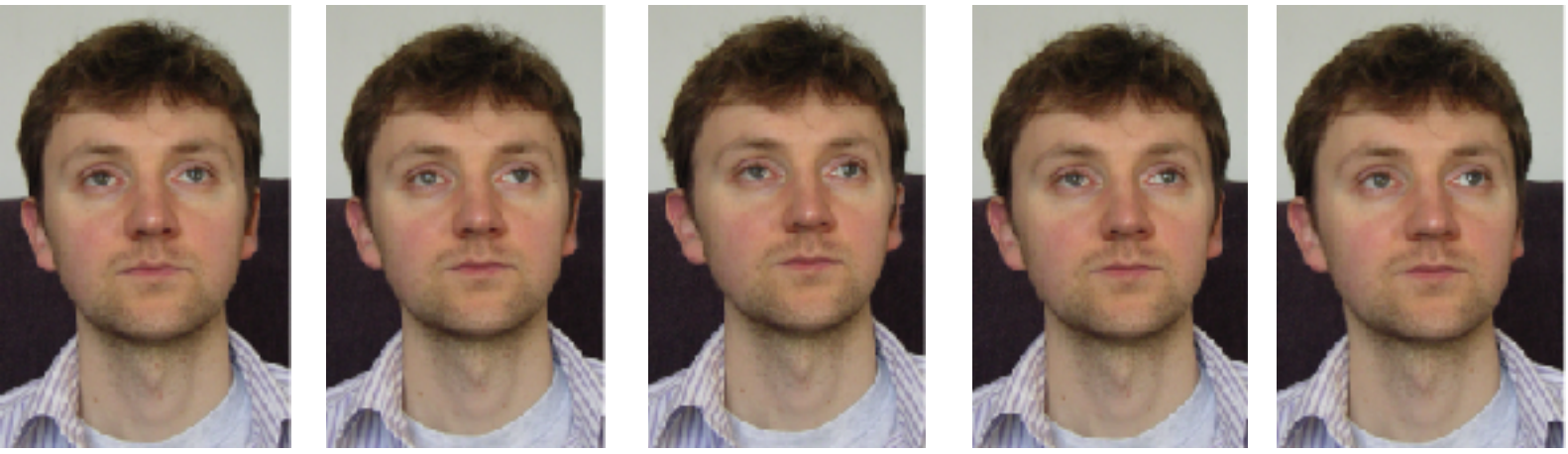
Where you can choose how my chignon (to change from a face) is moving.
I computed a function in the repository « opticalFlow » called « computeFlow » to be able to obtain the flow between images. To avoid this operation to last for ever, I rescaled my images to 493*327 size and put them in gray scale to compute the flows. My indices are computed the same way as you.
Here is my distance image :



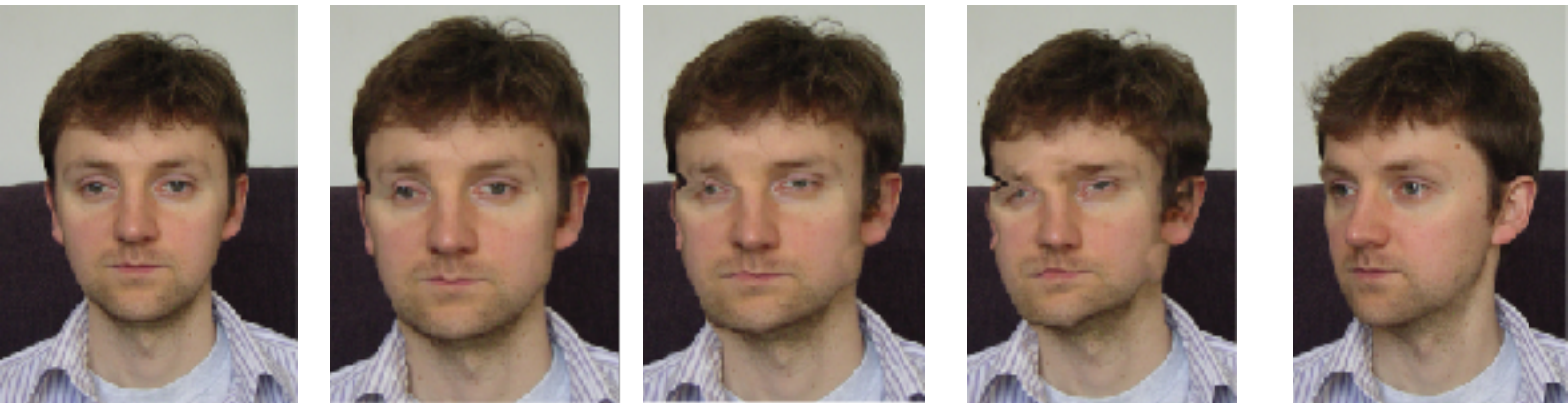
We can run the function using the file 'lab6_OwnData.m' by running lab6_OwnData('ownData_Gray', 'newData_0', 0, 104, 3, 'png') and then the process is the same as for the basic section. The result with the image showing how accurate the reconstruction is, is in the 'ownData_result' repository.

Render slow motion interpolation

To do this task, I used the function « warpFLColor » in the opticalFlow. Indeed, after computing the best path, I can find the flow between 2 successive images and divide this flow by the number of frames asked by the user. Then, I used the function to wrap the first image by the right amount of flow.
It works pretty well when images are closed :



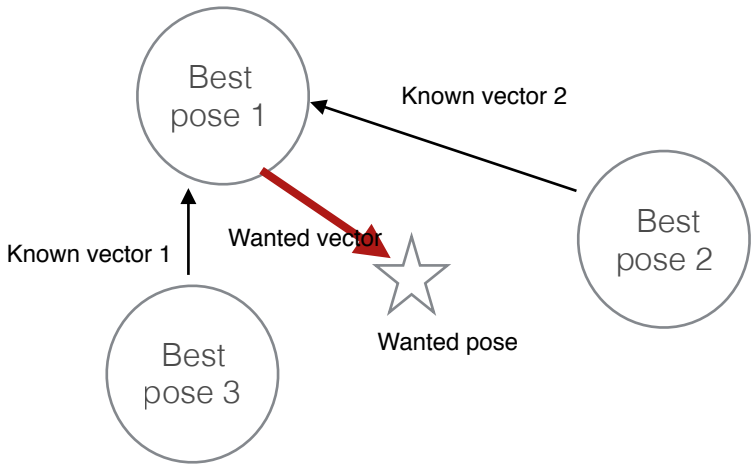
Despite some discret artifacts on the hear and on around the head the result is really convincing.
However, when the images are too different from the start, it gives lots of artifacts on the synthetic images. For example :



We can see the frames to interpolate between are not very similar. To avoid this artifact I smoothed the flow with a gaussian with a standard deviation of 30 to avoid abrupt changes and a smoother movement.
Then, artifacts are less visible but still important in very distinctive frames.
I think a good way to avoid that would be to take more pictures to obtain paths with smaller movements between frames.
I also cropped a bit my resulting sequence to not see black strips on the bottom of the images appearing because of the warp function. Indeed, as the motion is only in one direction, all the image can be shift a bit using wrap.

Multi-node interpolation

To do this task I also use the wrapFl_color function from the optical flow librairy and I reconstruct the wanted flow from a linear combination of 2 others.



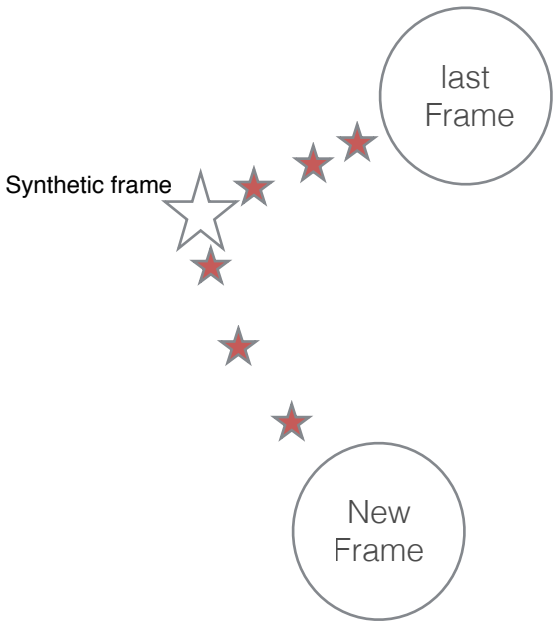
I know the 3 best position estimations and the wanted position. This way, I can write a equation to find how reconstruct the vector between the best pose and the wanted pose

using the 2 vectors from the best pose 3 and 2 to the best pose 1 (2 unknowns, 2 equations). Then, in the same way, I can express the wanted flow in function of the flows between 3-1 and 2-1 using the same coefficients found in the previous system.
As I can do that for the complete vector, I can also find the coefficients for half the vector or even the proportions the user asked.
Once I know the coefficients, I compute the new flow by linear combination and apply it to the frame with the best position. This way,I will make the given point arrives to the wanted position.

It works well a long as the wanted position is not too far from the best position.

You can test the algorithm using the 'lab6_multi_node_inter(path, prefix, first, last, digit, suffix)' function.

However, I computed the algorithm only for a single direction asked by the user. Indeed, as the result is a synthetic frame, it doesn't appear in the distance matrix function. Hence, to have a smooth effect I should have had computed the distance matrix again adding the new synthetic frame to be able to find a new path to arrive to the second point asked by the user :



As the scheme shows, I needed to compute a path between the synthetic frame and all the other one to find the best one.

So, this file is only working for 2 points (1 direction) of movement.
However, the program asks you how many frames do you want to interpolate between the images in the path and also, at the end, for the new synthetic pose.

All the results are in the result folders with an image showing the asked direction.

I did not do the automated system.