Optical Flow Estimation and Feature Extraction in Video for Action Recognition

Part 1: Face and Hand Monitoring using the Lucas-Kanade Optical Flow Method

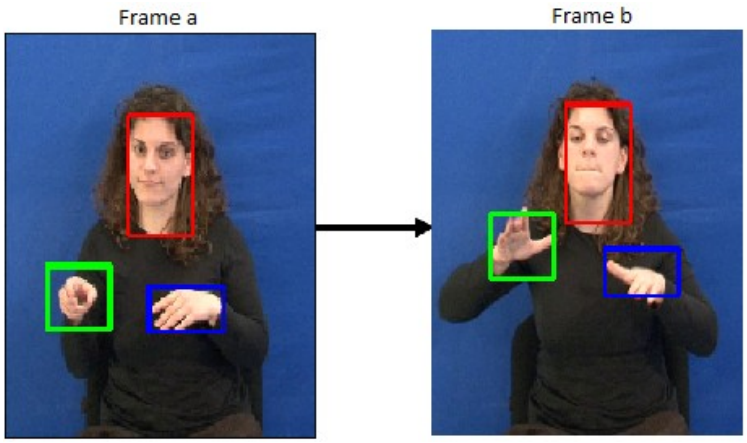

Figure 1: Example of face-hand tracking in a sign language video sequence.

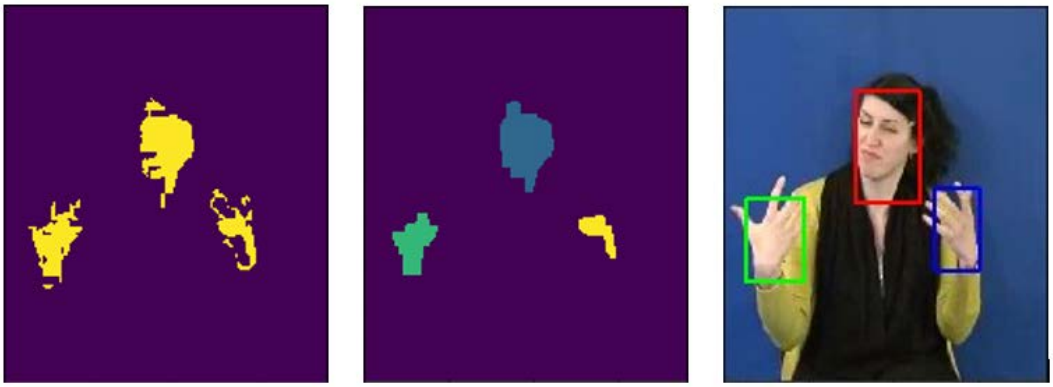

Figure 2: Example of skin detection in sign language video: (a) Skin mark detection. (b) Formation of coherent regions. (c) Final face area detection.

1.1. Face and Hand Skin Detection

The first question asks for the detection of skin spots in the first frame of the sequence and the final selection of the face and hands area. The color space YCbCr is used to detect skin spots by removing the luminance information Y and keeping the Cb and Cr channels describing the color identity. Skin color is modeled with a two-dimensional Gaussian distribution:

$$P(\mathbf{c} = \text{skin}) = \frac{1}{\sqrt{|\Sigma|\,(2\pi)^2}} e^{-\frac{1}{2}(\mathbf{c}-\mu)\Sigma^{-1}(\mathbf{c}-\mu)'}$$

where c is the vector of values Cb and Cr for each point (x, y) of the image.

1.2. *Face and Hand Monitoring*

The initialization of the Bounding Boxes containing the face and hands of the signer in the form [x, y, width, height] is as follows:
Face: [138, 88, 73, 123], Left hand: [47, 243, 71, 66], Right hand: [162, 264, 83, 48].

1.2.1 Lucas-Kanade Algorithm Implementation

Implement the Lucas-Kanade algorithm in Python. The algorithm is to be implemented as a stand-alone function, taking as inputs two images (a set of windows based on the bounding box from two consecutive frames of the video), a set of points of interest (angles) within the window, the range ρ of the Gaussian window, the positive normalization constant E, and the initial estimate d_0 for the optical flow field, and returning d.

In a sequence of N frames $I_n(x)$, where n = 1, . . , N and x = (x, y), the optical flow field -d, where d(x) = (dx, dy), brings two consecutive images into correspondence such that

$$I_n(\mathbf{x}) \approx I_{n-1}(\mathbf{x}+\mathbf{d})$$

The Lucas-Kanade algorithm calculates the optical flow at each point in the image x by the method of least squares, assuming that d is constant in a small window around the point and minimizing the squared error

$$J_{\mathbf{x}}(\mathbf{d}) = \int_{\mathbf{x}'\in\mathbb{R}^2} G_\rho(\mathbf{x}-\mathbf{x}')\left[I_n(\mathbf{x}') - I_{n-1}(\mathbf{x}'+\mathbf{d})\right]^2 \mathrm{d}\mathbf{x}',$$

where $G_\rho(x)$ is a windowing function, e.g. Gaussian with standard deviation ρ.

We assume that we have an estimate d_i for d and try to improve it by u, i.e. $d_{i+1} = d_i + u$. By Taylor expanding the expression $I_{n-1}(x+d) = I_{n-1}(x+d_i+u)$ around the point $x + d_i$, we obtain

$$I_{n-1}(\mathbf{x}+\mathbf{d}) \approx I_{n-1}(\mathbf{x}+\mathbf{d}_i) + \nabla I_{n-1}(\mathbf{x}+\mathbf{d}_i)^T\mathbf{u}$$

Thus, the least squares solution to improve the estimation of the optical flow at each point is

$$\mathbf{u}(\mathbf{x}) = \begin{bmatrix} (G_\rho * A_1^2)(\mathbf{x}) + \epsilon & (G_\rho * (A_1A_2))(\mathbf{x}) \\ (G_\rho * (A_1A_2))(\mathbf{x}) & (G_\rho * A_2^2)(\mathbf{x}) + \epsilon \end{bmatrix}^{-1} \cdot \begin{bmatrix} (G_\rho * (A_1E))(\mathbf{x}) \\ (G_\rho * (A_2E))(\mathbf{x}) \end{bmatrix}$$

where

$$A(\mathbf{x}) = \begin{bmatrix} A_1(\mathbf{x}) & A_2(\mathbf{x}) \end{bmatrix} = \begin{bmatrix} \dfrac{\partial I_{n-1}(\mathbf{x}+\mathbf{d}_i)}{\partial x} & \dfrac{\partial I_{n-1}(\mathbf{x}+\mathbf{d}_i)}{\partial y} \end{bmatrix}$$
$$E(\mathbf{x}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x}+\mathbf{d}_i)$$

and $*$ denotes convolution. The small positive constant E improves the result in flat areas of solid texture and thus solid information for the optical flow calculation. The renewal of the

optical flow vector $d_{i+1} = d_i + u$, with u calculated from its equation, is repeated several times until convergence.

1.2.2 Calculation of Window Displacement from Optical Flow Vectors

Having calculated the optical flow of the image I_n at the points defining the points of interest within the bounding box of image I_{n-1}, it is necessary to find the total displacement vector of the bounding box rectangle, with as much accuracy as possible. We know that optical flow vectors generally have a larger length at points belonging to regions with strong texture information (e.g. edges, vertices) and almost zero length at points belonging to regions with sharp and flat texture. Thus, as the majority of the points of interest (corners) are located in strongly textured areas, we could simply use the mean value of the displacement vectors. However, in order to achieve better accuracy or to reject outliers, alternative criteria could be applied, such as, for example, calculating the mean value of the shift vectors that have an energy higher than a threshold value. For the energy of a velocity vector, we define $\|\mathrm{d}\|^2 = \mathrm{d}_\mathrm{x}^2 + \mathrm{d}_\mathrm{y}^2$.

Implement a function that takes as input the vectors of the optical flow and calculates the final displacement vector of the rectangle.

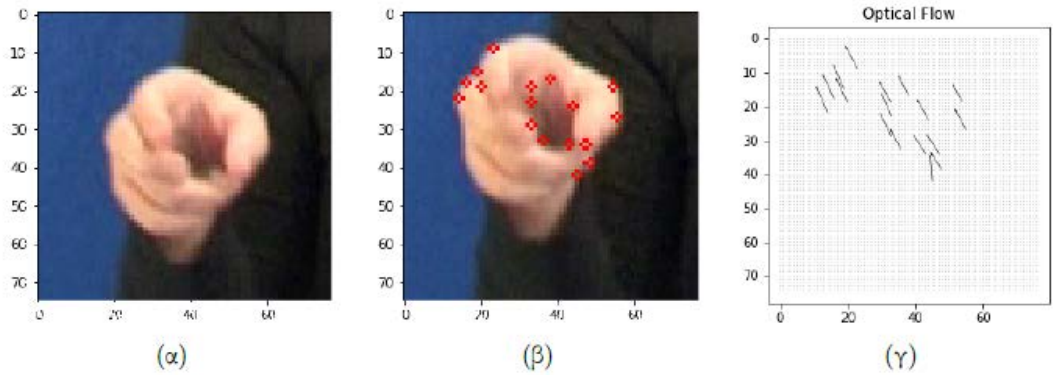

Figure 3: (a,b) Two consecutive frames of the mental language video sequence. The second frame shows and highlights the points of interest. (c) Diagram of the visual flow vectors at the points of interest, showing the upward and leftward movement of the hand.

1.2.3 Multiscale Optical Flow Calculation

Implement the multiscale version of the Lucas-Kanade algorithm. The algorithm will parse the original images into Gaussian pyramids and compute the optical flow from the smallest (coarse) to the largest (fine) scales, using the solution of the small scale as the initial condition for the large scale. The algorithm should be implemented as a stand-alone function, similar to before, but also taking as input the number of scales of the pyramid, and using the Lucas-Kanade single scale algorithm as a sub-routine. Useful instructions:

- For the transition from large to small scales when constructing the Gaussian pyramid, filter the image with a low-pass filter (e.g. Gaussian with a standard deviation of 3 pixels) before subsampling to reduce spectral aliasing of the image.

- When transferring d from small to large scales, do not forget to double it.

Then run your multiscale algorithm on the same sequence of images and comment on the differences you observe in the speed of convergence and the quality of the result compared to the single scale algorithm. The total displacement d for the bounding box is calculated as in 1.2.2.

Part 2: Locating Spatio-Temporal Points of Interest and Extracting Features in Human Action Videos

In Part 2 of the workshop we will deal with the extraction of spatio-temporal features in order to apply them to the problem of categorizing videos containing human actions. As we have already seen, local features have shown tremendous success in various computer vision recognition problems, such as object recognition. Local features describe the object to be observed by a set of local descriptors computed on neighborhoods of detected points of interest. Finally, the collection of local descriptors is integrated into a final global representation, such as the well-known 'bag of visual words', capable of representing their statistical understanding and proceeding to the next stages of recognition.

Representation using local features has also prevailed in the recognition of human actions, where a selection of data is made that on the one hand greatly reduces the dimension of the videos and on the other hand transforms them into a representation that makes them separable. In the exercise you will be given videos of 3 classes of actions (walking, running, boxing) from which you will extract spatio-temporal descriptors in order to categorize the actions they depict.

You can read the videos by calling the read video(name,nframes,0) function from the accompanying material, where name is the full name of the video and nframes is the number of frames (e.g. 200) you want to read. The video will be represented by a 3D matrix, whose 3rd dimension corresponds to time and is the sequence of frames, which are grayscale images.

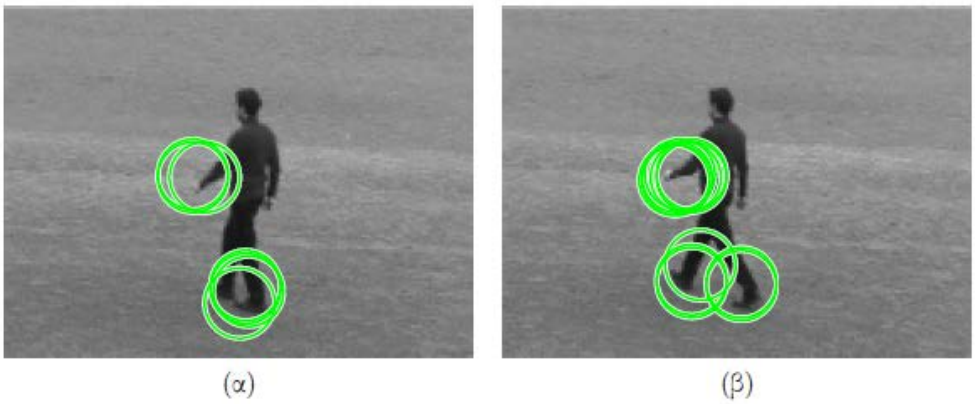

Figure 4: Example of spatio-temporal interest point detection (Harris Detector).

2.1 Spatio-temporal Points of Interest

Local feature detectors look for spatio-temporal points and scales of interest, which correspond to regions characterized by complex motion or abrupt changes in the appearance of the input video. This is achieved by maximizing a 'visual significance' function. Many detectors have been devised in recent years drawing sparse but robust points. In this laboratory exercise, we will deal with 2 different such detectors: 1) Harris detector and 2) Gabor detector.

2.1.1

Implement the Harris detector, which is a 3-dimensional extension of the Harris-Stephens angle detector, which you implemented in the 1st laboratory exercise. For each voxel in the video, compute the 3 × 3 matrix M (x, y, t) by adding to the 2D structural tensor and the time derivative:

$$M(x, y, t; \sigma, \tau) = g(x, y, t; s\sigma, s\tau) * (\nabla L(x, y, t; \sigma, \tau)(\nabla L(x, y, t; \sigma, \tau))^T)$$

or in tabular form:

$$M(x, y, t; \sigma, \tau) = g(x, y, t; s\sigma, s\tau) * \begin{pmatrix} L_x^2 & L_x L_y & L_x L_t \\ L_x L_y & L_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{pmatrix},$$

where g(x, y, t; sσ, sτ) is a 3D Gaussian normalization kernel and ∇(x, y, t; σ, τ) are the space-time derivatives for the spatial scale σ and the time scale τ. The derivatives (spatial and temporal) can be calculated by applying convolution with the kernel $[-1\ 0\ 1]^T$ (adjusted to the appropriate dimension).

The 3D angularity criterion also follows the same logic:

$$H(x, y, t) = det(M(x, y, t)) - k \cdot trace^3(M(x, y, t))$$

2.1.2

Implement the Gabor detector, which is based on temporally filtering the video with a pair of Gabor filters after it has been smoothed in spatial dimensions through a 2D Gaussian kernel g(x, y; σ) with standard deviation σ. The Gabor filters are defined as:

$$h_{ev}(t; \tau, \omega) = \cos(2\pi t\omega) \exp(-t^2/2\tau^2) \text{ και } h_{od}(t; \tau, \omega) = \sin(2\pi t\omega) \exp(-t^2/2\tau^2)$$

For the calculation of the Gabor impact response, assume a window size of [-2τ, 2τ] and normalize by the L1 norm.

The frequency ω of the Gabor filter is related to the time scale τ (deviation of its Gaussian component) by the relation: ω = 4/τ . The significance criterion is obtained by taking the quadratic energy of the output for the pair of Gabor filters:

$$H(x,y,t) = (I(x,y,t) * g * h_{ev})^2 + (I(x,y,t) * g * h_{od})^2$$

2.1.3

For each detector calculate the points of interest as the local maxima of the significance criterion. For simplicity, you can return the N points with the highest values of the significance criterion (e.g. the first 500-600). Illustrate for selected frames the significance criteria as well as the resulting points using the show detection function from the supplementary material. You can experiment with different spatial and temporal scales or even multiple scales. What do you observe about the type of points detected by the two methods?

2.2 Spatio-temporal Histographic Descriptors

The spatio-temporal descriptors to be used are based on the calculation of histograms of the directional derivative (HOG) and the histogram of oriented flow (HOF) around the calculated points of interest.

2.2.1

For each frame of the video, calculate the gradient and optical flow vector using the functions you have implemented in Part 1 of this lab.

2.2.2

Next, use the orientation histogram function from the supporting material in this section to calculate the 2 histographic descriptors. This function takes as input the vector field (directional derivatives or flow direction), the size of the grid and the number of bins and returns the histogram descriptor of the corresponding region. You are asked to extract the vector fields required for a 4 × σ (square) area around each point of interest (from the image corresponding to the frame where you detected points of interest). Pay attention to the boundaries of the image. To create the HOG/HOF descriptor, combine the two individual descriptors.

2.3 Construction of Bag of Visual Words and use of Support Vector Machines to classify actions

In this question we will categorise the videos of human actions into 3 categories/classes (each representing a different type of action) using the BoVW representations based on the HOG / HOF features extracted in the previous questions. The final result will be the percentage of successful classification of actions, using SVM (Support Vector Machine) classifier.

2.3.1

Separate the video set into a train set and a test set based on the file provided in the accompanying material.

2.3.2

Compute the global representation for each video using the bag of visual words (BoVW) technique described in Lab Exercise 1, using only the training videos. To compute the BoVW histograms, use the bag of words function from the accompanying material in this part.

2.3.3

The final stage consists in the final categorization of the images based on the BoVW representation. An SVM classifier suitably adapted for multiple classes is used for the categorization. The whole process is implemented by the svm train test function from the companion hardware, which accepts 2 numpy matrices of dimensions $N_{train} \times D$ and $N_{test} \times D$, where N_{train}, N_{test} are the number of training and test videos respectively and D is the dimension of each BoVW vector. The function returns the result of the recognition as well as the overall success rate.

2.3.4

Experiment with the different detector/descriptor combinations and observe the changes in classification. Indicate the best combination used and comment.

2.3.5

Experiment with different configurations of your data in train and test set and comment on the effect they have on your results.