Navigation

Dibris
- My home
  - Dibris
  - My profile
  - Current course
    - 65863-1314
      - Participants
      - Badges
      - General
      - Lab n. 1
      - Lab n. 2
      - Lab n. 3
      - Motion
      - SLIDES - motion
      - **Lab. 4 - Optical flow estimation (UPDATED 07/11)**
      - Lab session n. 4 - submission
      - Projective transformations

Lab. 4 - Optical flow estimation (UPDATED 07/11)

Computer Vision

EMARO - European Master on Advanced Robotics

Robotics Engineering *Master Degree*

Lab. Session n. 4

Implementation of the Lucas-Kanade algorithm

Write a Matlab function implementing the Lucas-Kanade algorithm to compute optical flow from a sequence of images.

You may use the following sequences, of rather different complexities, to test your implementation:

- sphere
- tree
- video-surveillance

Lucas-Kanade

For each position **x**=(x,y) of the image, we consider a NxN neighbourhood (e.g. N=5, 7) Q which produces a system of NxN equations in the two unknowns (u_x, v_y)=**u** (which are the components of the optical flow computed in position **x**). If we denote with $\mathbf{x}_i$ the points in Q, then

$$\nabla I(\mathbf{x}_i, t)^\top \mathbf{u} = -I_t(\mathbf{x}_i, t)$$

The left side of the equations is based on the computation of the image gradients (see the lab. session on edge detection), while on the right of the equal there is the temporal derivative, which may be approximated as

Single view geometry and camera calibration
Stereopsis
Other methods for 3D reconstruction
My courses
Courses

Administration
Course administration
My profile settings

$$I_t(\mathbf{x}_i, t) = I(\mathbf{x}_i, t) - I(\mathbf{x}_i, t-1)$$

Now, the NxN equations form a system of type A**u**=**b,** that can be solved as

$$\mathbf{u} = (A^T A)^{-1} A^T \mathbf{b}$$

What to produce

- A function that, given two images, computes the optical flow in each point:

```
function [u, v] = LucasKanade(im1, im2, windowSize);

% INPUT: im1, im2 adjacent image frames (istants t and t+1)

%        windowSize size of the neighbourhood

% OUTPUT: u, v matrices containing optical flow components
%-------------------------------------------------------------------------
```

  To evaluate the confidence of your estimated in each point, it may be of help seeing how well conditioned is the system.

- A main that loads the sequence image by image, calls the LucasKanade function on pairs of images adjacent in time, and visualizes the result. For the latter functionality you can use the matlab function quiver: assuming that U and V are the matrices of the two optical flow components,
  - first change the reference system (from origin in the top-left to bottom-left): `Uf=flipud(U), Vf=flipud(V)`
  - then try to visualize the full matrices: `quiver(Uf,Vf)`. It might be too dense...
  - thus try with a subsampling `quiver(Uf(1:10:size(Uf,1), 1:10:size(Uf,2)), Vf(1:10:size(Vf,1), 1:10:size(Vf,2)))`

Note: The Lucas-Kanade algorithm fails in presence of large displacements between consecutive frames. In this case, a hierarchical approach allows to improve the estimates.