MIT CSAIL

6.869 Advances in Computer Vision

Spring 2011

Problem Set 9: Motion Magnification

Posted: Wednesday, April 6, 2011 **Due:** Wednesday, April 13, 2011

You should submit a hard copy of your work in class, and upload your code (and all files needed to run it, images, etc) to stellar.
Your report should include images and plots showing your results, as well as pieces of your code that you find relevant.

As we've seen in class, motion analysis has numerous useful applications in computer vision. In this problem set, you will experiment with motion estimation, and use it to magnify motions that might be barely visible to the naked eye, following the work of our guest lecturer [1].

Problem 9.1 *Lucas-Kanade*

You will start by implementing the Lucas-Kanade (LK) optical flow algorithm for estimating dense motion between a pair of images. Since some of the motions might be too large for the Taylor approximation of the LK step, we will apply the algorithm in a coarse-to-fine manner. On the class website you will find skeleton code that we recommend you use for your implementation.

As discussed in class (refer to the lecture slides for further details), the algorithm operates as follows. Given two images I_1,I_2 and number of levels k,

Algorithm 1 Coarse-to-Fine-LK(I_1, I_2,k)

```
1: Build $k$-level Gaussian pyramids $G_1, G_2$ for $I_1, I_2$
2: Find the optical flow field $(u_k, v_k)$ from $G_2^k$ to $G_1^k$ at the coarsest pyramid level $k$ using the Lucas-Kanade algorithm
3: Upsample the flow field for level $k-1$, and transform $G_2^{k-1}$ towards $G_1^{k-1}$ using $(u_{k-1}, v_{k-1})$
4: Update the optical flow estimation $(u_{k-1}, v_{k-1})$ at level $k-1$
5: Repeat 3–4 for levels $k-2, k-3, \dots, 1$
```

(a) Implement the function

```
[u,v,warpI2] = lucaskanade(I1,I2,u0,v0,winsize,medfiltSize,nIterations)
```

that receives as input two images `I1` and `I2` and initial flow estimates `(u0,v0)`, and computes the optical flow field `(u,v)` from image `I2` to `I1` using the Lucas-Kanade algorithm. The function receives the following parameters: `winsize` - half the patch size, `medfiltSize` - the size of the window for the spatial median filter, and `nIterations` - the number of flow refinement iterations. `warpI2` is the image I2 warped according to `(u,v)`.

(b) Implement the function

```
[u,v,warpI2] = coarse2fine_lk(I1,I2,nlevels)
```

that receives as input two images `I1` and `I2`, and computes the optical flow `(u,v)` from `I2` to `I1` using the coarse-to-fine scheme of Lucas-Kanade (Algorithm 1) using your function from part (a).

(c) Run the algorithm on the `car` image pair supplied with the code. You can use the script `demoMotion`, which calls the `coarse2fine_lk` function from part (b) and displays the computed flow field and corresponding warp. The parameters in that script are set to values which we found to produce good results. Add to your report the resulting optical flow image and the warped `car2` image.

Hints: Use the supplied `warpFL` function to warp an image according to a given flow field. You can use the MATLAB functions `imfilter` and `fspecial` for filtering the images, and `imresize` for rescaling.

Problem 9.1 *Motion magnification*

In this problem you will implement a simple version of the motion magnification algorithm [1] discussed in class. On the class website you will find the file `bookshelf.zip`, which contains a short image sequence.

The algorithm is as follows, where I is the input video sequence, $\alpha > 1$ is the motion gain, and τ thresholds large velocities.

Algorithm 2 Simple-Motion-Magnification(I, α, τ)

```
1: Estimate the motion $(u_t, v_t)$ between frame $I_1$ and each frame $I_t$ in $I$.
2: Modify the flow fields according to $(u'_t, v'_t)_p = \begin{cases} \alpha(u_t, v_t)_p & ||(u_t, v_t)_p|| \le \tau \\ (u_t, v_t)_p & \text{otherwise} \end{cases}$
3: Generate the motion-magnified sequence by warping $I_1$ according to $(u'_t, v'_t)$ for each $t$.
```

(a) Implement the simple motion magnification algorithm, and run it on the `bookshelf` sequence using $\alpha = 10$ and $\tau = 2$. Produce a video showing the original frames and resulting frames side by side, and submit your result as uncompressed avi file. You can use the MATLAB functions `movie2avi`, `avifile` or `VideoWriter` for this purpose.

(b) **[Optional]** Capture your own short video with subtle motions that are hard to notice, and run the algorithm on it to magnify the motions.

References

[1] C. Liu, A. Torralba, W.T. Freeman, F. Durand, and E.H. Adelson. Motion magnification. pages 519–526. ACM, 2005. `http://people.csail.mit.edu/~celiu/motionmag/motionmag.pdf`.