shipout/lastpageshipout/foregroundshipout/background
shipout/lastpage

Global Motion Estimation

An indirect, multiscale and robust approach

Samuele Conti Filippo Daniotti

Department of Information Engineering
and Computer Science

April 20, 2022

Table of Contents

Introduction

In this work we are going to present:

- a broad introduction to the problem of global motion estimation
- an overview of the theoretical foundations of our solution
- the framework we developed from the aforementioed literature
- an analysis of the performances of our solution
- a brief discussion of the limitations and possible improvements

Table of Contents

Global motion estimation

The motion in videos can be analized as the displacement of pixel per pair of frames

Is usually the combination of two different motions:

- the actual motion of objects in the scene
- the egomotion of the camera

Figure: Local motion vs. egomotion

The problem

Global motion estimation aims to extract the camera global motion patterns in a video

Motion models

Motion models describe camera motion patterns through equations

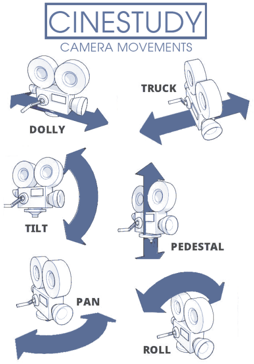


Figure: Different models describe different types of camera motion

The displacement of a pixel across frames as a function of some parameters

$$p' = \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} T_X \\ T_Y \\ T_Z \end{bmatrix} = Rp + T \quad (1)$$

E.g. for the **affine model** (our choice) parameter vecotr is $a = [a_0, a_1, a_2, b_0, b_1, b_2]$

Estimating parameters

There are two main approaches, both of which aim to minimize a prediction error

- **direct methods** work in the pixel domain
- **indirect methods** work with motion fields

For our solution we opted for **indirect methods**

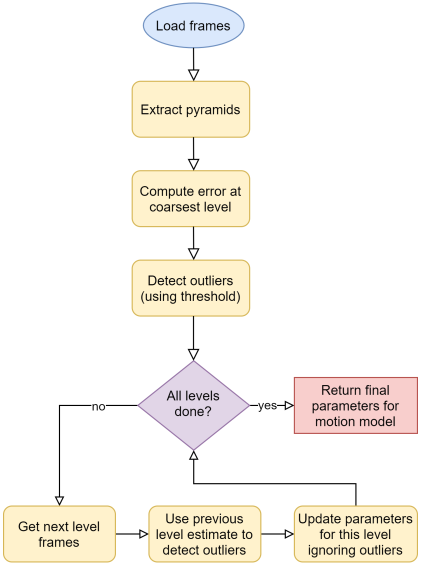
Indirect paramter update

The parameter vector a is updated by choosing the parameters that minimize the dissimilarity of the $gt(x)$ and the prediction $MM(x, a)$ for a point x

$$a = \arg\min_{a} \sum_{p} E(gt(p) - MM(p, a)) \tag{2}$$

Table of Contents

Overview



Load frames
Extract pyramids
Compute error at coarsest level
Detect outliers (using threshold)
All levels done?
yes
Return final parameters for motion model
no
Get next level frames
Use previous level estimate to detect outliers
Update parameters for this level ignoring outliers

Figure: Flowchart of our framework

Implementation choices:

- Affine model
- Indirect method
- Block-based motion estimation
- Multiscale
- Robust

Block-based motion estimation

BBME algorithms perform motion estimation and return the motion field to use as ground truth for indirect parameter estimation

We implemented 4 well-known BBME algorithms:

- exhaustive search
- three-step search
- 2D log search
- diamond search

Exhaustive search

(a) Target frame

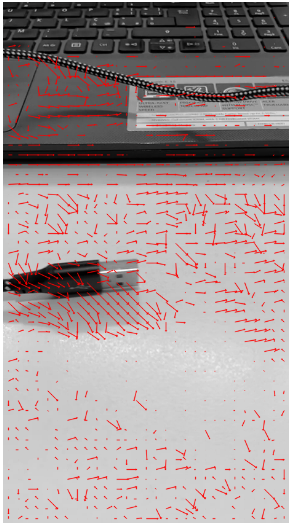
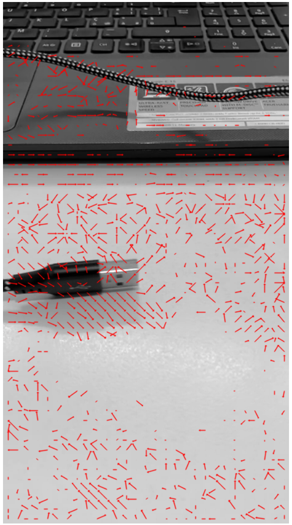

(b) Needle diagram

Figure: Motion field result of our EBBME implementation

Three-step search

(a) Target frame



(b) Needle diagram

Figure: Motion field result of our TSS implementation

2D Log search

(a) Target frame

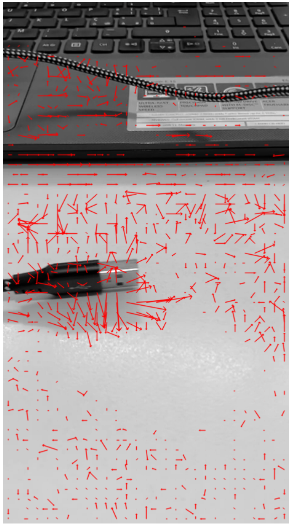

(b) Needle diagram

Figure: Motion field result of our TDLS implementation

Diamond search

BBME algorithm presented in Zhu and Ma, 2000.

Better performances thant classic algorithms (TSS) and comparable to latest ones (NTSS), yet more efficient in computation.

It uses two different diamond-shaped search patterns:

- large diamond search pattern (LDSP)
- small diamond search pattern (SDSP)

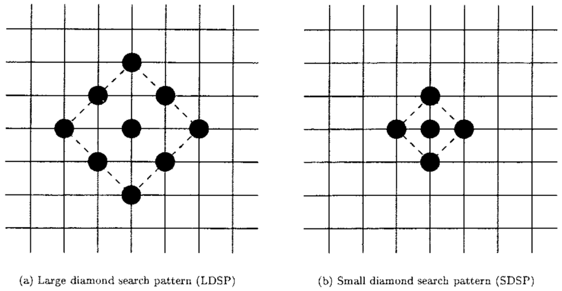

(a) Large diamond search pattern (LDSP)
(b) Small diamond search pattern (SDSP)

Figure: The shape of LDSP and SDSP

DS algorithm

Diamond search procedure

For each block:

1. place LDSP in the center of the block and compute DFD for all of the 9 displacements;
   - if the minimum DFD is in the center position, then go to 3
   - else, go to 2
2. reposition LDSP in the minimum DFD of the previous operation and recompute DFD;
   - if the minimum DFD is in the center position, then go to 3
   - else, repeat
3. place SDSP, compute DFD and return coordinates of the minum DFD block as motion vector

DS algorithm visualized

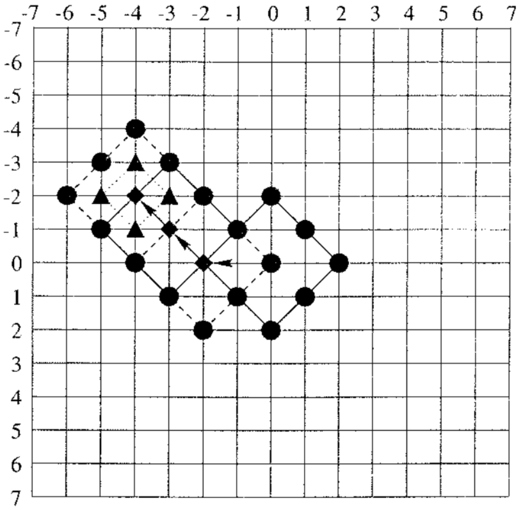
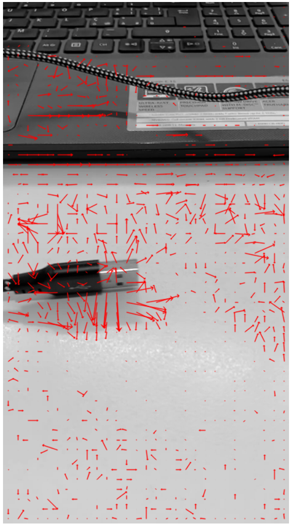


Figure: Example of a full execution cycle of diamond search

DS results

(a) Target frame



(b) Needle diagram

Figure: Motion field result of our DS implementation

Affine model prediction error

The affine model computes the displacement of a given pixel wrt parameters a as

$$d(p, a) = A[p]\; a \tag{3}$$

where $A(x)$ is an intermediate matrix

$$A[p] = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix}$$

Fitting error

The fitting error computation gets rewritten as

$$E = \sum_{p} |d(p, a) - d(p)|^P \tag{4}$$

Affine model parameter estimation

We are interested in the parameters that minimize the fitting error $\rightarrow \nabla_a E = 0$

Parameter update

Assuming $p = 2$, we get

$$a = \left(\sum_p A[p]^T A[p] \right)^{-1} \left(\sum_p A[p]^T d(x) \right) \tag{5}$$

Refinement

- each point p can be weighted
- parameter vector a can be split: $a_x = [a_1, a_2, a_3]$ and $a_y = b_1, b_2, b_3$
  - reduce the overall complexity

$$a_x = \left(\sum_p w(p) A_x(p)^T A_x(p) \right)^{-1} \left(\sum_p w(p) A_x(p)^T d_x(x) \right) \tag{6}$$

Table of Contents

Compensation

Table of Contents

Thanks!