3.3.3 Visual Energy

Daniel Chávez Heras

This first technique is concerned the computational representation of perceived motion in digital video, this is, with how viewers generally perceive and understand movement in audiovisual artefacts, and the extent to which this understanding can be analysed computationally.

Movement is a fraught category in moving image studies, it has a rich history that even predates the cinematographer, and whose most complex account is perhaps Gilles Deleuze's *Cinéma 1. L'Image-Mouvement* (1986 [1983]) , a tome solely devoted to the philosophical investigation of movement in film. Such an account is beyond the scope of the present work, although I expect part of this complexity will re-emerge towards the end of the chapter, as indeed some of Deleuze's ideas.

In terms of *Made by Machine*, and in order to approach it computationally, motion was defined as the aggregated data difference between consecutive frames in a given video segment. For any sequence of frames, therefore, as long as a frame is identical to the next it is assumed no motion occurs, and conversely, whenever consecutive frames are not identical, the measurable amount of change between them is then interpreted as a signal of motion.

From this initial definition, it is then possible to estimate motion computationally by calculating differences between corresponding pixels in consecutive still frames in a video file, given that each consecutive still frame can be represented as a numeric matrix of the same shape in *x* and *y* corresponding to the height and width of the video the frame. In image processing, this basic operation is known as *image subtraction* (Figure 1).

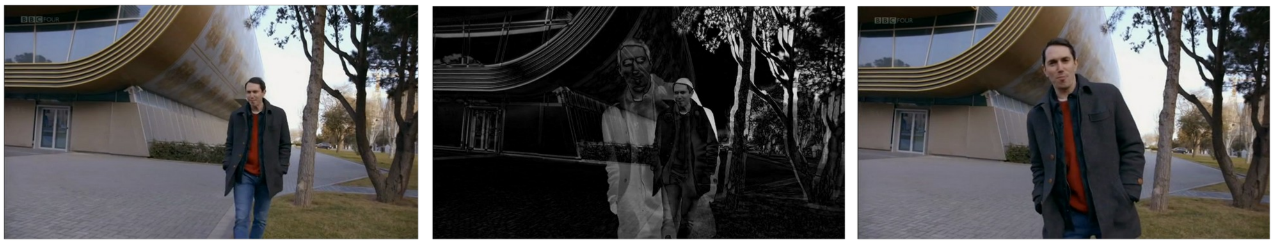


Figure 1: Calculation of pixel difference between two frames from a clip

Video encoders take advantage of this type of calculation to compress video and reduce file size. Consider for example a one-hour television programme: in high definition and at 25fps, the video file would be comprised of about 90,000 frames (See Appendix A), each of which contains little over two million pixels (See Appendix B). At the highest picture quality[1], the TV programme could be over 190GB in file size.

This file size is prohibitively large for most transmission purposes and too taxing even for local playback and storage. The solution to this is video compression, which in effect means to store fewer full “key frames” and to approximate the frames in between by estimating an interpolation, thereby greatly reducing the required data stream. This principle lies at the core of most video compression methods, and is, in broad terms, the logic of MPEG[2], the most common international standard for digital video encoding.[3]

In MPEG, the interpolation between key frames is achieved through the calculation of **motion vectors**, which are a one dimensional representation of frame to frame difference, obtained by averaging corresponding 16x16 pixel regions of subsequent frames known as **macro blocks**. For each of these macro blocks, if the weighted difference in pixel values is computed below a given threshold, the macro block as a whole is deemed to have remained unchanged from the previous frame and is therefore not encoded again. In this way, only macro blocks that compute a significant difference from one frame to the next need to be assigned new pixel values, while the more “static” macro blocks simply retain their previous pixel values without the storage overhead.[4] Different encoders and video file formats organise frame sequencing in different ways so as to strike the

1 This is calculated using the highest quality JPEG render, which without metadata requires about 8.25 bits per colour pixel (see: Minguillón Alfonso and Pujol Capdevila, 2001).

2 Acronym of *Moving Picture Experts Group*, “a working group of ISO/IEC with the mission to develop standards for coded representation of digital audio, video, 3D Graphics and other data.” See: https://mpeg.chiariglione.org/

3 According to their own metrics, MPEG is able to compress video streams up to a ratio of 200:1

4 If one imagines *f1* to be the initial frame of a sequence, MPEG encodes this frame in its entirety as an independent JPEG still frame, or *I*-frame, and predicts *f4* –a *P* frame– through these macro block motion vectors. In this way, *f2* and *f3* –known as *B* frames– are only encoded as the interpolation between an *I* frame and a *P* frame. In MPEG encoding, *I* frames are the least compressed, while *B* frames are the most compressed.

desired balance between data stream throughput, perceptual quality of video, and random access requirements.[5]

The concept of motion vector in this encoding process implies a correlation between stream size and motion estimation in digital video: greater change from to frame will tend to yield larger file sizes because this variance will have to be encoded, while very low variance from frame to frame will conversely result in more efficient compression and smaller file sizes because of fewer changing macro blocks.

From this observation of how MPEG works, the intuition was to exploit video compression as a measure of motion in our dataset of television clips. To do this, the difference in size between frames δ was calculated using the following formula[6], where *n* is the size of a frame (in bytes) and *t* is a time step[7]:

$$\delta = |n_t - n_{t-1}|$$

By calculating δ for each pair of consecutive frames in a clip, a consistent signal representing motion through time was obtained, and a global measure μ of motion per clip was assigned using this formula:

$$\mu = \frac{\sum \delta}{\sum frames}$$

5 *B* frames cannot be decoded without first decoding both *I* and *P* frames at either end, and therefore the more *B* frames that there are in a sequence the more limited random access is. Think for example of online video streaming services, which often have a progression bar which allows users to skip to a specific part of the video. The more *B* frames the stream is encoded with the less precise this skipping will be, as video needs to be rendered from the closest *I* frame.

6 In this case values are not normalised so as to preserve the order of magnitude for comparison, however, note that we do use the Manhattan (L_1) norm to avoid “negative motion”.

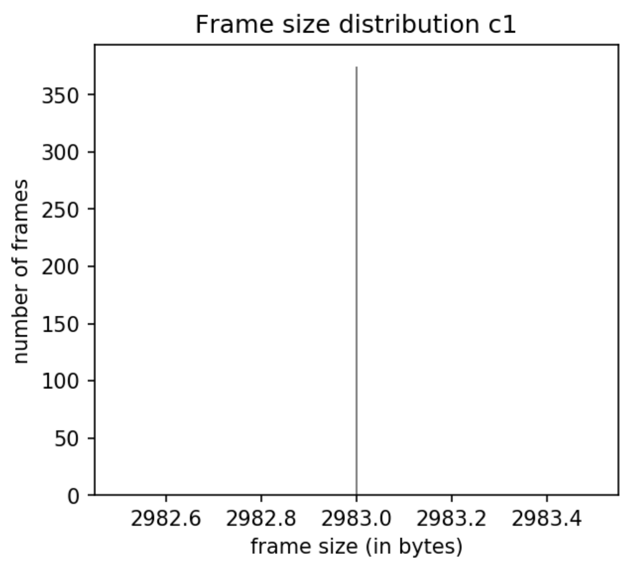
7 We use the most granular account of time steps which in this case is by the frame. But for larger datasets or other more computationally intensive processing it might make more sense to use a different time sample.

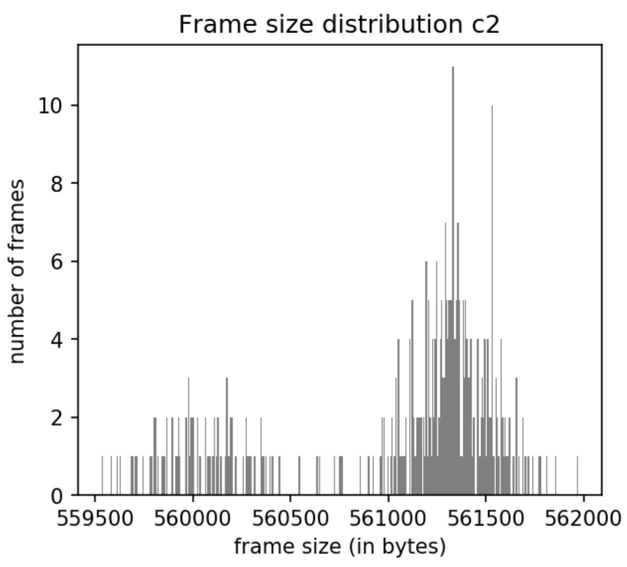
All the clips in the dataset were then annotated with their respective μ index so that they could be accessed, handled and processed according to this measure of motion. Let us see an example of how this feature can be used for inference.

Consider the following clips:

> *c1*: A control clip where every frame is blank – i.e. a single solid block of light grey.†
> *c2*: Random noise signal – similar to analogue TV static. †
> *c3*: An aerial shot of the city scape of Baku – fragment of a BBC Four documentary.‡
> *c4*: A piece to camera of a presenter – a different fragment of the same documentary.‡

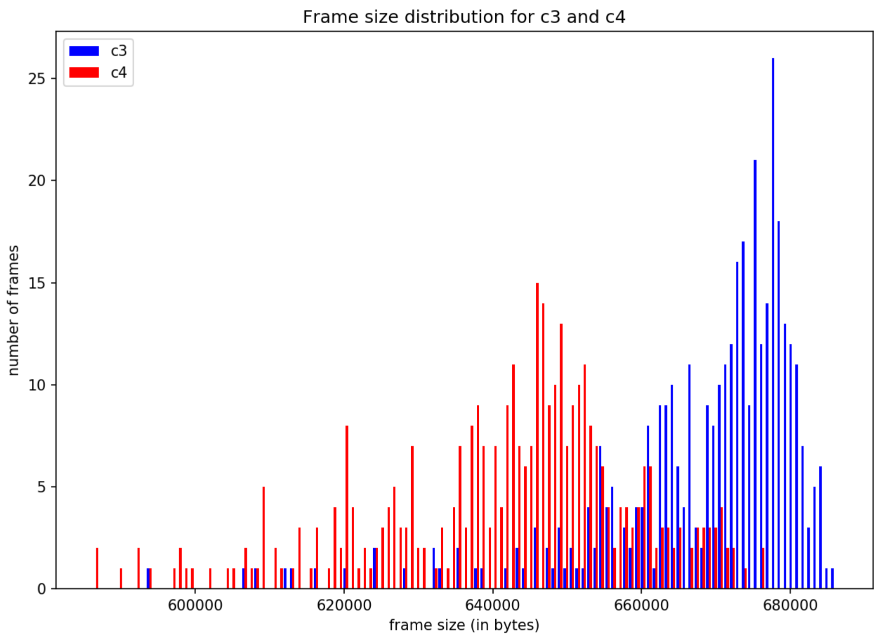
Each clip lasts exactly 15 seconds, which at 25fps means 375 frames per clip. Each frame is 704x576 pixels in shape. Using the method described above, the δ values values for all frames are calculated and their distribution is rendered as a histogram. As expected, *c1* shows a *degenerate* distribution, i.e. where every frame is identical at 2983 bytes and hence their δ=0. Random noise, meanwhile, presents a constant signal of relatively stable variance of δ:



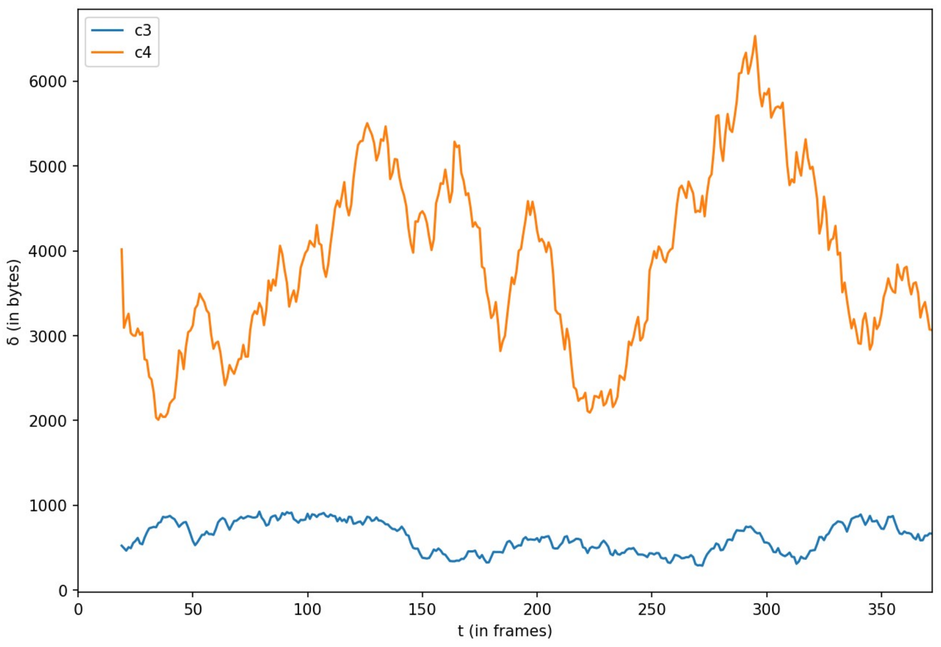


† This was created synthetically using *ffmpeg*, see: https://trac.ffmpeg.org/wiki/Slideshow
‡ *An Art Lovers' Guide: Baku* (https://www.bbc.co.uk/programmes/b0b0g5cj)

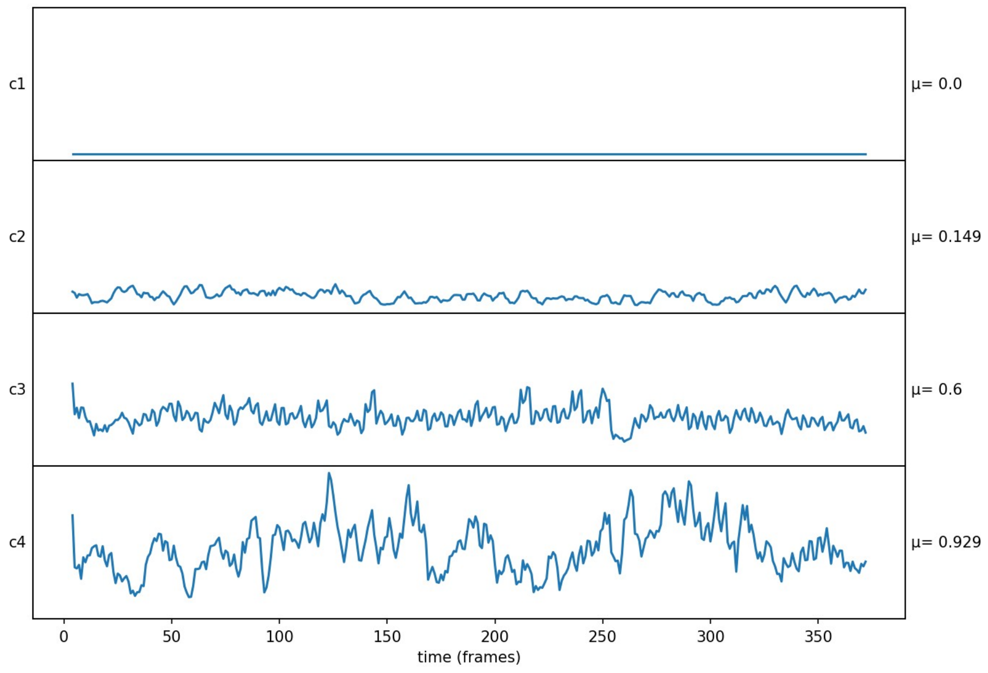
The distributions for *c3* and *c4* on the other hand show a much richer δ gamut and overlapping but differentiable shapes:

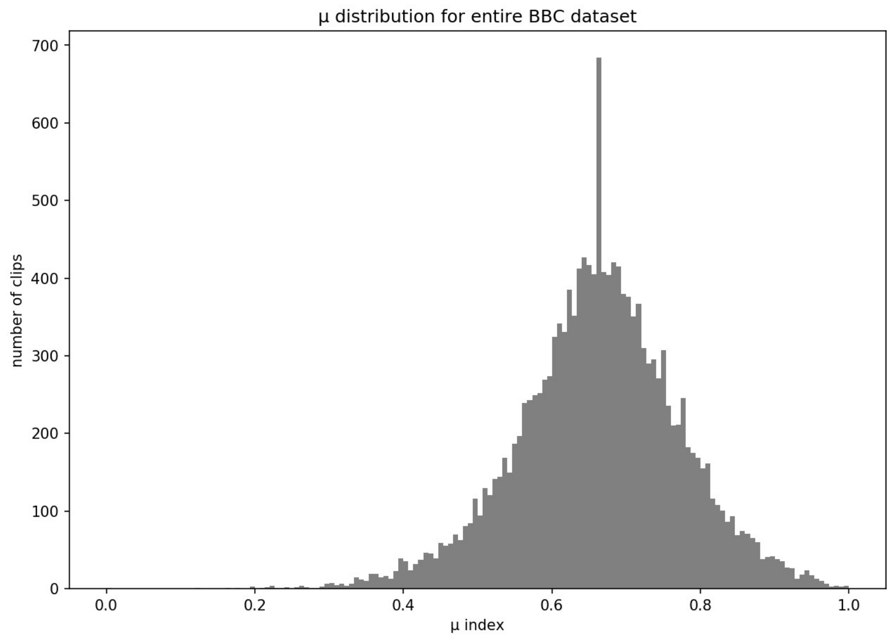


A comparison of δ visualised as a time series allows inferences about the clips in terms of their motion: while the documentary clips have similarly higher μ values, *c3* is relatively stable across time, which appears to correspond to the smooth camera motion of the aerial shot, slowly gliding by the skyline of Baku. *c4* on the other hand is less stable and can be said to represent better the more serendipitous motion of the presenter in two distinguishable movements: as he moves towards the static camera and when he stops in front of it to gesticulate:



Finally, at the level of clips, each one can be described by its μ index, which then allows their classification and retrieval on the basis of motion:





The BBC dataset was automatically annotated using this technique. The histogram above shows the distribution of all 15,572 clips in the dataset. Note how there is a significant number of clips at μ = 0.662735, quite possibly on account of duration-keyframe ratio convergence. This was not seen as an issue at the time, as the main goal was to use these annotation to produce a "watchable sequence", i.e. one that showed how motion was isolated and used as the driving editing logic. So in order to make this evident to a television audience and to avoid looping through convergent clips, it was decided to concatenate clips by enveloping them in a predefined "motion shape". The sinusoid curve selected for this was somewhat arbitrary, its only purpose being to provide a smooth *crescendo-diminuendo* from hyperkinetic to static clips —a transition that the audience could follow on screen along with the clips themselves. Note how contrary to common scientific usage, for the television interface the curve does not describe the clips, the clips are instead fitted to the curve according to their μ indeces (Figure 2):

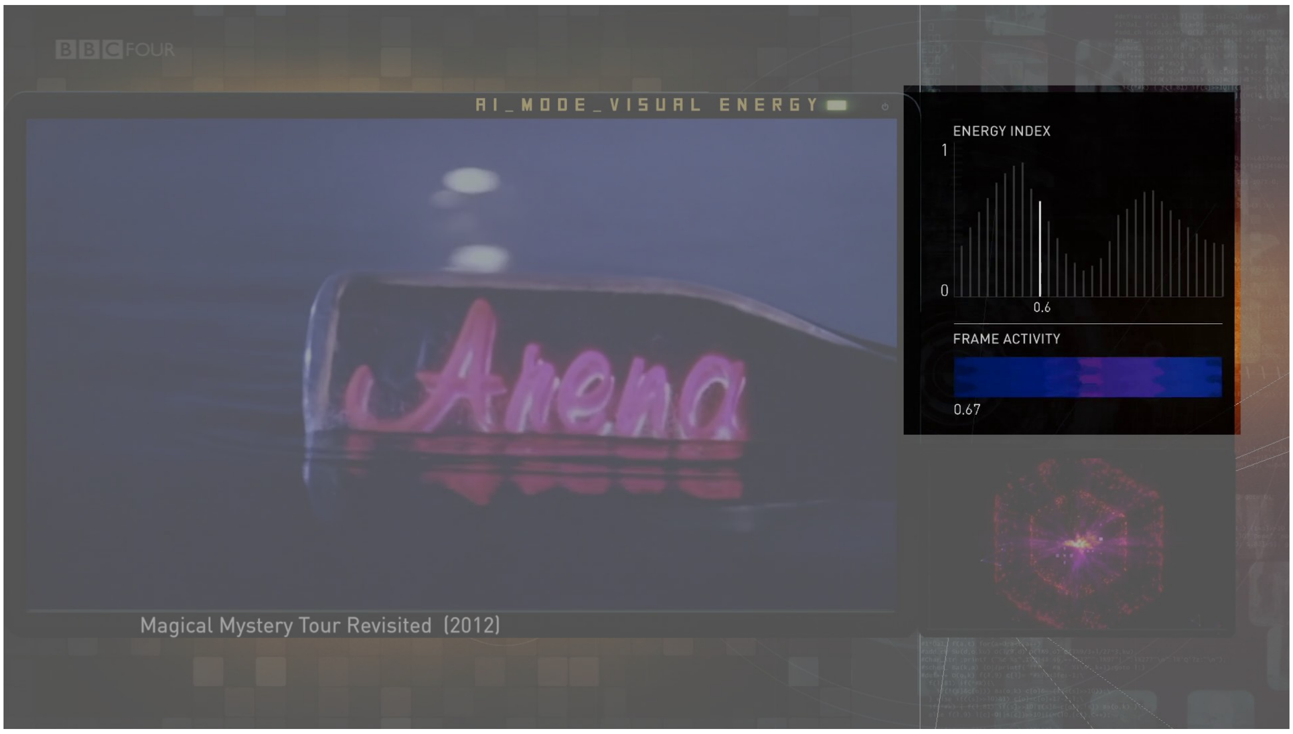


Figure 2: Motion envelope interface in Made by Machine

It is also important to note that there are several different more established ways to calculate motion. The estimation of perceived motion from video sequences is a well-known problem in computer vision, where it is often referred to as **optical flow**. A few sophisticated methods have been developed since at least the early 80s to estimate optical flow, most notably variational approaches such as the Horn-Shunck (Horn and Schunck, 1981; Horn, 1986) or the Lucas-Kanade (Lucas and Kanade, 1981; Lucas, 1985) methods. In recent years, and with renewed incentives from the autonomous vehicles industry, novel deep learning approaches have also been developed (Dosovitskiy et al., 2015; Revaud et al., 2015; Weinzaepfel et al., 2013), although to my knowledge the differential equations created during the 80s along with their multiple variants and improvements are still very much the standard for calculating optical flow today.

There were plans for calculating optical flow in MbM using these methods, but in the end the team settled for the approach described earlier because it was much leaner and therefore far more likely to be implemented in time given the constraints of the project. Also, for the purposes of the television programme it produced similar results to the more complex calculations of optical flow without the additional computational requirements.[8]

References

Deleuze, G. (1986) *The movement-image*. University of Minnesota Minneapolis.

Dosovitskiy, A. et al. (2015) '*FlowNet: Learning Optical Flow With Convolutional Networks*', in 2015 pp. 2758–2766. [online]. Available from: http://openaccess.thecvf.com/content_iccv_2015/html/Dosovitskiy_FlowNet_Learning_Optical_ICCV_2015_paper.html (Accessed 4 September 2019).

Horn, B. K. P. (1986) *Robot vision*. 2003rd edition. Cambridge, Mass.: MIT Press.

Horn, B. K. & Schunck, B. G. (1981) Determining optical flow. *Artificial intelligence*. 17 (1–3), 185–203.

Lucas, B. (1985) *GENERALIZED IMAGE MATCHING BY THE METHOD OF DIFFERENCES*.

Lucas, B. D. & Kanade, T. (1981) *An iterative image registration technique with an application to stereo vision*.

Minguillón Alfonso, J. & Pujol Capdevila, J. (2001) *JPEG standard uniform quantization error modeling with applications to sequential and progressive operation modes*. [Online] [online]. Available from: http://openaccess.uoc.edu/webapps/o2/handle/10609/6263 (Accessed 23 July 2019).

Revaud, J. et al. (2015) EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. *arXiv:1501.02565 [cs]*. [online]. Available from: http://arxiv.org/abs/1501.02565 (Accessed 4 September 2019).

Weinzaepfel, P. et al. (2013) 'DeepFlow: Large Displacement Optical Flow with Deep Matching', in *2013 IEEE International Conference on Computer Vision*. [Online]. December 2013 pp. 1385–1392.

[8] I did in fact implement a version of the Lucas-Kanade method to estimate optical flow in the BBC dataset. But in the end this implementation would have had to go through an internal process of validation and integration to R&D's COMMA platform in order to be used on the dataset. There was no time for this, nor for me to polish my very hacky optical flow script. But even if this had not been the case, note that many implementations of the Lucas-Kanade method are as sensitive to noise as our very basic method.