4
Proposed methods for keypoint tracking

This chapter introduces two algorithms devised for tracking keypoints in a video sequence. The following sections describe in details each one individually, but the general guidelines given in this preamble will help understanding their broad functioning, for they may actually be perceived as two variants of the same method, rather than different approaches to tackle the same problem.

In fact, the choice of which to adopt should be problem-driven, i.e., regarding the particular characteristics of a sequence, whose terms will be discussed in the forthcoming chapter. Both variants are a potpourri of the algorithms for point matching presented in chapter 3 - or a variation of some of them - and the contents of each mix are what makes one brand more suitable for dealing with a specific kind of movie.

Generally, in a video sequence, two subsequent frames are converted into the image pair for tracking each keypoint position. The processing time of the matching procedure determines the frame sampling rate; that is, the faster the procedure, the closer (in time scale) the two frames.

Consecutive frames that are closer in time tend to be more similar to one another, making keypoints easier to trace. Thus, in such cases, the robustness demanded of the matching technique may drop, to a certain degree, without corrupting the tracking progress. That said, one is left with the circular trade-off dilemma depicted below, in Figure 4.1.

![Figure 4.1: Scheme that drove the devising of the object tracking method.](image)

Figure 4.1: Scheme that drove the devising of the object tracking method.
Leaving the relation between robustness, speed and complexity of the matching procedures for a later discussion, there is still the matter of initialising - and maintaining - a keypoint collection, so that the tracking process may actually be carried on, to be taken under consideration. Typically, matching procedures that are less robust are also more likely to lose track of a greater number of keypoints during the process. Thus, such collection must be updated to either replace lost keypoints, refresh their coordinates, or even add new ones that are more suitable to the current scene.

Moreover, environmental problems, such as occlusion and change in illumination, make the selection of the initial keypoints even less trivial and in fact suggest such collection should not be statical. Which brings on the matter of how to assess the suitability of a keypoint to the current scene, as addressed in (26) and (27), for instance.

The method proposed herein makes needless to implement any heuristics in such assessment. The adequacy of the collection is automatically assured by the fact that its entries are the keypoints found as close in time as possible to the sequence frame being processed at a given instant, and thus they have a higher chance of being consistent with the upcoming scene. This is achieved by parallel processing, i.e., separate threads for the tracking and updating processes.

The block diagram depicted by Figure 4.2 illustrates the overall process throughout time.

Prior to running the tracking algorithm, an object model must be built off-line. This model consists of a keypoint list - which may be analyst-selected, to assure stability - containing the features’ x-y coordinates, associated to a particular descriptor, and their correspondent world coordinates, for the object pose detection calculations, which can be performed, for instance, by a third thread. Since the SIFT algorithm was adopted at the early tracking stage in both variants, the descriptor used is exactly as described in (19). The collection must contain keypoints corresponding to all views of the object.

The updating strategy is as follows. Using the past matching sequence frames, a thread should be held responsible for carrying on the collection update. New features, in these last frames, are matched to the original model image(s) to assemble a new starting list, i.e., the keypoint collection to be processed by the next tracking thread. Finally, a mask matrix is associated to the starting list. These keypoints are the ones most likely to be consistent with the scenes from this moment on - since they were spotted at frames corresponding to immediately past time instants - therefore making them the best candidates for tracking. They must be matched against the original model,
As soon as an update is ready (and the current tracking thread is idle),
the keypoint coordinates corresponding to the template image, or rather, the
template coordinates, are refreshed - or replaced by others - and the matching
process continues. The old tracking thread dies as another update thread starts.
All these new keypoints are flagged as “fit” in the mask matrix. During the
matching process, whenever the algorithm fails to keep track of a feature,
its template coordinates are tagged as “unfit” and cease being part of the
tracking process at the subsequent frames, remaining latent until the current
thread dies, but remembering that they may still be members of the next list
delivered by the following update.

The keypoint updating process is kept indefinitely ongoing. The only
requirement is that there must be a new set of keypoints ready, whenever the
number of fit pairs falls beneath a certain threshold, which may be around the
minimum points required for the 3-D reconstruction (pose detection). If, by
any chance, the list is entirely lost before the update process has delivered the
next one, the object will remain untraceable for that period.

Having introduced the kernel of the object tracking process, let us present
in details the two variants devised for running in the matching thread.
4.1

Variant 1

As previously stated, the matching algorithms implemented for the actual object tracking that are fast enough to keep a high frame sampling rate, are also less robust, due to their simpler nature. This lack of complexity demands keypoints that are themselves both inherently easy to match and consistent with the upcoming scene, which is achieved by the updating procedure described in the previous section. Having addressed the point stability matter, let us discuss how the matching techniques presented in chapter 3 can be used to take full advantage of the keypoint collection quality.

At the very beginning of the tracking process, there is hardly any knowledge about the camera position, in relation to the pose of object model (or the keypoint collection). Hence, the point correspondence in the template-matching frame pair at this stage is rather more difficult, in relation to the ones hereafter. Therefore, a more robust procedure should be adopted.

From hereon, as to assure keypoint stability and pairing consistency, an algorithm that can not only provide a match, but also eliminate false initial matches is desirable. As the video sequence moves further in time and the camera movements have stabilised, as well as the false matches have been eliminated, the tracking process will not require a great filtering ability nor a wide image transformation coverage. Consequently, the matching algorithm as the current tracking thread advances in time can be even simpler.

Based upon the discussion above, the mix proposed for this variant is as follows. Figure 4.3 shows the object tracking scheme.

- apply the SIFT algorithm and match the starting frame against the model
- whenever the former step is completed, grab the incoming frame and use LSM to adjust the coordinates of all keypoints, using their present values, i.e., the SIFT output, as initial guesses and with the last processed frame - the SIFT-frame - as the template
- consecutively apply NCC from now on to refresh the still fit keypoints; the initial guesses and template are analogous to the LSM-step

This procedure restarts every time a collection update is released. The update processes are in fact analogous to the first step and replace it as the scheme goes further in the time line. Notice that the number of frames lost, i.e., left unprocessed during each match calculation drop significantly from stage to stage, since the procedures themselves get faster and faster and, additionally,
more keypoints become unfit (and no longer adjusted), although the latter aspect is a direct consequence of adopting this procedure rather than properly an advantage.

Chapter 5 provides quantitative information to buttress the strategy devised for this variant.

Figure 4.3: Block diagram of the object tracking method: variant 1, also called the SLN variant.

In the remaining of this document, variant 1 will be designated by the initials of the algorithms that compound the mix (namely, SIFT, LSM and NCC), that is, by the SLN abbreviation.

4.2 Variant 2

Both the LSM and the NCC algorithms have matrices of neighbouring image pixel values as descriptors. Hence, not only they behave rather poorly in cases where the geometric difference between the pairs of frames is too significant - like in high movement sequences - but they also may become quite unstable when processing severely compressed video sequences, namely those where the quantisation blocks are noticeable and a pixel’s neighbours - and even location - are far from corresponding to the ones in the original uncompressed frames.

This second variant was devised to deal with such sequences and its functioning is very similar to its peer’s. Here, the LSM-NCC routine is replaced by
a SIFT-based matching technique. Figure 4.4 depicts the alternative matching strategy.

Running the SIFT algorithm as implemented in the first step throughout the whole sequence would incur a massive loss of image frames, due to its processing time. Again, if the subsequent frames used as template-matching pairs are kept close enough in time to one another, it is fair to admit that a keypoint location will not wander off, in terms of absolute image coordinates, regardless the significancy of the geometric transformation it went through. Therefore, processing only a patch of the matching frame, around the coordinates of each template keypoint, should be enough to determine the new location.

Regarding this variant, the matching technique consists then in consecutively applying the SIFT algorithm to each keypoint individually, considering the image to be only a small area around it (in other words, locally applying the algorithm), until the update process is completed.

![Figure 4.4: Block diagram of the object tracking method: variant 2, also called the SLS variant.](image)

Evidently, this second variant can be successfully applicable to all natures of video sequences, including the ones best suited to the one presented in the previous section. However, since the SIFT algorithm, even when applied locally, is slower than both LSM and NCC procedures, the frame sampling rate would be lower, leaving more frames unprocessed. In fact, if applying both methods to the same low movement, not severely compressed, sequence and then playing
only the processed frames as a movie, the video made using the frames output
by the second variant would appear rather jerky, when compared to the other.

Analogously, this variant will be referred to as the SLS variant from
hereon.

4.3 General remarks on performance

Commenting on the overall method in terms of programming code, there
are two identical structures that carry keypoint collections, one referred as
the updating list and the other, as the tracking list. Whenever an update is
ready, they simply switch their memory addresses, which makes the refreshing
of the tracking keypoints virtually instantaneous. Moreover, since multi-core
processors have become quite dominating, the parallelism of the method
proposed herein is easily implemented.

In addition, the greatest stumbling block, regarding the method’s per-
formance in both variants, is the processing time of the matching procedures.
Thus, as the CPU clock frequency increases, so does the quality of the results
achieved by the object tracking algorithm presented in this chapter, both in
number of frames processed and location accuracy, the latter being a direct
consequence of the former. Moreover, faster machines also lead to an increase
in the collection updating rate, hence the consistency of the keypoint lists with
the changing scene escalates, as well, which further contributes to the method’s
accuracy.