

experimental comparison of some of these algorithms on image retrieval datasets. You can also find more details on related techniques and systems in Section 6.2.3 on visual similarity search, which discusses global descriptors that represent an image with a single vector (Arandjelovic, Gronat *et al.* 2016; Radenović, Tolias, and Chum 2019; Yang, Kien Nguyen *et al.* 2019; Cao, Araujo, and Sim 2020; Ng, Balntas *et al.* 2020; Tolias, Jenicek, and Chum 2020) as alternatives to bags of local features, Section 11.2.3 on location recognition, and Section 11.4.6 on large-scale 3D reconstruction from community (Internet) photos.

7.1.5 Feature tracking

An alternative to independently finding features in all candidate images and then matching them is to find a set of likely feature locations in a first image and to then search for their corresponding locations in subsequent images. This kind of *detect then track* approach is more widely used for video tracking applications, where the expected amount of motion and appearance deformation between adjacent frames is expected to be small.

The process of selecting good features to track is closely related to selecting good features for more general recognition applications. In practice, regions containing high gradients in both directions, i.e., which have high eigenvalues in the auto-correlation matrix (7.8), provide stable locations at which to find correspondences (Shi and Tomasi 1994).

In subsequent frames, searching for locations where the corresponding patch has low squared difference (7.1) often works well enough. However, if the images are undergoing brightness change, explicitly compensating for such variations (9.9) or using *normalized cross-correlation* (9.11) may be preferable. If the search range is large, it is also often more efficient to use a *hierarchical* search strategy, which uses matches in lower-resolution images to provide better initial guesses and hence speed up the search (Section 9.1.1). Alternatives to this strategy involve learning what the appearance of the patch being tracked should be and then searching for it in the vicinity of its predicted position (Avidan 2001; Jurie and Dhume 2002; Williams, Blake, and Cipolla 2003). These topics are all covered in more detail in Section 9.1.3.

If features are being tracked over longer image sequences, their appearance can undergo larger changes. You then have to decide whether to continue matching against the originally detected patch (feature) or to re-sample each subsequent frame at the matching location. The former strategy is prone to failure, as the original patch can undergo appearance changes such as foreshortening. The latter runs the risk of the feature drifting from its original location to some other location in the image (Shi and Tomasi 1994). (Mathematically, small misregistration errors compound to create a *Markov random walk*, which leads to larger drift over time.)

experimental comparison of some of these algorithms on image retrieval datasets. You can also find more details on related techniques and systems in Section 6.2.3 on visual similarity search, which discusses global descriptors that represent an image with a single vector (Section 6.2.3.1) and local descriptors that represent an image with multiple vectors (Section 6.2.3.2). Ng et al. 2019; Cao, Araya, and Sun 2020; Ng, Baluja et al. 2020; Tolias, Jeniak, and Chum 2020 as alternatives to bags of local features, Section 11.2.3 on location recognition, and Section 11.4.6 on large-scale 3D reconstruction from community internet photos.

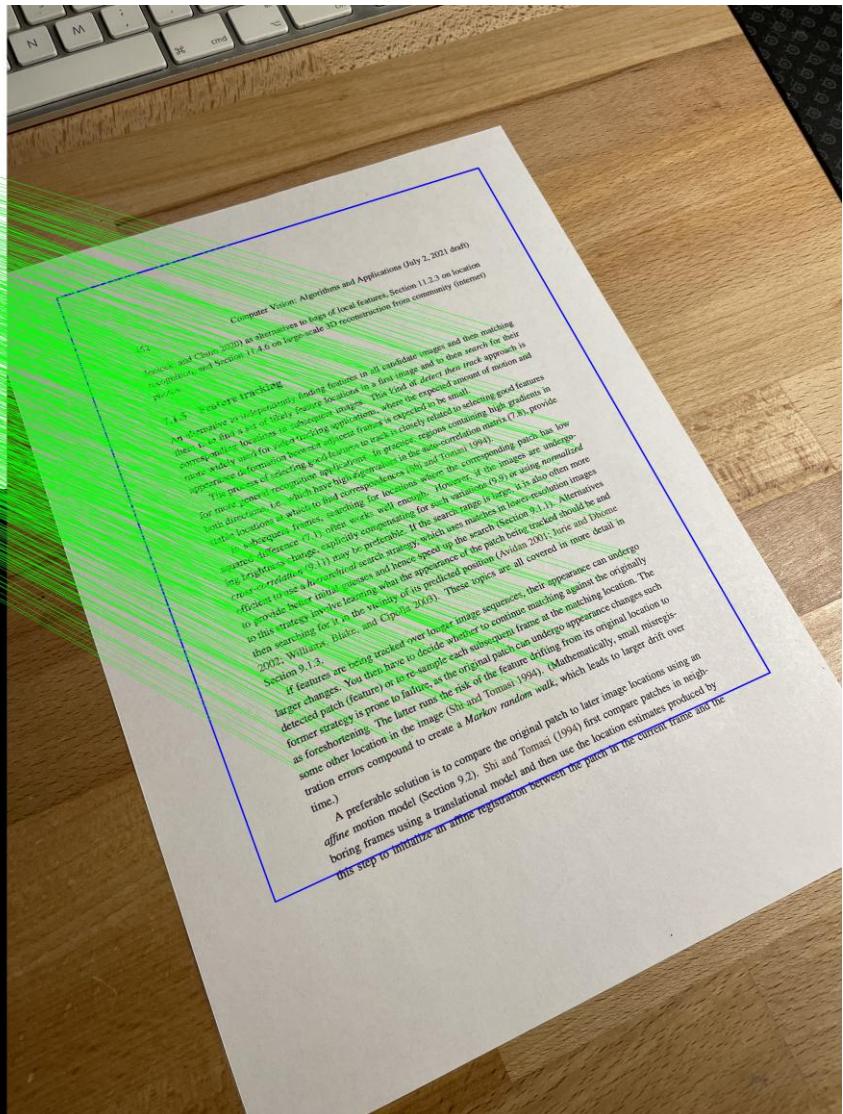
7.1.8. Feature tracking

An alternative to independently finding features in all candidate images and then matching them is to first find a set of initial feature locations in a first image and then search for their corresponding locations in subsequent images. This kind of approach is most appropriate and more widely used for video tracking applications, where the expected amount of motion and appearance deformation between adjacent frames is relatively small.

One way to implement feature tracking is to use a coarse-to-fine approach, starting with high resolution for most general recognition up to a frame and then gradually decreasing resolution in both directions, i.e., which has higher resolutions in the center and lower resolution at the most probable stable location at which to find feature tracks (see also Figure 10.4).

In subsequent frames, searching for features along the same trajectory leads to both spatial offset errors (i.e., other words, drift) through motion blur if the image is moving and/or being scaled, as well as appearance changes due to motion blur or camera motion. Using camera calibration (Section 10.1) can be preferable. If the search range is large, it is often more efficient to use adaptive local search techniques, such as random walks, as in the approach described to process image sequences and handle changes in the image background (Li et al. 2002; Williams, Blau, and Cappe 2004). These topics are all covered in more detail in Section 9.1.3.

Another way to implement feature tracking is to use a coarse-to-fine approach, but applying it to individual patches rather than to the entire image. You can have a smaller search area for each patch, which can result in better performance, as the former strategy is prone to failure, as the effect of patch motion under scaling and/or camera motion is compounded by motion blur and scale change (see also Figure 10.4).



Here **SIFT** is used for feature matching.

experimental comparison of some of these algorithms on image retrieval datasets. You can also find more details related techniques and systems in Section 6.2.3 on visual similarity search, which discusses global descriptors that represent [an image](#) ([v1/y2/2021/draft](#)) (Arandjelovic, Gronat *et al.* 2016; Radenović, Tolias, and Chum 2019; Yang, Kien Nguyen *et al.* 2019; Cao, Araújo, and Sim 2020; Ng, Balntas *et al.* 2020); [location](#) ([v1/2.5/on/location/2020](#)) as alternatives to bags of local features, Section 11.2.3 on [location](#) ([community](#) ([internet](#))) Section 11.4.6 on large-scale 3D reconstruction from community ([internet](#)) photos.

7.1.5 Feature tracking

An alternative to independently finding features in all candidate images and then matching them is to find a set of likely feature locations in a first image and to then *search* for their corresponding locations in subsequent images. This kind of *detect then track* approach is more widely used for video tracking applications, where the expected amount of motion and appearance deformation between adjacent frames is expected to be small.

The process of selecting good features to track is closely related to selecting good features for more general recognition applications. In practice, regions containing high gradients in both directions, i.e., which have high eigenvalues in the auto-correlation matrix (7.8), provide stable locations at which to find correspondences (Shi and Tomasi 1994).

In subsequent frames, searching for locations where the corresponding patch has low squared difference (7.1) often works well enough. However, if the images are undergoing brightness change, explicitly compensating for such variations (9.9) or using *normalized cross-correlation* (9.11) may be preferable. If the search range is large, it is also often more efficient to use a *hierarchical* search strategy, which uses matches in lower-resolution images to provide better initial guesses and hence speed up the search (Section 9.1.1). Alternatives to this strategy involve learning what the appearance of the patch being tracked should be and then searching for it in the vicinity of its predicted position (Avidan 2001; Jurie and Dhume 2002; Williams, Blake, and Cipolla 2003). These topics are all covered in more detail in Section 9.1.3.

If features are being tracked over longer image sequences, their appearance can undergo larger changes. You then have to decide whether to continue matching against the originally detected patch (feature) or to re-sample each subsequent frame at the matching location. The former strategy is prone to failure, as the original patch can undergo appearance changes such as foreshortening. The latter runs the risk of the feature drifting from its original location to some other location in the image (Shi and Tomasi 1994). (Mathematically, small misregistration errors compound to create a *Markov random walk*, which leads to larger drift over time.)

see the original patch to later image locations using an [SIFT](#) model and then use the location estimates produced by

