



UiA Faculty of Engineering
and Science

Ikt 213

Lab assignment 4

by

Erling Abrahamsen

Gruppe 7

Faculty of Engineering and Science

University of Agder

Grimstad, Nov 2025

Innhold

Harris Corner Detection and Feature-Based Image Alignment	2
Introduction	2
Harris Corner Detection	2
Feature-Based Image Alignment Using SIFT	2
Output	2
Conclusion	2
References	2
Pictures	3

Harris Corner Detection and Feature-Based Image Alignment

Introduction

This assignment explores two image processing techniques: Harris Corner Detection and feature-based alignment using SIFT. The goal is to detect key points in an image and, when possible, align another image to the reference. The reference image used is `reference_img.png`.

Harris Corner Detection

Harris Corner Detection identifies points in an image where intensity changes sharply, marking corners and edges. The output, `harris.png`, highlights these points, showing visually the areas of interest that are stable and distinctive for further processing.

Feature-Based Image Alignment Using SIFT

SIFT detects distinctive keypoints that are robust to scale and rotation. Feature-based alignment uses these points to match and transform a secondary image so that its features align with the reference. The output is an aligned image where objects or features match the reference visually.

Output

- **Harris image (`harris.png`):** Shows detected corners and edges in the reference image.
- **Aligned image:** Demonstrates how a secondary image can be matched to the reference using feature points.

Conclusion

Harris Corner Detection provides a quick way to find key points in an image, while SIFT-based alignment allows images to be matched and aligned based on those points. The outputs visually demonstrate the techniques' effectiveness and their potential for more advanced image analysis tasks.

References

- OpenCV Harris Corner Detection: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html
- Lowe, D.G., Distinctive Image Features from Scale-Invariant Keypoints, IJCV, 2004.

Pictures

452 Computer Vision: Algorithms and Applications, 2nd ed. (final draft, Sept. 2021)

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, Jeníček, 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 Dhome 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.)

Figur 1: Reference_img.png

452 Computer Vision: Algorithms and Applications, 2nd ed. (final draft, Sept. 2021)

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 (Arandjelović, 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, Jenkoek, 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 Dhome 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.)

Figur 2: Harris.png

452 Computer Vision: Algorithms and Applications, 2nd ed. (final draft, Sept. 2021)

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, Jeníček, 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 Dhome 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.)

Figure 3: aligned_iimage.png