2D and 3D feature-based alignment

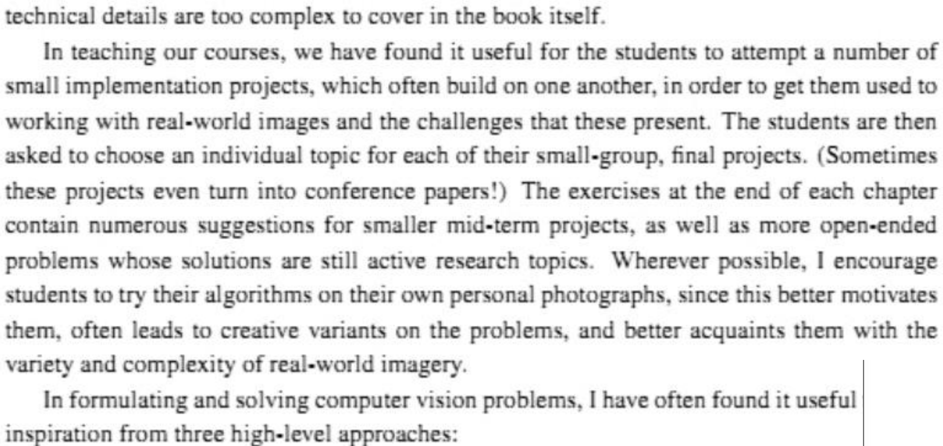
Objectives

- Learn about feature-based alignment
- Learn about 2D alignment using least squares, as well as some related algorithms
- Understand how 3D alignment works
- Their applications

Feature-Based Alignment

- After extracted features from images → match these features across different images(the set of matching features is geometrically consistent)
- Feature-based alignment is the problem of estimating the motion between two or more sets of matched 2D or 3D points.
- In the technique, a sparse set of features are detected in one image and matched with the features in the other image. A transformation is then calculated based on these matched features that warp one image onto the other.

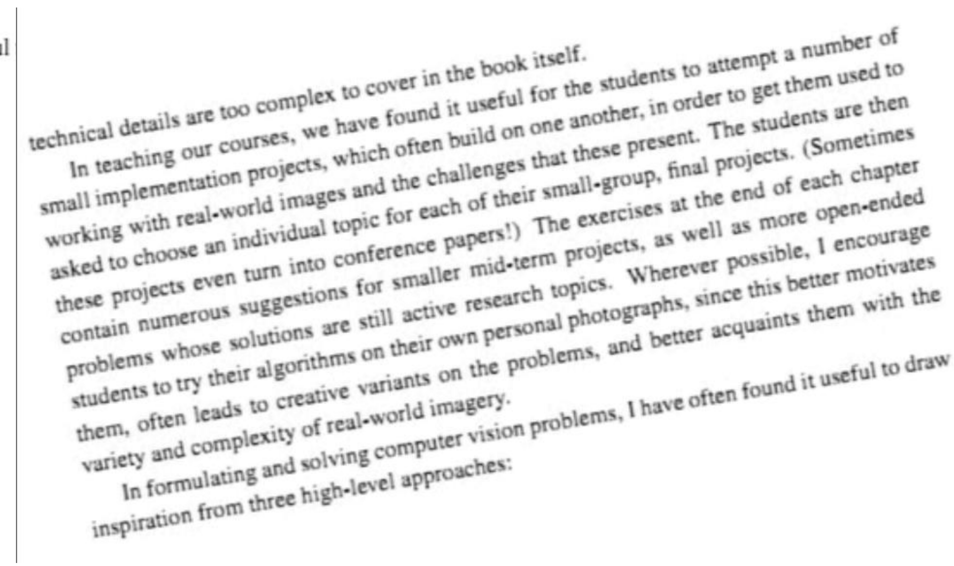
Feature-Based Alignment


technical details are too complex to cover in the book itself.

In teaching our courses, we have found it useful for the students to attempt a number of small implementation projects, which often build on one another, in order to get them used to working with real-world images and the challenges that these present. The students are then asked to choose an individual topic for each of their small-group, final projects. (Sometimes these projects even turn into conference papers!) The exercises at the end of each chapter contain numerous suggestions for smaller mid-term projects, as well as more open-ended problems whose solutions are still active research topics. Wherever possible, I encourage students to try their algorithms on their own personal photographs, since this better motivates them, often leads to creative variants on the problems, and better acquaints them with the variety and complexity of real-world imagery.

In formulating and solving computer vision problems, I have often found it useful

inspiration from three high-level approaches:


technical details are too complex to cover in the book itself.

In teaching our courses, we have found it useful for the students to attempt a number of small implementation projects, which often build on one another, in order to get them used to working with real-world images and the challenges that these present. The students are then asked to choose an individual topic for each of their small-group, final projects. (Sometimes these projects even turn into conference papers!) The exercises at the end of each chapter contain numerous suggestions for smaller mid-term projects, as well as more open-ended problems whose solutions are still active research topics. Wherever possible, I encourage students to try their algorithms on their own personal photographs, since this better motivates them, often leads to creative variants on the problems, and better acquaints them with the variety and complexity of real-world imagery.

In formulating and solving computer vision problems, I have often found it useful to draw inspiration from three high-level approaches:

2D alignment using least squares

- Give a set of matched feature points:

$$\{(\boldsymbol{x}_i, \boldsymbol{x}'_i)\}$$

  – where xi – po[illegible]nage, x'i – point in the other image

- A transformation:

$$\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$$

  – where f – transformation function, p is parameters

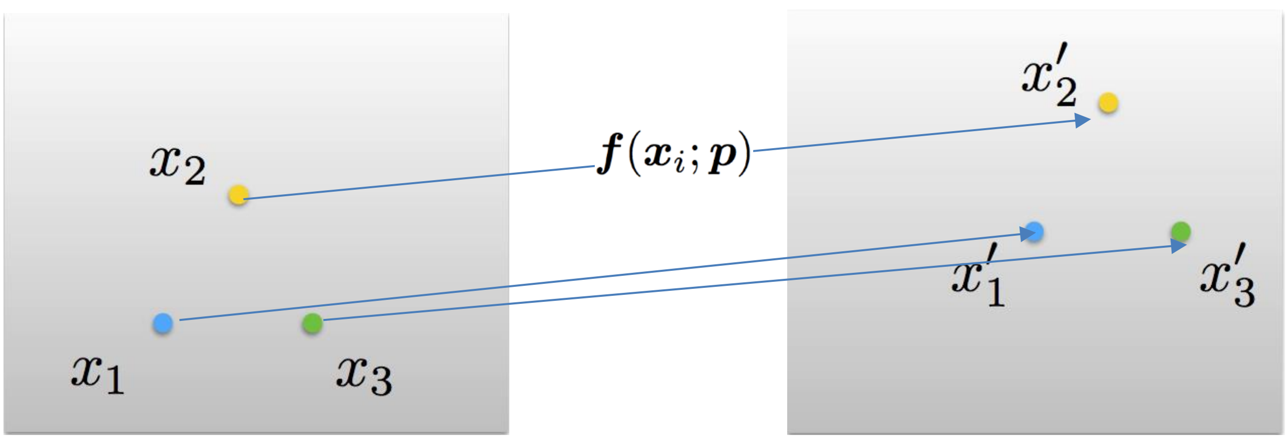
- How to find the best estimate of p?→ Least Squares Error:

$$E_{\mathrm{LS}} = \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p}) - \boldsymbol{x}'_i\|^2$$

2D alignment using least squares

- Find the parameter that minimize squared error:

$$\hat{\boldsymbol{p}} = \arg\min_{\boldsymbol{p}} \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i;\boldsymbol{p}) - \boldsymbol{x}'_i\|^2$$



FPT Education FPT UNIVERSITY

- Uncertainty weighting→ minimize the weighted least squares problem:

$$E_{\mathrm{WLS}} = \sum_i \sigma_i^{-2} \|\boldsymbol{r}_i\|^2$$

Iterative algorithms

- Most problems in computer vision do not have a simple linear relationship between the measurements and the unknowns
- The resulting problem is called non-linear least squares
- To minimize the non-linear least squares problem:

$$
\begin{aligned}
E_{\mathrm{NLS}}(\Delta \boldsymbol{p}) &= \sum_i \|\boldsymbol{f}(\boldsymbol{x}_i; \boldsymbol{p} + \Delta \boldsymbol{p}) - \boldsymbol{x}_i'\|^2 \\
&\approx \sum_i \|\boldsymbol{J}(\boldsymbol{x}_i; \boldsymbol{p}) \Delta \boldsymbol{p} - \boldsymbol{r}_i\|^2 \\
&= \Delta \boldsymbol{p}^T \left[\sum_i \boldsymbol{J}^T \boldsymbol{J}\right] \Delta \boldsymbol{p} - 2 \Delta \boldsymbol{p}^T \left[\sum_i \boldsymbol{J}^T \boldsymbol{r}_i\right] + \sum_i \|\boldsymbol{r}_i\|^2 \\
&= \Delta \boldsymbol{p}^T \boldsymbol{A} \Delta \boldsymbol{p} - 2 \Delta \boldsymbol{p}^T \boldsymbol{b} + c,
\end{aligned}
$$

Robust least squares

- Robust least squares refer to a variety of regression methods designed to be robust, or less sensitive, to outliers:
  - M-estimation addresses dependent variable outliers where the value of the dependent variable differs markedly from the regression model norm
  - S-estimation is a computationally intensive procedure that focuses on outliers in the regressor variables.
  - MM-estimation is a combination of S-estimation and M-estimation. The procedure starts by performing S-estimation, and then uses the estimates obtained from S-estimation as the starting point for M-estimation.

RANdom SAmple Consensus- RANSAC

- RANSAC is a parameter estimation approach designed to cope with a large proportion of outliers in the input data
- RANSAC is a resampling technique that generates candidate solutions by using the minimum number of data points required to estimate the underlying model parameters.

RANSAC Algorithm

- 1. Select N data items at random
- 2. Estimates parameter $\vec{x}$
- 3. Finds how many data items (of M) fit the model with parameter vector $\vec{x}$ within a user given tolerance. Call this K.
- 4. If K is big enough, accept fit and exit with success.
- 5. Repeat 1..4 L times
- 6. Fail if you get here

RANSAC Algorithm

- The likelihood that S such trials will all fail is

$$1 - P = (1 - p^k)^S$$

- The required mini

$$S = \frac{\log(1 - P)}{\log(1 - p^k)}$$

3D alignment

- If the 3D transformations are linear in the motion parameters → regular least squares can be used.
- The case of rigid (Euclidean) motion

$$E_{\mathrm{R3D}} = \sum_i \|\boldsymbol{x}'_i - \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{t}\|^2$$

3D alignment

- The orthogonal Procrustes algorithm
- The absolute orientation algorithm
- → find that the difference in accuracy is negligible

Applications

- Document processing: align the scanned or photographed document to a template.
- Medical applications: multiple scans of tissue may be taken at slightly different times and the two images need to compare.
- Creating panoramas image

Summary

- Learn about feature-based alignment
- Learn about 2D alignment using least squares, as well as some related algorithms
- Understand how 3D alignment works
- Their applications