Feature-based Image Matching

Image and Speech Recognition Project

25/01/2013 - *Sura Al-Bayaty, Michael Clement*

Subject

Detection of similar images in a set of images, where one of them is the distorted (e.g. cropped, rotated or with enhanced saturation copy of the original image. In order to achieve this goal, divide images, extract a set of distortion-independent features and perform a block to block matching.

Particular steps required:

(P1, P2) Propose a set of image distortions.

(P1) Propose and implement a method of features extraction.

(P2) Implement the feature-based matching method.

(P1, P2) On-line visualization of intermediate and final results.

Chapters

Introduction

The goal of this project is a design a feature-based matcher detecting similar images where one of them has been distorted. This can be made in two main steps: extract features from the images, and match them together to display the similarities.

Because we are talking about altered images, this matching process must be as distortion-independent as possible. We used several well-known computer vision algorithms to extract the features and to match them together. This documentation is made of several parts:

- First, we describe which kind of image distortions we will be working on.
- Next, we present the feature extraction step. We used SIFT and SURF, and we briefly present how they work.
- Then, we describe the feature matching algorithm we used, FLANN.
- We finish with results analysis to conclude.

Image distortions

In this section we introduce interesting image distortions for the project.

Cropping

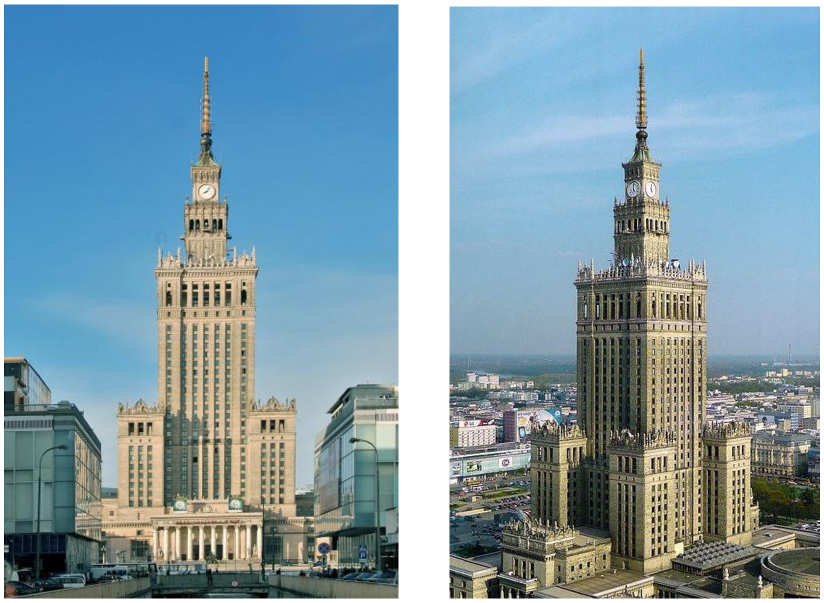
Cropping is the action of removing parts from a picture: we cut the image and keep only a small part of it. In photography, it is usually used to remove unwanted parts from an image, or to emphasize on an object of the image.

Example of cropping used to emphasize on an object

In our case, we will be given an image and its cropped version, and we are going to try to recognize the cropped image in the original one. In fact, if the cropping is made on a generic object (for example, a chair, a lamp, a face, etc.), we could use the cropped image as a template, in order to recognize this object in several different pictures.

Rotation

There are different types of rotations for an image. Seen from the photographer point of view, you can either rotate the camera without moving, or take the picture from a different angle by walking around.

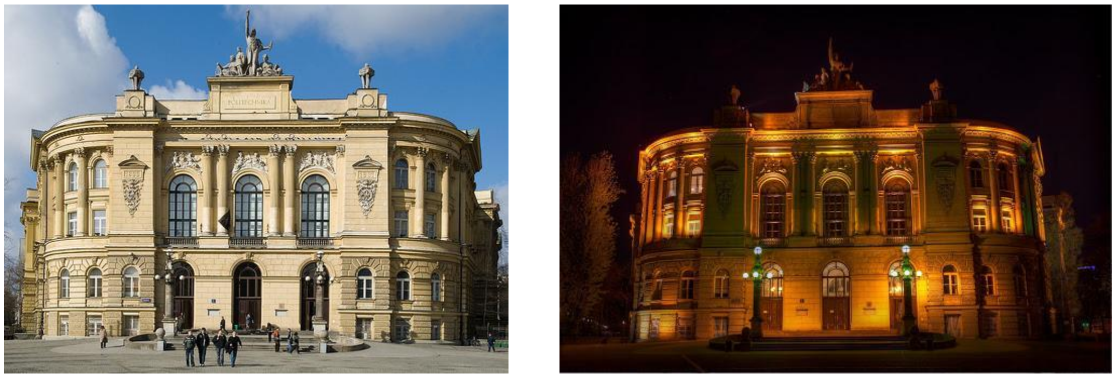


Palace of Culture and Science seen from different angles

Here, we will focus on detection of images which are taken from different angles. It has a lot of applications, such as, for instance, building panoramas using several pictures taken at different locations.

Illumination

The idea of this distortion is to detect images where the lighting is not the same. For example, as seen on the pictures below, we want to detect that this is the same building even though one of them is taken during the day and the other one at night.



Warsaw University of Technology, main building by day and by night

Feature extraction

Features are informally defined as interesting points of an image. By interesting, we usually mean points that can be used to detect and describe objects and characteristics of the image. Image features are usually:

- **Edges** – Delimitation of objects and regions of the image. Edge points are characterized by strong gradient values. We usually chain those points together in order to form actual edges.
- **Corners** – Corners cover both corners in its literal meaning, which are rapid changes in edges directions, and also isolated points which have interesting local properties, like completely different color from its neighbors, for example.
- **Blobs** – They are regions of close interest points.

Feature extraction is the process where we automatically examine an image to extract its interesting points as described above. In order to extract similar features in different images, we want those features to be as much distortion-independent as possible. Several algorithms exist to extract such invariant features from images. This feature extraction process can be divided in two stages:

- **Detection** – Automatically identify interest points, the same feature should always be detected regardless of the point of view.
- **Description** – Each interest point should have a unique description that does not depend on the features scale and rotation.

Technically speaking, we are going to use the OpenCV library to perform the feature extraction. This library comes with built-in well-known algorithms like SIFT and SURF. In fact, they were designed with the aim to be invariant to distortions. Depending on the distortion, it might be better to extract features from the images using one algorithm or the other.

Scale-Invariant Feature Detection (SIFT)

SIFT is an algorithm performing the feature extraction process described above. SIFT features are based on the appearance of the object at particular interest points. It transforms the image into a collection of feature vectors invariant to translation, scaling and rotation. Key locations are defined using Difference of Gaussians (DoG) based on several steps to ensure the key points are more stable for recognition.

Feature detection

Potential interest points are determined by computing the extrema of Difference of Gaussian. DoG is a band pass filter discarding all but handful frequencies present in the original image. In order to search these extrema, each pixel of the image is compared with its neighborhood of 3*3 size. If the pixel is lower/larger than all its neighbors, then it is labeled as a candidate keypoint. Each of these keypoints is exactly localized by fitting a quadratic function computed using a second order Taylor expansion around the neighborhood. Hence, keypoints are obtained by discarding low contrast pixels and pixels that correspond to edges [2].

Feature orientation

An orientation in assigned to each candidate keypoint based on its gradient data. For a candidate keypoint pixel in a certain region, the first-order gradients are calculated like this:

$$g_x = L(x+1, y, \sigma) - L(x-1, y, \sigma)$$
$$g_y = L(x, y+1, \sigma) - L(x, y-1, \sigma)$$

Where $L(x, y, \sigma)$ is the value of the pixel $p(x, y)$ in the image blurred by a Gaussian Kernel whose size is determined by σ.

The gradient magnitude and orientation are respectively computed as follows:

$$m(x,y) = \sqrt{{g_x}^2 + {g_y}^2}$$

$$\theta(x,y) = \arctan\left(\frac{g_y}{g_x}\right)$$

An orientation histogram with 36 bins is formed, with each bin covering 10 degrees. Each sample in the neighboring window is added to a histogram. Bin is weighted by its gradient magnitude and by a Gaussian-weighted circular window with that is 1.5 times that of the scale of the key-point. The peaks in this histogram correspond to dominant orientations.

Once the histogram is filled, the orientations corresponding to the highest peak and local peaks that are within 80% of the highest peaks are assigned to the key-point. In the case of multiple orientations being assigned, an additional key-point is created having the same location and scale as the original key-point for each additional orientation [2].

Feature description

Descriptor is computed at each key-point based on the gradient data. The region around the keypoint is divided into 16 square boxes. For each box an eight bin orientation histogram is calculated from gradient data of pixels within the corresponding box relative to the feature orientation to provide rotation invariance. Finally, all 16 resulted eight bin orientation histograms is transformed into 128-D vector. The vector is normalized to unit length to achieve the invariance against illumination changes [2].

Speeded-Up Robust Features (SURF)

SURF uses a Hessian-based blob detector to find interest points. The determinant of a Hessian matrix is an expression of the local change around the area.

$$H(x,\sigma) = \begin{matrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{matrix}$$

Where $L_{xx}(x,\sigma)$ is the convolution kernel of the image with the second derivative of the Gaussian. The core of SURF feature detection is non-maximal-suppression of the determinants of the Hessian matrices. The convolutions are approximated and speeded-up with the used of integral images and approximated kernels.

Where the integral image $I(x)$ is the image where each point x=$(x,y)^T$ store the summation of all pixels in a rectangular area.

$$I(x) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(x,y)$$

The second order Gaussians kernels used for hessian matrix, SURF algorithm approximate the kernels in rectangular boxes, box filters:

$$Det(H_{approx}) = D_{xx}D_{xy} - (\omega D_{xy})^2$$

Kernels D_{yy} for $L_{yy}(x,\sigma)$ and D_{xy} for $L_{xy}(x,\sigma)$ using the approximated kernels to calculate the determinant of the Hessian matrix - we have to weight it with w which is sensitive to scale but can kept constant at 0.9.

To detect features across scale we have to examine several octaves and levels. The integral images allow the SURF algorithm to calculate the responses with arbitrary large kernels. When finding an extrema at one of the highest octaves, the area covered by the filter is rather large and this introduces a significant error for the position of the keypoint. To remedy this, the exact locations of the interest points are interpolated by fitting a 3D quadratic in scale space. A keypoint is located in scale space by (x,y,s) where x, y are relative coordinates and s is the scale [1].

Feature description

SURF descriptor based on Haar wavelet responses to provide unique description of a feature and can be calculated efficiently with integral images based on area surrounding a keypoint, by determine a unique orientation for the key-point achieving rotational invariance, so that the key-area (surrounding the keypoint) is rotated to its direction.

The interest area in SURF is of size 20s divided in 4x4 sub areas of x and y directions. The wave let referred to as dx and dy and weighting these areas with Gaussian centered at the keypoint to give robustness for deformation and translation.

$$v = \{\sum dx , \sum |dx| , \sum dy , \sum |dy|\}$$

Vector v calculated for each subarea based on 5x5 samples. The descriptor for a key-point is the 16 vectors for the subareas concatenated. Finally the descriptor is normalized, to achieve invariance to contrast variations that will represent themselves as a linear scaling of the descriptor [3].

Feature matching

Once features have been extracted from our image by one of the methods described above, we have plenty of feature vectors at our disposal in each image, and we want to perform a matching between them in order to recognize similar parts.
Here again, this matching can be done in plenty of ways. *SURF* and *SIFT* are also commonly used for feature matching, but apparently these parts are not implemented in the *OpenCV*. Instead, we will with the built-in *FLANN* feature matcher.

Fast Library for Approximate Nearest Neighbors (FLANN)

FLANN is a library for performing fast approximate nearest neighbor searches in high dimensional spaces. It contains a collection of algorithms to work best for nearest neighbor search and a system for automatically choosing the best algorithm and optimum parameters depending on the dataset.

Automatic algorithm configuration allows a user to achieve high performance in approximate nearest neighbor matching by calling a single library routine. The user only needs to provide an example of the type of dataset that will be used and the desired precision. All remaining steps of algorithm selection and parameter optimization are performed automatically [4].

We can define the nearest neighbor search problem in the following way: given a set of points *P = p1, p2, , , , , pn.* in a metric space *X*, these points must be preprocessed in such a way that given a new query point $q \in x$, finding the point in *P* that is nearest to *q* can be done quickly.

The accuracy of the approximation is measured in terms of precision, which is defined as the percentage of query points for which the correct nearest neighbor is found, one of two algorithms obtained the best performance depending on the dataset and desired precision. These algorithms used either the hierarchical k-means tree or multiple randomized kd-trees.

Hierarchical k-means tree algorithm

The hierarchical k-means tree is constructed by splitting the data points at each level into K distinct regions using a k-means clustering, and then applying the same method recursively to the points in each region. We stop the recursion when the number of points in a region is smaller than K.

Randomized kd-tree algorithm:

It is the improved version of the classical kd-tree in which multiple normalized kd-tree are created, the randomized trees are built by choosing the split dimension randomly from the first D dimensions on which data has the greatest variance. When searching the trees, a single priority queue is maintained across all the randomized trees so that search can be ordered by increasing distance to each bin boundary. The degree of approximation is determined by examining a fixed number of leaf nodes, at which point the search is terminated and the best candidates returned.

Results Analysis

We have implemented this feature-based image matcher using the OpenCV library. The application allows you to select two sample images and to run the feature-based matcher on them. You have the choice between extracting features with SIFT or SURF. The application then displays the two images side by side and shows their shared similar keypoints with lines.

The answer whether the two images are similar is really arbitrary. This diagnosis can be made in several ways:

- *Count the number of matchings.* The number of similar keypoints between the two sample images is a good measurement of their similarity. However, sometimes we have some completely different images with a lot of similar keypoints, thus making this criteria only partially representative of the similarity.
- *Check the relevance of matchings.* For a given matching, we want to check if the keypoint in each sample image corresponds to the same part of the image (for example the same corner in both sample images). This diagnosis has to be done mostly manually and is therefore highly subjective.

Due to these somehow inconstant ways to diagnose the similarity, the best way to analyze the performance of the algorithms is to test sample images empirically. We did it for each distortion.

In general, SURF extracts more features than SIFF, resulting in more matchings found overall. However, SIFT ones are often more likely to be relevant. Therefore, we think that SURF is more resistant to distortions in general, but also introduces matches which are not interesting.

Cropping

SIFT stands for “Scale-Invariant” and thus has been designed to be invariant to cropping. The algorithm performed well on our image samples showing cropping: similar keypoints are detected. SURF was also quite good at detecting similar keypoints in a cropped image.

In general, whether it was just cropping or cropping + zoom, both algorithms performed really well for these distortions. It is not a big surprise considering that they were designed to be completely invariant to scaling distortions.

Rotation

Rotation is a harder distortion to be invariant to. We got satisfying results for this distortion with algorithms: similar keypoints were detected in both the original sample images and their rotated versions, even when with several sample images taken from different angles. We can conclude that SIFT and SURF are pretty robust to rotation.

Illumination

Results on the illumination distortion are mixed. The algorithms manage to extract features and to find matches, especially with SURF, but a lot of them are mistaken. Those results were somehow expected, because it is quite known that illumination is one of the hardest distortions to be robust to.

References

[1] *Study group SURF: Feature detection & description, Jacob Toft Pedersen 19983275, jtp@cs.au.dk.*

[2] *SPEEDED UP IMAGE MATCHING USING SPLIT AND EXTENDED SIFT FEATURES, Faraj Alhwarin, Danijela Ristić–Durrant and Axel Gräser, Institute of Automation, University of Bremen, Otto-Hahn-Alle NW1, D-28359 Bremen, Germany, {alhwarin, ristic, ag}@iat.uni-bremen.de.*

[3] *SURF: Speeded Up Robust Features, Herbert Bay1, Tinne Tuytelaars2, and Luc Van Gool12, 1 ETH Zurich, {bay, vangool}@vision.ee.ethz.ch, 2 Katholieke Universiteit Leuven, {Tinne.Tuytelaars, Luc.Vangool}@esat.kuleuven.be.*

[4] *FAST APPROXIMATE NEAREST NEIGHBORS WITH AUTOMATIC ALGORITHM CONFIGURATION , Marius Muja, David G. Lowe, Computer Science Department, University of British Columbia, Vancouver, B.C., Canada, mariusm@cs.ubc.ca, lowe@cs.ubc.ca.*