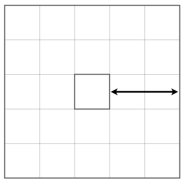
Lab 1

# The SIFT descriptor

In this lab you will implement a SIFT-like descriptor and try it on some classification tasks. Keep your code well commented, both for your own sake and because it is required.

**Ex 1.1** Make a function

```
patch = get_patch(image, x, y, patch_radius)
```



that takes out a quadratic patch from the image centred at $(x, y)$ with a range `x/y` $\pm$ `patch_radius`. (The figure on the right shows a patch with `patch_radius` two pixels.) Make sure that your function works both for grayscale and color images.

**Ex 1.2** Use

```
test_image = reshape((11:100), 10, 9)
```

to create a test image with numbered pixels. Try extracting a few patches from this image to verify that your function works. Make sure that the $x$-variable corresponds to the column index and the $y$-variable to the row index.

**Ex 1.3** Modify `get_patch` so it returns an error with an informative error message such as

```
'Patch outside image borders'
```

if the patch doesn't fit inside the image. Use the `error` function for this.

## Gradient histograms

**Ex 1.4** Create a Matlab function `gaussian_filter` that takes two arguments, one grayscale image and one number specifying a standard deviation. The output should be the image filtered with a Gaussian filter of the specified standard deviation. Example usage:

```
result = gaussian_filter(img, 3.0);
```

The filter size should be at least four standard deviations not to loose precision. Use `fspecial` to construct a Gaussian filter. It is a good idea to use the `'symmetric'` option with `imfilter`. (Do not use the built-in function for Gaussian filtering.)

**Ex 1.5** Make a function

```
[grad_x, grad_y] = gaussian_gradients(img, std)
```

that takes a grayscale image and estimates both gaussian derivatives for each pixel. Use filtering with derivative filters and your function from the previous exercise. The output should be two matrices of same size as the input image. Be careful about the definition of `x` and `y`.

**Ex 1.6** Plot your gradients in the image using

```
imagesc(img)
axis image
hold on
quiver(grad_x, grad_y)
```

and verify visually that the gradients are correct.

**Ex 1.7** Make a function

```
histogram = gradient_histogram(grad_x, grad_y)
```

that places each gradient into one of eight orientation bins. A useful function is `atan2` in Matlab. Use the bins and bin order from the lecture notes. The provided `plotBouquet` lets you plot the histograms as a bouquet of vectors and might be helpful for debugging. It assumes that you have given all functions exactly the names suggested here and used the bin ordering from the lecture notes.

## A SIFT-like descriptor

Next, we will create a SIFT-like descriptor by computing gradient histograms for $3 \times 3$ regions and stacking them into a vector. The exact positions and sizes of the regions are not crucial. For example, you might choose whether to use overlapping regions or not. Figure 1.1 shows an example of how to place the nine regions. You can use the provided `paper_with_digits.png` as an example image. For example, there is a digit at $(1290, 950)$.

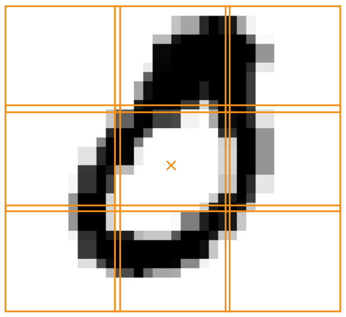


Figure 1.1: Example patch placement. The $x$ marks the coordinates in the argument position .

**Ex 1.8** Make a function `region_centres = place_regions(centre, radius)` that creates $3 \times 3$ square regions for the descriptor; see Figure 1.1. The input `radius` specifies the *radius* of each region (as in Ex 1.1). The output should be an $2 \times 9$ array with the centre points of each region. Use the provided function

```
plot_squares(img, region_centres, radius)
```

to plot your regions in an example image. Increasing the input `radius` with a factor $K$ should scale the whole region pattern with a factor $K$. Save a screenshot of the plotted regions for the report.

**Ex 1.9** Make a function

```
desc = gradient_descriptor(image, position, radius)
```

that computes a SIFT-like descriptor at a certain position. The input `radius` controls the size of the regions just as in Ex 1.8.

- Compute gaussian gradients. Let the standard deviation be proportional to `radius`.
- Divide your gradients into $3 \times 3$ regions defined by `place_regions`.
- Compute a gradient histogram for the gradients from each region.
- Stack the histograms into a 72-vector.
- Normalize that vector to unit length.

## Digit classification

In the file `digits.mat`, there are two lists of images, `digits_training` with 100 training images and `digits_validation` with 50 similar images. Our next goal is to classify each of the validation images by finding the most similar image in the training set and assuming that the query image has the same label. Load the digit data by running

```
load digits.mat
```

The examples are stored in a struct array. To get image number `12` you write `digits_training(12).image` and to get its label you write `digits_training(12).label`.

**Ex 1.10** Make a script `prepare_digits.m` that computes a descriptor for each digit in `digits_training`. You need to choose the `position` and `radius` parameters so that all the descriptor regions fit into the images. Store the descriptors in an appropriate place. A suggestion is to store the 12th descriptor in

```
digits_training(12).descriptor
```

**Ex 1.11** Make a function

```
label = classify_digit(digit_image, digits_training)
```

that computes a descriptor for the given digit image, goes through all the digits in `digits_training` to find the one with the most similar descriptor and outputs the label of that digit.

You can use `disp` to display text in matlab. For example

```
disp(['I am ' num2str(age) ' years old'])
```

will display your age, assuming that it is stored in the variable `age`.

**Ex 1.12** Make a script `classify_all_digits` that runs `classify_digit` for each of the digits in `digits_validation` and displays the percentage of correct answers.

**Ex⋆ 1.13** Try to classify a few of the large digits in `paper_with_digits.png`. For example there are digits at

$$(1290, 950) \quad , \quad (820, 875) \quad , \quad (220, 570) \quad , \quad (170, 330) \tag{1.1}$$

Note that you need to change the `radius` parameter. Does it work? Any ideas on how to set the parameter automatically?

## Using the SIFT code from vlfeat

To speed things up a bit, we will use the SIFT descriptor from the vlfeat toolbox in the next few exercises. It is written in C, so it is much more efficient than your Matlab implementation. Use

```
[coords, descriptors] = extractSIFT(img);
```

to compute positions and descriptors for the SIFT features in an image. (Make sure that the sift folder is in your path. If you still have problems running sift ask the lab assistant for help.)

**Ex 1.14** To prepare for the next exercise, try to work out how to use the built-in function `matchFeatures`.[1] To match descriptors using the Lowe criterion with threshold 0.7, add the following options:

```
corrs = matchFeatures(descs_1, descs_2, 'MatchThreshold', 100, 'MaxRatio', 0.7);
```

**Ex 1.15** In `church_data.mat` there is a collection of stored feature points, `feature_collection`. This is a struct with a $128 \times N$-array `feature_collection.descriptors` containing descriptors and a $1 \times N$ array of labels

```
feature_collection.labels
```

indicating what church the feature was collected from. The link between labels and church names is given by

```
feature_collection.names
```

Make a function

```
[label, name] = classify_church(image, feature_collection)
```

that tries to classify a new image by computing feature points for the new image, matching them to the features in the list and letting each match *vote* for the correct church.

**Ex 1.16** Try classifying all ten provided church images in `church_test`. How many do you get right? The correct labels are stored in `manual_labels.mat`.

### Report

Make a zip file with your code for this lab. Also include the plot from Ex 1.8. Upload it on Canvas and write in the message what percentages you got on the digit and church experiments.

[1] Note that Matlab and vlfeat does not agree on whether to store the descriptors as rows or columns.