Sehbau: Overview

An overview of the building blocks for creating a visual system is given. It explains the architecture for feature extraction and the presently available descriptors. The individual programs are introduced and it is announced what other processes are to be released.

Repository `https://github.com/Sehbau`

The system consists of a number of programs that carry out different processes. We firstly explain some basic parameters of the architecture and the descriptors developed so far (Section **??**). Then we introduce the core processes, namely descriptor extraction and their matching (Section **??**). A learning procedure is in the making (Section **??**). And some utility processes will be available at some point in time (Section **??**). Most of this is already implemented in *C*, but it will take some time to make it available as individual programs with optional arguments.

This is a huge framework. It might fail with unusual images. If you find one, inform me on `info@sehbau.com`. It should take sizes up to [3000 x 4000], such as a cell phone photo, but have not tried larger images yet. The program output (stdout) should terminate with the expression `EndOfProgram`. It might also fail if you choose unreasonable parameter values, as I have not included reasonability checks everywhere.

I provide utility scripts in Matlab, as that code is easier to read and maintain than Python (less parentheses and dots). For variable notation see also my Computer Vision overview:
`https://www.researchgate.net/publication/336460083`

1 Architecture and Descriptors

For an image, a pyramid of `nLev` levels is generated, ie. `nLev=5` for a 256 x 256 pixels. Each level is processed by a hierarchical segmentation, whose tree `depth` is typically set to 3 for images up to ca. 400 x 500 pixels; for larger images `depth=4` can be useful. This returns an abundance of features, more than in any previous engineering approach, yet still faster due to the use of simplest techniques.

The features have been abstracted by many types of descriptors, four of which have been translated to *C* so far:

- Contour (**`cnt`**): ridge (**`rdg`**), river (**`riv`**) and edge (**`edg`**) contours obtained from the intensity topology.
- Radial shape (**`rsg`**): parameterization based on the radial signature of a region (connected component).
- Arc segment (**`arc`**): curved boundary segments obtained from partitioning the region boundaries.
- Straighter segment (**`str`**): straighter boundary segments obtained from partitioning the region boundaries.

Many more descriptors have been tested already in Matlab. Their translation to *C* is in progress.

2 Core Processes

The core processes are:

- **`dscx`** carries out feature extraction and description and outputs various types of information.
  `https://github.com/Sehbau/DescExtraction`

- **`mvec`** [beta] matching descriptor vectors. Can be applied to any structure: shape, object or scene. Beta version. More options to be included.
  `https://github.com/Sehbau/MtchVec`

- **`motv`** [planned] the motion between two frames. This is essentially the same as **`mvec`** but will output in addition the motion vectors between nearest neighbors. That will allow to estimate motion.

- **`knnv`** [planned] nearest-neighbor search of structures using a coarse-to-fine strategy to accelerate the retrieval process, starting with a matching of the top (pyramid) layers and then subselecting and progressing toward finer levels. Early experiments have shown that this strategy not only speeds up the search, but also improves the sorting.

A matching between histograms is not foreseen so far, as for that there exist already statistical packages, ie. `sklearn` in Python.

3 Learning

A learning process will be provided that determines category-characteristic descriptors by individual matching of vectors.

- **`kksearch`** [planned]: searches for nearest neighbors across images of one category to find candidate descriptors, based on **`mvec`** essentially.

- **`kkgrup`** [planned]: refines the search and selects final set of category-characteristic descriptors.

- **`kkmtc`** [planned]: matches the category-characteristic descriptors against a new image and outputs the degree of dissimilarity per category-characteristic descriptor.

4 Utility Processes

The following utility processes will enable a fast search process.

- **`focs`** [planned]: selecting descriptors from a region (focus). This will subselect descriptors from a vector file. This will enable to perform a rapid local analysis, ie. object search.

- **`sgrRgb`** [planned]: segregates an RGB patch for a given target color, resulting in a black-white image with region boundaries that are more precise than with the gray-scale information as used in program **`dscx`**.