Sehbau: Descriptor Extraction

The use of the program for descriptor extraction is explained, called **dscx**. It extracts the features and describes them geometrically by a few parameters. It generates a number of output files: bounding boxes for the regions, the keypoints of contour segments and the descriptor vectors and their histograms.
See also: https://www.researchgate.net/publication/360033329

The program generates features for an image pyramid, each level of which is segmented separately. The pyramid is generated using downsampling with integer factor equal 2. The segmentation occurs with a tree of depth equal 3. The use of the program is explained first and what types of options are available (Section 1). Then we explain the type of output that is generated (Section 2).

1 Program Use

The program **dscx** takes two input arguments, the image path and the output path:

```
> dscx pathImg pathOutFile
```

The input image can be jpg or png. The output path must include a slash, because the program checks for that. Here is an example in which output file name `img1` is chosen to be the same as the image name (for convenience):

```
> dscx Imgs/img1.jpg  Desc/img1
```

This will write the following files into directory **Desc**:

- **img1.Bbox** bounding boxes of regions (connected components)
- **img1.CntEpt** endpoints of ridge, river and edge segments
- **img1.dsc** descriptor vectors (attributes)
- **img1.hst** descriptor histograms

You can load the first two files into Matlab as demonstrated in directory UtilMb, see script `exampleLoadBbox.m` for explanations.

1.1 Options

The following (long) options are available, to be specified with a double dash '–' (it does not show with this font!). The first set concerns 'architecture' parameters, the second set contour parameters, followed by region parameters.

The parameter names always start with a lowercase letter and then uses uppercase letters as in the Java notation.

1.1.1 Architecture:

–depth: depth of the segmentation process. Default `depth=3`. For `depth=1` no tree is grown: this corresponds to global thresholding only. `depth=4` can be useful for large images, e.g. larger than 1000 pixel for one image side.

–nLev: number of levels of the pyramid. Default is automatically calculated with top level not smaller than 16 pixels for one map side. For example for a 256x256 image, a five-level pyramid is generated: 256, 128, 64, 32, 16, whereby 256 is the original image resolution.

1.1.2 Contours:

–cntMinCtr: contrast threshold for contours. Default `=0.05`. This is a relative threshold to be set $\in [0.0..1.0]$. It is relative to the largest difference found in the range map for the gray-scale intensity image.

The following two parameters - starting with **skl** - modify the output of the contour selection, the skeleton. This concerns only the vectors - the histograms are generated with all contours without subselection.

–sklMinSpc: minimum spacing. Default `=0.05`. This is as proportion of the image side length $\in [0.0..1.0]$.

–sklMinLen: minimum length. Default `=0.05`. This as well is as proportion of the image side length $\in [0.0..1.0]$.

1.1.3 Regions:

–regMinPixNode: minimum number of pixels for a region to be segregated by the thresholding mechanism. This will affect the region count from the 2nd segmentation map on. It will not affect the 1st segmentation map, as that is the input to the 2nd for which segregation starts. Default equal 6. With larger values, processing occurs more rapidly, but may also skip tiny, low contrast regions.

–rsgMinPix: minimum number of boundary pixels for a radial region descriptor. The number is set for the original image resolution. For higher levels of the pyramid, a correspondingly lower number is used, namely `rsgMinPix-level`. E.g. for a value of 10, the higher pyramidal levels utilize values `9, 8, 7,...`.

2 Output Files

An example for loading the bounding boxes and contour endpoints to Matlab is given in the script `exampleLoadBbox.m` in directory UtilMb. Below follow a few explanations.

2.1 Bounding Boxes (.Bbox)

The bounding boxes for regions, in file **name.Bbox**, are organized as follows. The first two integers hold the number of levels `nLev` and segmentation depth `depth` used for the run. The following numbers hold the region count for each segmentation map, `nBbox`, saved looping levels as the outer loop and looping depth as the inner loop. The example below shows that for two levels and depth equal 3 (zero-indexing).

```
nLev
depth
nBbox_Lev0_Depth0
nBbox_Lev0_Depth1
nBbox_Lev0_Depth2
nBbox_Lev1_Depth0
nBbox_Lev1_Depth1
nBbox_Lev1_Depth2
```

Then the bounding boxes follow. They are organized analogously to the above inner/outer loop: first all bounding boxes of [lev=0,depth=0], then those of [lev=0,depth=1], etc. A bounding box contains 6 parameters.

```
top, bottom, left, right, area, border       (for l
```

```
top, bottom, left, right, area, border
...
top, bottom, left, right, area, border       (for lev=1, depth=2)
```

The bounding boxes are absolute coordinates that correspond to the map size of the pyramidal level.

2.2 Contour Endpoints (.CntEpt)

The stored points for the contour segments are the two endpoints as well as their midpoint. The points are written per level, per contour type and per point type.

The first value holds the number of levels. Then each level of the pyramid is written separately with firstly the points of the ridge contours, then those of the river contours and eventually those of the edge contours. The points are written blockwise (and not rowwise as in case of the bounding boxes). The first value holds the number of descriptors. Then follow first all coordinates of the first endpoint (for that level); then all coordinates of the second endpoint; followed by all coordinates for the midpoint. The coordinates `coords` are saved as row/column pairs, per point.

```
nLev
nRdg  (# of ridge contours for lev=0)
[ridge coords of 1st endpoint for lev=0]
[ridge coords of 2nd endpoint for lev=0]
[ridge coords of midpoint     for lev=0]
nRiv  (# of river contours for lev=0)
[river coords of 1st endpoint for lev=0]
[river coords of 2nd endpoint for lev=0]
[river coords of midpoint     for lev=0]
nEdg  (# of edge contours for lev=0)
[edge  coords of 1st endpoint for lev=0]
[edge  coords of 2nd endpoint for lev=0]
[edge  coords of midpoint     for lev=0]
nRdg  (# of ridge contours for lev=1)
[ridge coords of 1st endpoint for lev=1]
[ridge coords of 2nd endpoint for lev=1]
[ridge coords of midpoint     for lev=1]
...
```