Sehbau: Matching Vectors

The use of the program for vector matching is explained, called **mvec**. It matches the descriptor vectors of two images (**.vec** files) as generated by the descriptor extraction program (**dscx**), or two focus files (**.vef**) as generated by **focx**. It can be used for any two structures expressed by the vectors, be it a scene, an object, a shape or a texture.

Read https://www.researchgate.net/publication/360033329
Repository https://github.com/Sehbau/MtchVec
PREVIOUS descriptor extraction with **dscx**, https://github.com/Sehbau/DescExtraction
focus extraction with **focx**, https://github.com/Sehbau/FocExtr

The directories in the repository contain the following:

- **/Desc** sample vector and focus files (output of **dscx**/**focx**)
- **/Imgs** sampe images
- **/UtilMb** Matlab scripts to read the output data files

The program **mvec** matches the two list of descriptors using pairwise distance (similarity) calculation and choosing the nearest neighbor. As we usually have unequal list length, the results are different for matching the first image against the second image, and with order reversed (2nd versus 1st). The program outputs both matching measurements. The user can then combine the values ad libitum.

For each descriptor and each (pyramid) level, a dissimilarity and a similarity value is measured, abbreviated `dis` and `sim` (or sometimes also abbreviated as `dist` and `simi`, resp.):

- **dis** returns the Euclidean distance.
- **sim** returns the number of matches that are below a fixed threshold value, set with option **[dsc]TolMtc** or **tolMtc** for all descriptor types.

The program can be applied to two vector files (**vec**) as outputted by **dscx**, or to two vector focus files (**vef**) as outputted by **focx**. The combination of a **vec** and **vef** file is not not possible yet. The vector files are required to have the same number of levels (pyramid height), otherwise the program returns no results (more flexibility to be included in the future).

The program comes in two variants:

- **mvec1** : matches one pair of images (or focii): one versus one. The Matlab script `runMvec1.m` demonstrates how to deploy the program and read its output. This program is useful for exploring parameter variations.
- **mvecL** : matches a list of images (or focii): one versus multiple. Its use is demonstrated in Matlab script `runMvecL.m`. It outputs only integrated measurements (and not the variety as for **mvec1**). It can be used to match at large scale.

1 Program Use

We firstly explain the use of **mvec1**, the matching of two vector files. Their file paths are given as arguments. For example for two image files (**.vec** from **dscx**) we write:

```
mvec1 Desc/img1.vec Desc/img2.vec
```

Or for two focus file (**.vef** from **focxv**):

```
mvec1 Desc/foc1.vef Desc/foc2.vef
```

The vector files are required to have the same number of levels, ie. generated by similarly sized images, otherwise it returns no results. The output is written to **stdout** and will be further explained in Section 2.

For the use of the program **mvecL** we specify a file path and a text file containing the file paths for a list of files to be matched with:

```
mvecL Desc/img1.vec FinasImg.txt
```

This will write the metric measurements into a file named **MesRep.txt** in the same directory. You can specify a different file name by giving a third argument, ie.:

```
mvecL Desc/img1.vec FinasImg.txt MesMtcImg.txt
```

By default, the program matches all four descriptor types for the entire pyramid. The options allow to subselect descriptors and levels, as well as to set attribute weight values.

1.1 Options

The first list of options set a parameter value to the same value for all descriptor types (Section 1.1.1), which can be unspecific in some cases, but which is convenient for coarse tuning. The second list of options allows to adjust parameters of individual descriptor types for fine tuning (Section 1.1.2).

1.1.1 General (All Descriptor Types):

The following options set values for all descriptor types simultaneously and are useful for coarse tuning. Default values are given in approximate values only, as they are often individual to the descriptor type:

--tolMtc [similarity metric]: sets matching tolerance to a fixed value, arcoss all pyramidal levels (and across all descriptor types). This is for the similarity metric only (section `Simi` in output). Default: ca. 0.05.

--wgtRGB: sets the weight value for the RGB difference. Set this parameter to zero if chromatic information is irrelevant, for example when places are to be recognized at either day or night. Keep in mind that three difference values are taken (R,G,B) and that this weight parameter therefore has more influence than the others. Default: ca. 1.0.

--wgtPos: sets the weight value for the position parameter for each descriptor type. A value of 0 turns off the influence. Default equal ca. 1.0.

1.1.2 Individual (Per Descriptor Type):

--cntTolMtc: tolerance for contour matches, for the similarity metric. Fixed value for all levels, but will try to provide something more flexible. Default: complicated.

--rsgTolMtc: tolerance for radial descriptor matches. Analogous to option **cntTolMtc**.

--arcTolMtc: tolerance for arc segment matches (see **cntTolMtc**).

--strTolMtc: tolerance for straighter segment matches (see **cntTolMtc**).

More in progress.

1.1.3 Utility:

--prms: displays the parameter values used.

2 Output

*2.1 Program **mvec1***

The results of **mvec1** are written to **stdout** and look as follows (breaks across page). If you are interested in only the total, then use the last two values of section `---- img ----` and add them.

```
-----  MatchResults  -----
nLevRes 5
-----  12  -----
Dist:
SklDis12 13.175 39.664 45.217 41.913 13.573
RsgDis12 109.845 27.919 12.824 8.965 3.398
ArcGstDis12 114.219 46.689 53.185 21.986 8.605
StrGstDis12 38.233 14.913 32.930 21.659 11.041
ArcDis12 0.000 0.000 0.000 0.000 0.000
StrDis12 0.000 0.000 0.000 0.000 0.000
Simi:
SklSim12 0.000 16.000 108.000 201.000 30.000
RsgSim12 68.000 42.000 10.000 1.000 0.000
ArcGstSim12 2.000 2.000 19.000 2.000 2.000
StrGstSim12 0.000 0.000 35.000 6.000 1.000
ArcSim12 0.000 0.000 0.000 0.000 0.000
StrSim12 0.000 0.000 0.000 0.000 0.000
-----  21  -----
Dist:
SklDis21 53.248 83.856 93.762 80.252 23.530
RsgDis21 223.865 77.948 22.488 7.217 3.611
ArcGstDis21 169.397 54.075 42.516 8.441 2.109
StrGstDis21 99.205 21.655 37.785 17.365 2.734
ArcDis21 0.000 0.000 0.000 0.000 0.000
StrDis21 0.000 0.000 0.000 0.000 0.000
Simi:
SklSim21 0.000 16.000 111.000 236.000 30.000
RsgSim21 68.000 42.000 11.000 1.000 0.000
ArcGstSim21 2.000 2.000 19.000 2.000 2.000
StrGstSim21 0.000 0.000 35.000 6.000 1.000
ArcSim21 0.000 0.000 0.000 0.000 0.000
StrSim21 0.000 0.000 0.000 0.000 0.000
-----  Ndsc  -----
Ndsc1:
SklNdsc1   40  132  155  140   35
RsgNdsc1  496  109   39   23    6
ArcGstNdsc1  368  132  152   50   17
StrGstNdsc1  124   34  102   57   19
ArcNdsc1    0    0    0    0    0
StrNdsc1    0    0    0    0    0
Ndsc2:
SklNdsc2  129  256  304  252   58
RsgNdsc2  963  269   65   20    6
ArcGstNdsc2  547  162  135   22    6
StrGstNdsc2  294   50  121   47    7
ArcNdsc2    0    0    0    0    0
StrNdsc2    0    0    0    0    0
-----  img  -----
disV12 727138240.000
disV21 5543590912.000
```

The output is organized in two measurement sections, `---- 12 ----` and `---- 21 ----`, each of which lists the dissimilarity (`Dist`) and the similarity (`Simi`) values for each descriptor type (`skl`, `rsg`, `arc` and `str`) and each pyramid level. For example for the `12` matching:

```
-----  12  -----
Dist:
SklDis12 13.175 39.664 45.217 41.913 13.573
...
Simi:
SklSim12 0.000 16.000 108.000 201.000 30.000
...
```

the line `SklDis12...` lists the dissimilarity values for the skeleton (selected contours) for each level (5 in this case); the line `SklSim12...` lists the similarity values for each level and are integers as they represent the count of succesful matches (given fixed threshold **cntTolMtc**).

The section `---- Ndsc ----` lists the descriptor count for each level, which gives you the chance to scale as you wish. `Ndsc1` is the descriptor count for the first image, `Ndsc2` the count for the second image.

The very last section `---- img ----` contains the (total) dissimilarity value for 1 versus 2 (`disV12`), and 2 versus 1 (`disV21`), whereby values are cumulatively multiplied across levels. The combination of both values achieved better results in visual place recognition.

So far, I've mainly used the dissimilarity measure. But I've observed that the similarity count is occasionally significantly better. Perhaps a combination of the two could improve results significantly.

The syllables summarized:
- `Skl`: (selected) contours
- `Rsg`: radial descriptors
- `ArcGst`: subselected arc segments
- `StrGst`: subselected straighter segments
- `Arc`: full set of arcs, by default *not* matched
- `Str`: full set of arcs, by default *not* matched

The function scripts in directory **/UtilMb** give an example how to read the output, called from the test script `runMvec1.m`.

*2.2 Program **mvecL***

The results of matching multiple files is written to file as text, and for the moment only a dissimilarity and similarity value are returned. The text file contains four columns, where the first corresponds to dissimilarity and the second to similarity. The third and fourth column are empty (zero) for the moment. Matlab script `LoadMtchMes.m` shows how to load the file.