

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**). It can be used for any two structures expressed by the vectors.

Read <https://www.researchgate.net/publication/360033329>
Repository <https://github.com/Sehbau/MtchVec>
PREVIOUS [descriptor extraction with **dscx**](https://github.com/Sehbau/DescExtraction), <https://github.com/Sehbau/DescExtraction>

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**. For dissimilarity, the Euclidean distance is taken; for similarity, the number of matches that are below a fixed threshold value is taken, set with option **[dsc]TolMtc**.

1 Program Use

Two vector files (**.vec**) are given as arguments:

```
> mvec Desc/img1.vec Desc/img2.vec
```

The vector files are required to have the same number of levels, ie. generated by similarly sized images. More flexibility will be added later.

The output is written to **stdout**.

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

--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**).

--wgtRGB: sets the weight value for the RGB difference for each descriptor type. Set to 0 if chromatic information is irrelevant, if for example 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.

--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.

More in progress.

1.1.1 Utility:

--prms: displays the parameter values used.

2 Output

The program writes the dissimilarity and similarity values to **stdout** and looks as shown below (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:
Sk1Dis12 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:
Sk1Sim12 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:
Sk1Dis21 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:
Sk1Sim21 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:
Sk1Ndsc1 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:
Sk1Ndsc2 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:
Sk1Dis12 13.175 39.664 45.217 41.913 13.573
...
Simi:
Sk1Sim12 0.000 16.000 108.000 201.000 30.000
...
```

the line Sk1Dis12... lists the dissimilarity values for the skeleton (selected contours) for each level (5 in this case); the line Sk1Sim12... 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

An example to read a measurement section in Matlab would be as follows, with typ being 'Dis' or 'Sim':

```
ixSk1      = strfind(OutStr, ['Skl' typ '12']) + 9;
ixRsg      = strfind(OutStr, ['Rsg' typ '12']) + 9;
ixArcGst   = strfind(OutStr, ['ArcGst' typ '12']) + 12;
ixStrGst    = strfind(OutStr, ['StrGst' typ '12']) + 12;

Mes.Skl12   = sscanf(OutStr(ixSk1:end), '%f', nLev);
Mes.Rsg12   = sscanf(OutStr(ixRsg:end), '%f', nLev);
Mes.Arc12   = sscanf(OutStr(ixArcGst:end), '%f', nLev);
Mes.Str12   = sscanf(OutStr(ixStrGst:end), '%f', nLev);

ixSk1      = strfind(OutStr, ['Skl' typ '21']) + 9;
ixRsg      = strfind(OutStr, ['Rsg' typ '21']) + 9;
ixArcGst   = strfind(OutStr, ['ArcGst' typ '21']) + 12;
ixStrGst    = strfind(OutStr, ['StrGst' typ '21']) + 12;

Mes.Skl21   = sscanf(OutStr(ixSk1:end), '%f', nLev);
Mes.Rsg21   = sscanf(OutStr(ixRsg:end), '%f', nLev);
Mes.Arc21   = sscanf(OutStr(ixArcGst:end), '%f', nLev);
Mes.Str21   = sscanf(OutStr(ixStrGst:end), '%f', nLev);
```