Subject: Identification of points after transformations
Posted by Helder Marchetto on Tue, 24 Jul 2012 11:01:35 GMT

View Forum Message <> Reply to Message

Hi,

*** sorry for the long post... ***

I'm tangled up with a problem that is at first sight not that easy.
I have two sets of images where I have identified n-points in the first image and m-points in the first image. The two sets of points do not have a common order within the array where they are stored.
Now the points in the second set are rotated and scaled (magnified) with respect to the first set around an arbitrary pivoting point.
For complicated reasons I cannot explain here, I cannot use FFTs or other direct image alignment procedures. I have to work with "bare" points.
So here is what I was thinking doing. First I select a random starting point (for instance the first element of the array) in the first array and then I sort the array based on the distances with respect to this reference point.
In IDL terms it would look something like this:
nPoints = 10
ImageSize = 512
Arr1=RANDOMU(SEED,nPoints,2) * FLOAT(ImageSize)
RefPoint = Arr1[0,*]
RefPntArr1 = Arr1
RefPntArr1[*,0] = Arr1[*,0]-RefPoint[0,0]
RefPntArr1[*,1] = Arr1[*,1]-RefPoint[0,1]
DistArr = SQRT(TOTAL((RefPntArr1)^2,2))
SortingArr = SORT(DistArr)
ReSortArr1 = Arr1[SortingArr ,*]
PRINT, Arr1
PRINT, ReSortArr1

Now I that I have the sorted array from the first image, I have to define relative distances and relative angles. To do that I select a second reference point, for instance the last element of the array, and reference all distances to the distance between the first and last element of the array. The same goes for the angles. That would look something like this:
RefDistArr1 = DistArr[SortingArr]
RefDistArr1 = RefDistArr1/RefDistArr1[-1] ;valid only starting from IDL 8.xx

Similarly for the angles I would calculate:
RefAngleArr1 = ATAN(ReSortArr1[*,1]-ReSortArr1[0,1],ReSortArr1[*,0]-ReSortA rr1[0,0])
RefAngleArr1 = RefAngleArr1-ATAN(ReSortArr1[-1,1]-ReSortArr1[0,1],ReSortArr 1[-1,0]-ReSortArr1[0,0])

Now these two quantities RefDistArr1 and RefAngleArr1 will be 'conserved' after rotation.
Therefore I need to find the best fit of the points from image two.

What it will probably come down to is the fact that some points will be missing in one of the two images and therefore I cannot do a simple comparison of the arrays. To make things worst, after rotation and scaling there will be most likely an error that will mess up a bit more the coordinates.

Do you know of clean ways to compare such sets of points? I will probably have to introduce some sort of error measurement (square difference?) and take care of the missing elements in some way (but I don't know yet in what way).

So, the question are:
General: does anybody have to solve similar problems and has good tips for that?
Programming/statistics specific: how should I compare an array that looks like this: [1.2, 3.6, 9.4, 12.4] with other arrays that look like: [1.1, 3.7, 12.3] or [1.1, 2.1, 8.9, 11.9]. Where the first should turn out to be a match and the second not.

Any tips are welcome.

Cheers,
Helder