3D Registration Toolbox Documentation

1 – 3D Camera acquisition

This part of the software allows the user to create synthetic 3D views from an object represented by a cloud of points. Note that the data used in the examples have been acquired from INRIA database (http://www-c.inria.fr/gamma/download/download.php). The process involves 3 main functions:

a) **paraview2matlab.m:** (returns the points and the triangles that compose the object)
b) **normals_extraction:** (computes the normals of each point of the object)
c) **scan_synthetic:** Simulation of a real camera. Acquisition of several partial views. This function will provide n partial views of the object in matlab (.m) and paraview (.g) format.

[P,tri]=paraview2matlab(img_name);

img_name: name and path of the object to be acquired (including extension).
P: Points that compose the object.
tri: Triangles that compose the object.

[P,N,tri]=normals_extraction(P,tri,img_name);

P: Points that compose the object (obtained by paraview2matlab function)
Tri: Triangles that compose the object (obtained by paraview2matlab function)
N: Normal of each point that composes the object.
img_name: name of the object (without extension).

scan_synthetic(P,N,tri,60,6,view_name);

P: Points that compose the object (obtained by paraview2matlab function)
Tri: Triangles that compose the object (obtained by paraview2matlab function)
N: Normal of each point that composes the object.
view_name: path and root name of the view (without extension).

Example:

The next code will run an example using a Beethoven object obtained in INRIA database. The used object is provided with the Toolbox. In Matlab command window write: "sample_acquisition ()", which automatically will run the functions:

```
[P,tri]=paraview2matlab('img/beethoven.g');
[P,N,tri]=normals_extraction(P,tri,'beethoven');
scan_synthetic(P,N,tri,60,6,0.60.0.60,'img/view');
```

Note that the partial views will be saved in the directory 'img/'.

2 – Pairwise Registration

This part of the software registers views one-to-one. The most important functions used in the pairwise registration process are:

pairwise.m: register of views one-to-one
metode_park2.m: Computes the ICP point-to-plane method based on the improved version of Chen's method proposed by Park.
normal_space_sampling.m: discards the irrelevant points in order to speed the process. The discarded points are mainly the ones of the planar areas since they do not provide useful information for the success of the process.
park2.m: Computes the rotation and translation between two registered views.

pairwise(view_name, initial_view, extension, gaussian_noise)

view_name: path and root name of the partial views
initial_view: view desired to start the pairwise
extension: extension of the partial views which contains the needed information
Gaussian_noise: sigma value. This allows simulating a real camera in a more realistic way.

Example:

The next code will run an example using partial views obtained in the acquisition step from a Beethoven object (INRIA database). Note that the process in the example starts in view 1 and the introduced error is 0.

pairwise('img/final',1,'.mat',0)

The function returns a file "img/final_soroll_0_PairWise.mat" which contains all the information about the registration including rotation and translation matrices between views.

In order to visualize the results obtained by pairwise registration use the function "visualize(pairwise_file) ":

visualize('img/final_noise_0_Pairwise.mat')

To run an automatic example of the registration which includes a visualization of the results use the function: "sample_pairwise()"

3 – Multiview Registration

This part of the software registers multiples views. The methods of Matabosch and Sharp are methods based on cycle detection and the Pulli and Chen's ones are based on metaview. More information about this methods can be fount at ??

3.1- Method of Matabosch

Main function:

cycles_registration_matabosch(img_name, extension, initial, method, noise, pairwise)

img_name: path and root name of the views and files to be used .
extension: usually '.mat'
initial: starting view
method: 1: robust Matabosh (more than 24 hours)
2: fast Matabosch (less than 20 minutes)
gausian_noise: Gaussian noise (sigma)
pairwise: 0: No compute pairwise. Use an already existing file (img_name).
1: Compute pairwise.

Example:

The next code will run an example of multiple view registration using Matabosch technique and using an already existing pairwise file (only multiview will be computed). Note that the process in the example starts in view 1 and the introduced gaussian noise is 0.

cycles_registration_matabosch('img/final', '.mat', 1, 2, 0.0025, 0)

Note that the result of this multiview registration using Matabosch method can be found in: "img/finalnoise0.0025Matabosch_fast.mat"

To visualize the results: visualize('img/finalnoise0.0025Matabosch_fast.mat')

To run an automatic example of the registration which includes a visualization of the results use the function: "sample_matabosch()"

3.2- Method of Sharp

Main function:

cycles_registration_sharp(img_name, initial, gausian_noise, pairwise)

img_name: path and root name of the views and files to be used .
initial: starting view
gausian_noise: Gaussian noise (sigma)

pairwise: 0: No compute pairwise. Use an already existing file (img_name).
1: Compute pairwise.

Example:

The next code will run an example using an already existing pairwise file (only multiview will be computed). Note that the process in the example starts in view 1 and the introduced gaussian noise is 0.0025.

cycles_registration_sharp('img/final', 1 , 0.005, 0);

Note that the result of this multiview registration using Sharp method can be found in: "img/finalnoise0.005Sharp.mat"

To visualize the results: visualize('img/finalnoise0.0025Sharp.mat')

To run an automatic example of the registration which includes a visualization of the results use the function: "sample_sharp()"

3.3- Method of Chen

Main function:

chen(img_name, extension, initial, max_views, gausian_noise)

img_name: path and root name of the views and files to be used .
extension: usually '.mat'
initial: starting view
max_views: number maximum of views to be registered
gausian_noise: Gaussian noise (sigma)

Example:

The next code will run an example using Chen's approach. Note that the process in the example starts in view 1 and the introduced gaussian noise is 0.0025.

chen('img/final', ',mat', 1, 65, 0.0025)

Note that the result of this multiview registration using Chen method can be found in: "img/finalnoise0.0025Chen.mat"

To visualize the results: visualize('img/finalnoise0Chen.mat')

To run an automatic example of the registration of 20 views which includes a visualization of the results use the function: "sample_chen()"

3.4- Method of Pulli

Main function:

Pulli(img_name, gaussian_noise, initial, max_views, error)

img_name: path and root name of the views and files to be used .
gausian_noise: Gaussian noise (sigma)
initial: starting view
max_views: number maximum of views to be registered
error: threshold of the maximum error allowed between neighbour views registration (0.5)

Example:

The next code will run an example using Pulli's approach. Note that the process in the example starts in view 1 and the introduced gaussian noise is 0.0025.

pulli('img/final', 0, 1, 65, 0.5)

Note that the result of this multiview registration using Pulli's method can be found in: "img/finalnoise0Pulli.mat"

To visualize the results: visualize('img/finalnoise0Pulli.mat')

To run an automatic example of the registration of 20 views which includes a visualization of the results use the function: "sample_pulli()"

4.- Validation results

4.1- 3D error

This part of the software allows the user to verify the registration results, comparing the real 3D points with the obtained ones. The process involves 3 main steps:

4.1.1- To obtain the initial model (points composing the initial object in the same coordinate system of the results)

In order to obtain the initial model execute the function:

obtainModel3D(path, image, extension);

path: path were the image is found
image: name of the image/object to be treated
extension: extension of the image

Example:

obtainModel3D('/img','beethoven', '.g');

To run an automatic example: "sample_obtainModel3D();"

4.1.2- To obtain the registered model (points composing the registered object)

transforma3D(image,method);

image: name of the image/object to be treated
method: 1: Matabosch
2: Sharp
3: Chen
4: Pulli

Example:

transforma3D ('results/finalnoise0Chen.mat', 3);

To run an automatic example: "sample_transforma3D();"

4.1.3- Comparison between initial and obtained model

calcul_3D_diff(initial_object, registered_object);

initial_object: clowd of points obtained throw process 4.1

registered_object: clowd of points obtained throw process 4.2

Example:

calcul_3D_diff('model3D/img/beethoven_Model3D_60_6.mat',
'registre3D/results/finalnoise0Chen_Registered_Model3D.mat');

To run an automatic example: " sample_calcul_3D_diff ();"

4.2.- Registration error (Comparing rotation and translation error between registration and real motion)

[error_angle,error_axis] = calcul_error (inicial_image, registered_image, n_views, method);

inicial_image: initial real movement ("final.mat")
registered_image: result obtained from the registration process
n_views: number of views
method: (to treat different type of matrix formats)
0: Pulli and Chen
1: Sharp and Matabosch

Example:

[error_angle,error_axis] = calcul_error ('initial_model/final.mat',
'registered_model/finalnoise0Chen.mat', 40, 0);

To run an automatic example: " sample_calcul_error ();"