OptFlow:
Matlab-Tools for the calculation of optic flow from geometry and trajectory

Jens Peter Lindemann

May 25, 2009

1 Overview

Given the geometry of an environment (3D-model) and a trajectory (3D-positions and gaze directions), calculate for each step of the trajectory the optic flow sampled at given positions on the surface of a unit sphere.

1.1 Algorithm

1.1.1 Input

- 3D-model of the environment defined by
  - a list of 3D-coordinates (vertices)
  - a list of polygons: For each polygon a list of references to vertices
- the trajectory defined by
  - 3D-coordinates of the current position
  - three angles for the current gaze direction: yaw, pitch, roll
- angular coordinates of the sampling positions on the unit sphere for the calculation of the optic flow

1.2 Output

The optic flow sampled at the given sampling points for each step in the trajectory.

1.3 Calculation path

1. translation and rotation of the 3D-model according to the trajectory
2. calculation of environmental distance for each sampling point:
   - intersection of all polygon planes with sampling ray
   - test of intersections: Intersection point inside of the polygon?

- distance for this sampling point: Minimum of all valid intersections

3. calculation of the translational and rotational velocities by differentiation of the trajectory
4. calculation of the optic flow using the Koenderink equation

2 Data structures

2.1 3D-Model

Minimum requirement for the model is the enumeration of the polygons forming the enviroment. Thus the minimal representation would be two lists: A list of vertices and a list of polygons formed by these.

The most general approach to represent a 3D model is to represent a scene graph, a standard data structure used in 3D modelling. However, this would make the handling of models very complex.

We start with an intermediate level of representation: A model is constructed from a list of objects. These objects can be single polygons or complex structures gouped as the user needs them: Objects are the level of abstraction that can be referenced for visualisation purposes.

Example:

The model consists of a cylindrical arena and three cylindrical objects. The wall and the objects are separately represented as objects. This allows to draw the flow originating from the wall in one color, the flow originating from object movement in a different color.

2.1.1 Matlab data structure

An object is represented by a Matlab struct:

obj.vert	a $N \times 3$ matrix containing the N vertices of the object.
obj.tri	a $M \times 3$ matrix enumerating M triangles via indices of obj.vert
obj.poly	a cell of index vectors for the polygons
obj.label	a string label for object reference

Why represent triangles and polygons? For several reasons it is more conventient for the flow caluclation to work with triangles. Every polgon can be represented by a set of triangles. But for display purposes it may be confusing to draw triangles where the user specified a square or a hexagon. Thus we store the original polygon for displaying alongside the polygon split in triangles for calculation.

Object structures are collected in a Matlab cell array to represent a model.

Example:

```
model={struct('vert', [0,0,0; 1,0,0; 0,1,0; 1,1,0],...
              'tri', [1,2,3; 3,2,4],...
              'poly', {[1,2,3,4]},...
              'label', 'Hsquare'),...
       struct('vert', [0,0,0; 0,1,0; 0,0,1; 0,1,1],...
              'tri', [1,2,3; 3,2,4],...
              'poly', {[1,2,3,4]},...
              'label', 'Vsquare')}
```

2.2 Trajectory

Matlab matrix (double): $T \times 6$ for T trajectory steps.

Column order: x, y, z, yaw, pitch, roll.
x, y, z given in an righthanded cartesian coordinate system. Yaw pitch and roll describe the rightwise rotation around the axis in the folowing order:
yaw around the z- axis,
pitch around the rotated y- axis and
roll around the rotated x- axis.
All angles given in radian.

Example: An trayectory of [1, 1, 1, 0, 0, 0] would be the position [1, 1, 1] with viewing-direction along x- axis and direction of z- axis as top.

2.3 Sampling points

Matlab matrix (double): $S \times 2$ for S sampling points. The sampling points given in spheric coordinates, again all angles in radian:

Azimuth (Longitude) is the angle between x-axis and SamplePoint. Azimuth angle must be between 0 and $2 * \pi$.
$[\pi/2, 0] = [0, 1, 0]$ left,
[3*π/2, 0] = [0, -1, 0] right.

Zenith (Latitude) given in geographic coordinate system is the elevation to the x-y-plane. Zenith is between $-\pi/2$ and $\pi/2$
$[0, \pi/2] = [0, 0, 1]$ top,
$[0, -\pi/2] = [0, 0, -1]$ bottom.

3 Functions

All functions of the toolbox use prefix "OF".

3.1 Model generation functions

All functions dealing with models use prefix "OFMod"

3.1.1 OFModSphere

mod=OFModSphere(label, r, density) returns a sphere with radius r. Optional parameter density specifies density of the mesh.

The sphere forms a single object labeled by the given label.

3.1.2 OFModRing

mod=OFModRing(label, r, h, density) returns a circular ring made from rectangles: r gives the radius of the circle, h the height of the rectangles. Optional parameter density gives the number of rectangles.

The ring forms a single object labeled by the given label.

3.1.3 OFModDisc

mod=OFModDisc(label, r, density) returns a planar circular disc with radius r. Optional parameter density gives the number of vertices used to approximate the circle.

The disc forms a single polygon in the object labeled by the given label.

3.1.4 OFModCylinder

mod=OFModCylinder(label, r, h, density) returns a cylinder. See OFModRing for parameter description. The ring is closed by two Discs.

The cylinder forms a single object labeled by the given label.

3.1.5 OFModCube

mod=OFModCube(label, w, h, l) returns a cube with height h, width w and length l.

The cube forms a single object constructed from rectangles ane labeled by the given label.

3.1.6 OFModRectangle

mod=OFModRectangle(label,w,h) returns a rectangle width width w and height h.

3.1.7 OFModPolygon

mod=OFModPolygon(label, v1, v2, v3, ...) returns a polygon defined by any number of vertices.

The function tests for planarity of the polygon and gives a warning otherwise.

3.1.8 OFObjJoin

obj=OFObjJoin(label, o1, o2) returns an object constructed from the components of the two objects o1 and o2. The labels of both input objects are replaced by the new label given as a parameter.

3.1.9 OFModJoin

mod=OFModJoin(mod1, mod2) returns a model composed from two models mod1 and mod2. This is equivatlent to:

```
mod={mod1, mod2}
```

3.1.10 OFModTranslate

mod=OFModTranslate(mod, x, y, z) returns mod moved by translation vector (x,y,z).

3.1.11 OFModRotate

mod=OFModRotate(mod, axis, angle) returns the model rotated by angle given in radian around the axis given as a three dimensional vector.

3.1.12 Example

This small script should create an arena with a cylinder as obstacle slightly out of center:

```
arena=OFModTranslate(OFModRing('wall', 2000, 300), 0, 0, 150);
arena=OFModJoin(arena, OFModDisc('floor', 2000));
arena=OFModJoin(arena, OFModTranslate(OFModCylinder('obstacle', 100, 200, 0);
```

3.2 Optic flow calculation functions

All functions involved in optic flow calculation use prefix "OFCalc"

3.2.1 OFCalcDistances

dist=OFCalcDistances(mod, tra, sp) returns the distances to the environment at the sampling points for all trajectory steps. model is a cell of object structs, tra a $T \times 6$ matrix and sp a $S \times 2$ matrix. Return value dist is a $T \times S$ matrix with environment distances. This method is called by OFCalcOpticFlow and dosn't have to be called manually.

3.2.2 OFCalcOpticFlow

[hof, vof]=OFCalcOpticFow(mod, tra, sp) returns the optic flow with model being a cell with object structs, tra a $T \times 6$ matrix and sp a $S \times 2$ matrix. Return values are both matrices with $T \times S$ elements where hof(t,s) stored the horizontal and vof(t,s) the vertical component of the "optic flow-shifted sampling point" s at trajectory position t.

3.2.3 OFCalcPoE_PoC

[PoE, PoC]=OFCalcPoE_PoC(sp, hof, vof, t) returns the Points which are PoE or PoC, derived by the Optic Flow at position t. PoE and PoC are $n \times 2$ and $m \times 2$ matrices with spheric coordinates (same as sp) of the points.
CAUTION: This works only for SamplePoints generated by OFGenerateSp! The points con only be found, if their not on the border of the Optic Flow.

3.2.4 OFCalcVelocities

[hvel, vvel]=OFCalcVelocities(sp, hof, vof) returns the velocities of the optic flow calculated by the OFCalcOpticFlow method.

3.2.5 OFGenerateSp

sp=OFGenerateSp(left, right, top, bottom, density) generates a samplepoint matrix acording to user specifications:
left - the left border of the sp [pi - 0]
right - the right border of the sp [pi - 2*pi]
top - the top border of the sp [0 - $pi/2$]
bottom - the bottom border of sp [0 - $pi/2$]
density - the number of sp-rows and columns
Example:

- OFGenerateSp(*pi*, *pi*, 0, 0, 12) would make 244 samplepoints all at the front viewing direction (no sence...)
- OFGenerateSp(*pi*/2, 3**pi*/2, *pi*/4, -*pi*/4, 11) would make 221 samplepoints in an area 90 ° left to 90 ° right and 45 ° top to 45 ° bottom relative to front viewing direction

3.3 Visualisation functions

3.3.1 OFDrawMesh

OFDrawMesh(mod, show_tri) draws the mesh defined by mod in the current figure. If show_tri is "True", the triangles instead of the polygons are drawn.

3.3.2 OFDrawTra

OFDrawTra(mod, tra, mdist) draw the mesh of mod and the trajectory represented by a line and markers: mdist gives the distance between markers. Markers are dots on the trajectory and short line segments illustrating yaw and pitch angles.

3.3.3 OFDrawCylPlot

OFDrawCylPlot(sp, hof, vof, t, scale) plots the optic flow field from hof and vof (see OFCalcOpticFlow) at trajectory position t in an cylindric coordinate system. Scale scales the Flow Vectorsas follows:
1=no scale, 0.5= half length, 2= double length
Caution: Flow near the poles seems bigger than it is, due to projection from spheric to cylindric coordinate system.

3.3.4 OFDraw3dPlot

OFDraw3dPlot(tra, sp, hof, vof, t, scale) plots the optic flow field from hof and vof (see OFCalcOpticFlow) at trajectory position t on an unitsphere. Scale scales the Flow Vectors as follows:
1=no scale, 0.5= half length, 2= double length
If OFDrawTra has been ploted first, this function will insert the unitsphere with the Optic Flow into the plot at trajectory position t.

3.4 Subroutine's

OFSubroutine... are sub-functions called by several other functions. They do not have to be called manually.