Visual 3D registration with flexible kinematic prior

# Introduction

This document is a draft describing the current state of the project. It keeps tracks of the various option at hand and also detail the mathematics consideration on which numerical implementation will be based.

## Project overview

Our aim is to extract the n-deflector configuration as well as the surface topology of the arm from the image of an RGB-D camera even with partial occlusion.

For this study different model for the arm description will be assessed first then they’ll be used as prior for the implementation of 3D registration.

## Evaluation criterion

## Current development

For now, the work is being divided into two part, on one side the development of geometric/kinematic model suitable for real-time registration (Part 2) and on the other side the development of the registration software (Part 3.1 and 3.2). The two part once fully developed should come together make the final registration method (Part 3.3)

Thus far the equation governing 3 different model have been detailed:

* Rigid ball joint based skeleton
* Piecewise constant curvature
* PH cubic spline

Furthermore, a model based on eccentric axial load buckling is being considered, at least to provide a physical point of comparison for the previous model.

## To do

### 3D PH modelling

* ~~Extension to 3D from 2D~~
* ~~Code the computation of along the whole 3D PH curve~~
* ~~Surface generation from 3D PH curve~~

### Visual acquisition

* ~~Set-up vision with python on Linux~~
* ~~Compute exact cylinder from set of points or points+normal~~
* ~~Code RANSAC algorithm with cylinder model~~
* ~~Code b-spline interpolation~~
* ~~Code prior and length and starting point~~
* Code model reconstruction (selection of points)

### Model development

* Redaction of mechanical analysis
* Prior from symmetry on PH cubic
* Sum-up and comparison of the different models

# Robotic arm modelling

To make a prior based registration we need to model the arm deformation to limit the possible configuration. This has already been done with rigid skeleton as prior for human body or hand registration. They used the rigid skeleton to deform the 3D model and then compare the result to a visual input. This method as the advantages to limit the number of configuration parameter (limited number of nodes) and to limit the possible state of those parameter (parameter linked by kinematic model) possibly reducing again the number of parameters.

Here we want to adapt this method to represent a soft robotic arm. The model needs to have the following qualities:

-Limit the number of parameters to describe the configuration (discretisation)

-Constrain the parameter together (Kinematic prior)

-Smoothly reconstruct the deformation along the whole length of the arm (at least G1 continuity)

We can see that we have two different prior, the prior on the configuration parameter and the prior on the interpolated point.

In the case of a rigid skeleton the configuration prior is a fixed distance between point reducing the parameter to angle of rotation and the interpolation prior is to consider straight line interpolation. Additionally, prior limiting the angles of rotation can be used.

Add analysis on the errors of this model for our case

To account for a flexible arm both prior needs to be modified

## Global parameterization

The arm will be divided in section of equal length by points , each point will be defined by its position vector in the global frame of reference , and by its orientation represented as the rotation from the global frame of reference to the local orientation frame by a unitary quaternion .

Later discuss the possibility of varying section length

The first point is fixed to the global reference frame

Each point can also be defined in the frame of reference of the point

And also

This notation , can be more convenient for the construction of the prior.

Then the shape of the arm is constructed using a function

If the interpolation is only computed piecewise we have

With

## Rigid skeleton

Classic rigid skeleton kinematic prior with C2 spline interpolation

This model gives a kinematic prior

The endpoint is totally defined by its rotation :

It’s a strong prior, decreasing the coordinate parameter by 1

The interpolation can be computed with spline to get C2 continuity

## Piecewise constant curvature (PCC) description

Each section is defined as an arc. The end point is completely defined by its rotation If we break down this rotation two angles and see figure we get:

We can also denote the curvature radius and the center of the arc :

The interpolation is solely defined piecewise and is defined in the frame of by:

Which implies

L

This model gives a kinematic prior

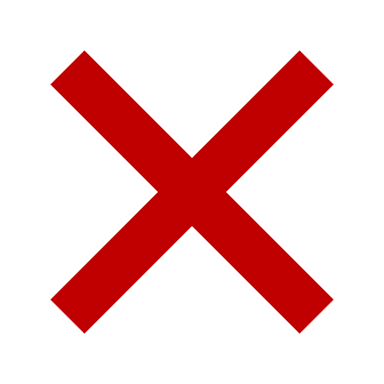
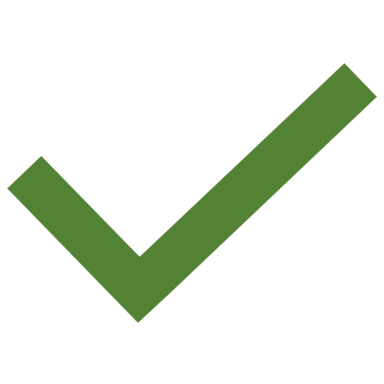
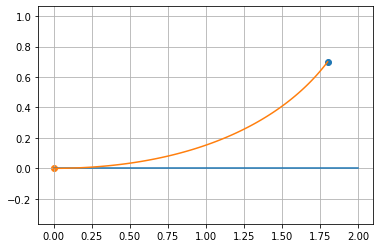
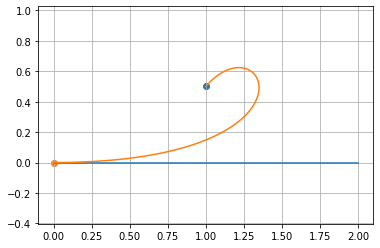
It’s a strong prior, decreasing the coordinate parameter by 1, additionally there is no degree of freedom on the orientation at end point

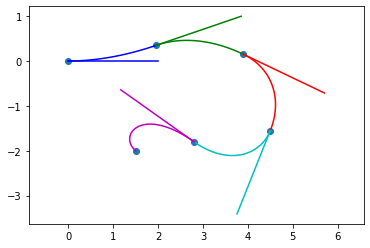
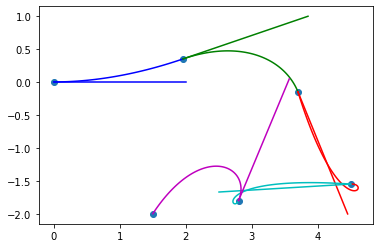
## Cubic Pythagorean hodograph curve (PH curve) 2D conceptualization

Pythagorean hodograph curves have defined length. We will use this property to construct curves of fixed length L to model a section of the arm deforming. Additionally, we want the curve to be at the initial point. This method of construction gives two possible solution one exhibiting looping behavior we will exclude those solution

This model gives a kinematic prior < L

A prior limiting the rotation of the orientation (hypothesis: the section is short thus can’t deform a lot)





Considering the interpolation between and for a length L and a tangent at defined by a complex number. We put , c being on the real line.

The following demonstrations are taken from \*ref and further detail are added some calculation for coding and prepare for later expansion to 3D cases

### PH curves

In our case (cubic) and thus

### Rectifying polygon

is the rectifying polygon of is length is L

To have tangency at

can be defined as a linear Bezier function from P using

For a non-looping behaviour, we want all sign to be of same sign \*ref

From \*ref

If then

Solution outside are discarded

With those solution we have the coordinate of points , we can get the coordinate using the ellipse equation, knowing that we are on the upper semi-ellipse we have

Then

### Hodograph reconstruction (

### Interpolation

If “sign” are alternating the term changes but not and

### Symmetry prior

If

If

else

2nd method

Let’s construct P for a symmetrical curve

P

Can then be reconstructed:

and can be reconstructed:

Tangency condition

### Reparameterization

The speed along the spline is not constant and is equal to , thus if we want to keep the speed constant (to generate evenly spaced point on it) we need to change the parameterization.

For this we introduce the variable

## Degenerated 3D cubic Pythagorean hodograph curve (PH curve)

Given the same constraint, , and the problem lies on a single plane. We can solve the problem in 2D if we translate the 3D problem onto the plane.

First, we apply a translation by to get ,

Then we find the normal vector of the plane

And compute the rotation that gives , which is the rotation around the vector by an angle

Then we apply this rotation to , and and achieve , and . All point now lies on the plane. We can thus treat their and coordinate as complex number and apply the same process as for the 2D case, the only difference being that the problem is already centered, to get to the canonical problem only the in-plane rotation is needed.

The process (see 2.4) will give us the curve as well as the hodograph control point and The 3D information is finally computed using:

## Mechanical model: eccentric buckling

Each section can be model as a column under an eccentric axial load

## Surface reconstruction

The surface is reconstructed from by generating k lines defined by

With offsetting the curve by r the radius of the shape

Lastly quad are generated from each set of points ( being the discretized version of )

As well as closing sets

# Prior based 3D registration

## Cylinder estimation

### Naïve approach: regression to a line

### Computing cylinder from a set of points

An infinite cylinder is defined by a direction a point of passage and a radius

We can find a cylinder using 5 points but this method can yield non-unique solution depending on the input points. To get around this issue we can first estimate the normal of the surface at each point (using open3d) then we can use 3 points and 2 normal to compute the cylinder

We first find the direction of the cylinder using the normal:

* If and are colinear this method does not work. In this configuration there is either no solution or an infinite number of solution if points and normal are aligned.

Then to find a point of passage and the radius of the cylinder we can solve for a circle in any plane orthogonal to . For convenience we solve it in the plane by applying the rotation which put on the axis

We then use the complex points

Equation of a circle

Applying the linear transformation we get

The system becomes:

### Computing cylinder with prior on diameter

To mitigate the impact on noise on the calculated radius we want to have the possibility to compute cylinder with a prescribed radius . This task can be solve using 2 points with normal. The solution is analogous to the previous one the difference lies in the center point computation.

Given two point and in the plane orthogonal to the cylinder we have two potential circle center being at a distance from both points.

To discriminate the center, we choose the one closer to the intersection of the projected normal of the points.

### Finite cylinder computation

The two previous method define exactly an infinite cylinder given two points and normal. What we want to achieve is computing a finite cylinder given a noisy point cloud containing artifacts.

Here we will use the RANSAC algorithm to reject outliers such as 3D artifacts from the camera or from imperfect filtering. Then we will find the length and center of the cylinder using the extreme inliers points in the cylinder direction.

The RANSAC algorithm error is the distance to the cylinder and is computed as follow:

With the center of the cylinder (corrected or not) the direction of the cylinder and the radius of the cylinder

### Cylinder computation using covariance and radius prior

## Non-rigid Cylinder

In order to use the algorithm seen in 3.1 on flexible tubes the later need to be discretized into rigid cylinder. To achieve this, we divide the input point cloud into sub-cloud based on a voxel grid. Voxel needs to be taken small enough to consider the surface as a portion of rigid cylinder.

This method returns a set of cylinder center point, depending on the noise and camera artifact those can contain a number of outliers.

Those outliers occur when the noise create a stronger curvature than the cylinder, misleading the algorithm in choosing a center point outside the actual tube.

## Tube reconstruction

### B-spline

### B-spline with length constraint

### B-spline with prior on starting position

### PCC reconstruction

### PCC reconstruction with occlusion