Mini-project 1
Position Analysis of wing mechanisms and quadrotor formations

In this mini-project you'll put into practice the position analysis methods explained in the course. You'll begin by simulating the motion of Smartbird's wing mechanism, which is constructible, and hence can be analyzed using Geogebra. You will next attempt the localization of a robot formation using range measurements. The problem is equivalent to solving the forward kinematics of a parallel 3-RPR robot, which is known to have no constructible solution. Thus, it is an ideal setting of application for the methods in the CUIK suite.

Simulation of Smartbird's wing (3/10 points)

Figure 1 shows the mechanical structure of Smartbird's wing. Every small circle represents a revolute joint.

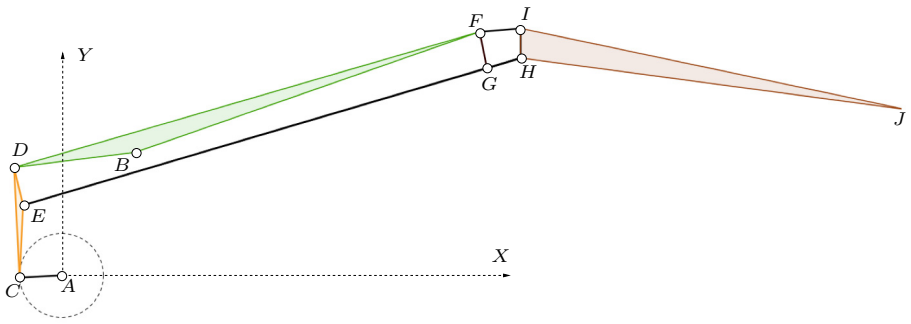

Figure 1: Mechanical structure of Smartbird's wing.

Points A and B are connected to the bird's body, which we consider as the "ground" link. Bars EG and GH are separately drawn, but form a single rigid body. The wing is moved by actuating the revolute joint in A. There is only one input coordinate: the angle θ of bar AC relative to the X axis.

1. (0.5 points) Show that once θ is fixed, the configuration of the wing is geometrically constructible by means of a sequence of bilaterations. Describe the sequence in human language (i.e., do not use Geogebra commands).

2. (2 points) Use Geogebra to construct the configuration shown, assuming that

$$A = (0, 0) \qquad B = (1.596, 2.638)$$

and the following point-to-point distances:

$$\begin{array}{llll} l_{AC} = 0.898 & l_{BD} = 2.640 & l_{GH} = 0.755 & l_{GH} = 0.755 \\ l_{CD} = 2.368 & l_{DF} = 10.415 & l_{EH} = 11.158 & l_{HI} = 0.639 \\ l_{CE} = 1.546 & l_{BF} = 7.821 & l_{FG} = 0.792 & l_{IJ} = 8.472 \\ l_{DE} = 0.843 & l_{EG} = 10.403 & l_{FI} = 0.893 & l_{HJ} = 8.360 \end{array}$$

Activate the animation of point C following the drawn circle. See how the wing moves. A motion similar to the one in http://youtu.be/N9b45bRSlG8 should arise. Provide the `ggb` file you obtained, with all auxiliary objects hidden.

3. (0.5 points) Activate the tracing of points F, H, and J and provide a snapshot of the resulting paths with one wing configuration overlaid.

Quadrotor localization (7/10 points)

Figure 2 shows a terrain with three range sensors placed at points 1, 2, and 3. At the shown instant, three quadrotors are hovering at points 4, 5, and 6. We wish to localize the sensors and quadrotors in the XY reference frame, assuming that all sensor-sensor distances are known, and that the distances indicated as dashed segments are measured. The problem does not admit a geometrically constructible solution so that we inevitably need to solve it numerically using CUIK. We denote by (x_i, y_i) the coordinates of point i, and by $d_{i,j}$ the distance between points i and j. The sensor in point 1 can also measure the angle θ indicated, with some error. I.e., it provides an additional constraint of the form $\theta \in [\theta_{min}, \theta_{max}]$.

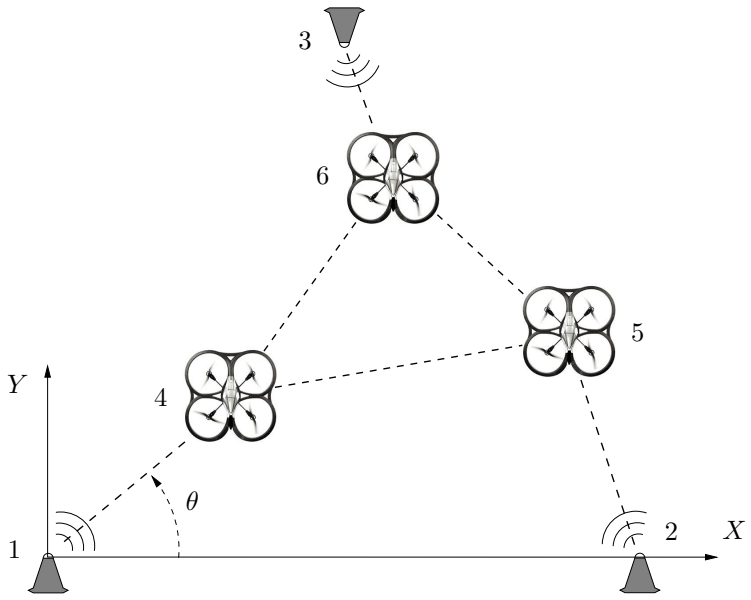

Figure 2: A localization problem on a quadrotor formation.

1. (0.5 points) Using constraints of the form $(x_i - x_j)^2 + (y_i - y_j)^2 = d_{i,j}^2$, formulate a system of equations to determine the (x_i, y_i) positions of all points[1]. Indicate which are the constants and variables of the system.

2. (0.5 points) Since CUIK is a solver of polynomial equations, it cannot work directly with angles. Then, how can we enforce the constraint $\theta \in [\theta_{min}, \theta_{max}]$ using CUIK?

3. (0.5 points) Consider the system formed by the constraints in 1 and 2. Determine the number of variables and equations, and the expected dimension of its solution set?

4. (0.5 points) Provide a box bounding the solutions of this system. Justify your choice.

5. (2 points) Using CUIK, solve the system obtained in 1 and 2 using the box in 4, in each of the three situations indicated in the following table. Use $\sigma = 0.05$ and assume that all sensor-to-sensor distances are d_s, while all quadrotor-to-quadrotor ones are d_q.

Situation	d_s	d_q	$d_{1,4}$	$d_{2,5}$	$d_{3,6}$	θ_{min}	θ_{max}
A	3	1	2	2	2	$-3°$	$3°$
B	3	1	2	2	2	$0°$	$360°$
C	3	3	4	4	4	$0°$	$360°$

   Provide the `*.cuik` and `*.param` file of Situation A, and a table with the quadrotor positions in situations A and B.

   Note 1: The constraint $(x_i - x_j)^2 + (y_i - y_j)^2 = d_{i,j}^2$ should be written as follows in CUIK:

```
xi^2+xj^2-2*xi*xj+yi^2+yj^2-2*yi*yj = dij^2;
```

   Note 2: In situations B and C there was a failure of the angle sensor; that's why θ is left unconstrained in them.

6. (2 points) For each situation, draw the solution configurations. I.e., obtain a drawing similar to the one in Fig. 2 for each solution returned by CUIK. To this end, use the command

```
~/CuikSuite/bin/cuiksols2samples file.sol,
```

   which generates a `*.links` file with the center points of all solution boxes, and then feed this file into the MATLAB program of Appendix A or a similar one.

7. (1 point) For each situation, also plot the (y_4, x_5) coordinates of the solution points using the `plot2D` function in Appendix B. Note that case C has a solution space formed by isolated points and a closed curve. Why does the closed curve of solutions arise? What do the isolated points correspond to?

[1]The problem can also be formulated using loop-closure constraints, but we here prefer distance constraints to simplify the problem, and to work out alternative formulations.

Appendix A

The following Matlab code can be used to plot the quadrotor configurations, assuming that the (x_i, y_i) variables in your CUIK file are $x_4, y_4, x_5, y_5, x_6, y_6$, and appear declared in this order.

```
function plotQuadrotor(filename)

    % Intersensor distance
    ds = 3;

    % Sensor coords
    x1 = 0;    y1 = 0;
    x2 = ds;   y2 = 0;
    x3 = ds/2; y3 = ds*sin(pi/3);

    % Get the (x,y) coords of the solution configurations
    M = dlmread(filename);

    % Bounding box of the solution (x,y) coords
    Xs = [M(:)' x1 x2 x3];
    Ys = [M(:)' y1 y2 y3];
    Xmin = min(Xs);  Ymin = min(Ys);
    Xmax = max(Xs);  Ymax = max(Ys);

    % Open figure
    figure(); hold on; axis equal;

    % Plot sensors
    plot([x1,x2,x3],[y1,y2,y3],'ko','markersize',8,'markerfacecolor','k');

    % Plot quadrotor configurations
    Msize = size(M,1);
    for i=1:40:Msize
        % Get quadrotor locations
        x4 = M(i,1); y4 = M(i,2);
        x5 = M(i,3); y5 = M(i,4);
        x6 = M(i,5); y6 = M(i,6);

        % Plot quadrotors
        patch([x4 x5 x6],[y4 y5 y6],'k','linewidth',1,'facealpha',0.1);

        % Plot sensor-quadrotor lines
        plot([x1 x4],[y1 y4],'--k');
        plot([x2 x5],[y2 y5],'--k');
        plot([x3 x6],[y3 y6],'--k');
    end

    % Set axes ranges
    eps = Xmax/10;   axis([Xmin-eps Xmax+eps Ymin-eps Ymax+eps]);
```

Appendix B

```
function plot2D(filename)

    % Read file
    M = dlmread(filename);

    % Get the y4 and x5
    X = M(:,2); % y4
    Y = M(:,3); % x5

    Xmin = min(X); Xmax = max(X);
    Ymin = min(Y); Ymax = max(Y);

    % Start figure
    figure(); hold on;

    % Plot points (y4,x5)
    plot(X,Y,'.','markersize',20);

    % Set axes ranges
    eps = 0.2;
    axis([Xmin-eps Xmax+eps Ymin-eps Ymax+eps]);

    % Equally-scaled boxed axes, with labels y4 and x5
    axis equal;
    box ON;
    xlabel('y_4');
    ylabel('x_5');
```