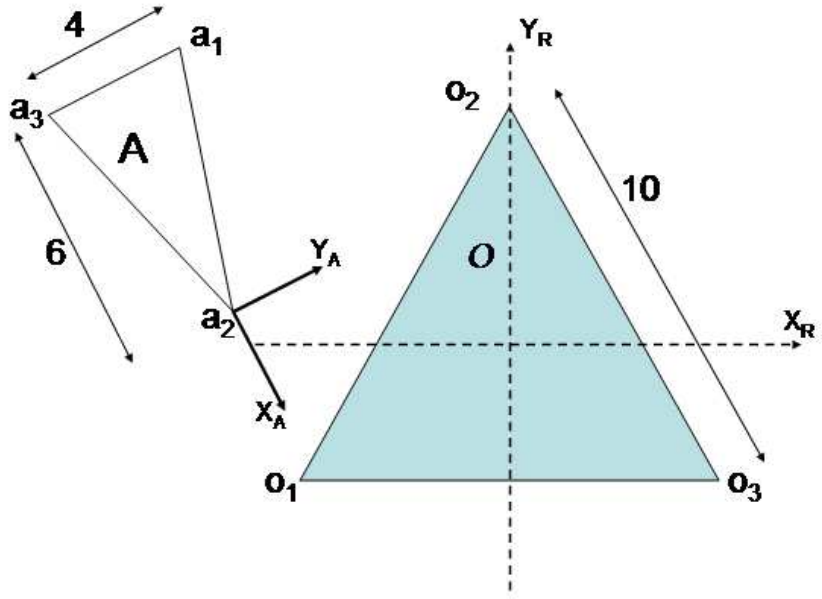
ME 115(b): Homework #3 Solutions

(Due Friday, May 2, 2008)

Consider the convex polygonal robot, $\mathcal{A}$, and obstacle, $\mathcal{O}$, shown in Figure . The obstacle is an equilaterial triangle with side dimension of 10 units, whose center is coincident with the origin of the fixed workspace observing reference frame (whose axes are denoted by X_R and Y_R). One triangle face is parallel to the x-axis of the workspace reference frame. The robot is an isoceles triangle whose base dimension is 4 and whose height is 6. Its body fixed reference frame is located so that its x-axis is aligned with the triangle's centerline, and its origin



Problem 1 & 2:

- Write a Mathematica (or other programming language) function to create the outline of the c-obstacle for a fixed orientation of $\mathcal{A}$. Create the c-obstacle outline for the case of $\theta = 45^o$, where θ is the orientation of $\mathcal{A}$.
- Using the function from Problem 1, create an visualization of the c-obstacle by superimposing on a single 3-dimensional view the constant orientation c-obtacle boundaries for orientations of $\mathcal{A}$ in 10^o increments (in the range $\theta \in [0^o, 360^o]$). That is, plot 36 constant orientation slices (with each orientation differing by 10^o) on a single 3-dimensional view (with the axes being x, y, and θ).

Solution: NOTE: There are many ways to solve this problem. The following MATLAB code is one possible approach, using the Minkowski Difference Method:

```
%======================
%PROBLEM 1 & 2
%======================
% The approach taken here was to use the Minkowski Difference Method. We
% construct a vector of points corresponding to the robot's boundary. We
% then construct a vector of points corresponding to the obstacle
% boundaries. Once we have both sets of vectors, we then proceed to take all
% possible combinations of the difference of both sets and populate an
% entire space of points. The convex hull of those points then yields the
% outline of the c-obstacle for varying values of theta

GENERATE_PROB1_FIGS = 0;
GENERATE_PROB2_FIGS = 1;

%======================
%construct the set of Obstacle vectors
%======================
c = 10*sin(30*pi/180)/sin(120*pi/180);
O1.x = -c*sin(60*pi/180);
O1.y = -c*cos(60*pi/180);
O2.x = 0;
O2.y = c;
O3.x = c*sin(60*pi/180);
O3.y = -c*cos(60*pi/180);

%incremental step
dth = 0.1;

x = [O1.x:dth:O3.x];
for(k=1:length(x))
    if(x(k)>=O1.x && x(k)<O2.x)
        m = (O2.y-O1.y)/(O2.x-O1.x);
        b = O2.y-m*O2.x;
        y1(k) = m*x(k)+b;
    elseif(x(k)>=O2.x && x(k) < O3.x)
        m = (O3.y-O2.y)/(O3.x-O2.x);
        b = O3.y-m*O3.x;
        y1(k) = m*x(k)+b;
    end
end

y2 = O1.y*ones(1,length(x));

V = [x x; y1 y2];
```

```
X_o = [x x];
Y_o = [y1 y2];


%==================
%Construct the set of A-robot vectors for varying theta values
%==================
d.x = O1.x;
d.y = O1.y;

if(GENERATE_PROB1_FIGS)
    THETA = 45;
elseif(GENERATE_PROB2_FIGS)
    THETA = [0:10:360];
else
    THETA = 45;
    display('did not specify which figures to generate; THETA = 45 by default');
end

for(n=1:length(THETA))

    R = [cos(THETA(n)*pi/180) -sin(THETA(n)*pi/180); sin(THETA(n)*pi/180) cos(THETA(n)*p

    v = R*[-6; 2];
    a1.x = v(1);
    a1.y = v(2);

    v = R*[-6; -2];
    a3.x = v(1);
    a3.y = v(2);

    v = R*[0; 0];
    a2.x = v(1);
    a2.y = v(2);

    x = [];
    y = [];

    if(a1.x==a2.x)
        x = a1.x*ones(1,20);
        y = [min([a1.y a2.y]):1/20:max([a1.y a2.y])];
    else
        x = [min([a1.x a2.x]):dth:max([a1.x a2.x])];
        for(k=1:length(x))
```

```
        m = (a2.y-a1.y)/(a2.x-a1.x);
        b = a2.y - m*a2.x;
        y(k) = m*x(k)+b;
    end
end

X_a = [x];
Y_a = [y];

y = [];


if(a2.x==a3.x)
    x = a2.x*ones(1,20);
    y = [min([a2.y a3.y]):1/20:max([a2.y a3.y])];
else
    x = [min([a2.x a3.x]):dth:max([a2.x a3.x])];
    for(k=1:length(x))
        m = (a3.y-a2.y)/(a3.x-a2.x);
        b = a3.y - m*a3.x;
        y(k) = m*x(k)+b;
    end
end
X_a = [X_a x];
Y_a = [Y_a y];

y = [];

if(a3.x==a1.x)
    x = a3.x*ones(1,20);
    y = [min([a3.y a1.y]):1/20:max([a3.y a1.y])];
else
    x = [min([a3.x a1.x]):dth:max([a3.x a1.x])];
    for(k=1:length(x))
        m = (a1.y-a3.y)/(a1.x-a3.x);
        b = a1.y - m*a1.x;
        y(k) = m*x(k)+b;
    end
end
X_a = [X_a x];
Y_a = [Y_a y];

k = 1;
for(i=1:length(X_o))
    for(j=1:length(X_a))
```

```
            X_q(k) = X_o(i)-X_a(j);
            Y_q(k) = Y_o(i)-Y_a(j);
            k = k+1;
        end
    end

    if(GENERATE_PROB2_FIGS)
        Q_z(:,n) =THETA(n)*ones(1,length(X_q));
        Q_x(:,n) = X_q;
        Q_y(:,n) = Y_q;;
    end

    if(GENERATE_PROB1_FIGS)
        %============================
        %create world coordinates of robot space for plotting purposes
        %============================
        for(k=1:length(X_a))
            v_local = [X_a(k); Y_a(k)];
            D  = [d.x; d.y];
            v_global = v_local + D;
            X_ag(k) = v_global(1);
            Y_ag(k) = v_global(2);
        end
    end
end

figure(1)
if(GENERATE_PROB1_FIGS)
    K = convhull(X_q,Y_q);
    plot(X_q(K),Y_q(K),'g'); hold on
    plot(X_o,Y_o,'.b'); hold on
    plot(X_ag,Y_ag,'.r'); hold on; grid on
    xlabel('x-axis');
    ylabel('y-axis');
elseif(GENERATE_PROB2_FIGS)
    for(i=1:length(THETA))
        u = convhull(Q_x(:,i),Q_y(:,i));
        plot3(Q_x(u,i),Q_y(u,i),Q_z(u,i)); hold on
    end
    grid on
    xlabel('x-axis');
    ylabel('y-axis');
    zlabel('\theta-values');
end
```

Problem 3: Create the function that describes the surface boundary "patch" of the c-obstacle associated with Type EV contact between robot edge $E_1^{\mathcal{A}}$ (which connects vertices a_1 and a_2) and obstacle vertex o_1. Also determine the boundaries of this patch. Plot this patch using Mathematica, Matlab, or another approach.

Solution:

```
%====================
% START PROBLEM 3
%====================
% The approach taken here for this problem was to use the patch
% equations that essentially result in :
% 1) finding an appropriate range of THETA
% 2) Solving one system of equations for one endpt
%    -- Ax + By = -C
%    -- Ex + Fy = -D
% 3) Solving another system of equation for the other endpt
%    -- Ax + By = -C
%    -- Ex + Fy = -D + norm(E_ia)^2

% range of theta was found to be:
% theta = [ -101.565, 18.435 ];

% Start by calculating set of constants
xi_j = 270*pi/180;
xi_jm1 = 150*pi/180;

phi = pi/2-atan2(2,6);

a_i = 0;
a_ip1 = pi-atan2(2,6);

r_i = [0;0];
r_ip1 = [-6; 2];

b_j = 210*pi/180;

o_j = [-5; -5/tan(60*pi/180)];

E_ia = [-6;2];

%create range of theta
THETA_MIN = xi_jm1-phi-pi;
THETA_MAX = xi_j-phi-pi;
```

```
NUMPTS = 50;

D_TH = (THETA_MAX-THETA_MIN)/NUMPTS;

THETA = [THETA_MIN:D_TH:THETA_MAX];

%==========
%PLOT FIGURE
%==========
figure(2)
for(k=1:length(THETA))
   A = -cos(phi+THETA(k));
   B = -sin(phi+THETA(k));
   C = norm(o_j)*cos(phi+THETA(k)-b_j)-norm(r_i)*cos(phi-a_i);

   D = norm(o_j)*norm(r_ip1)*cos(a_ip1-b_j+THETA(k)) ...
       - norm(o_j)*norm(r_i)*cos(a_i  -b_j+THETA(k)) ...
       - norm(r_ip1)*norm(r_i)*cos(a_i-a_ip1)+norm(r_i)^2;
   E = norm(r_i)*cos(a_i+THETA(k))-norm(r_ip1)*cos(a_ip1+THETA(k));
   F = norm(r_i)*sin(a_i+THETA(k))-norm(r_ip1)*sin(a_ip1+THETA(k));

   % Now solve the equations:
   % Ax + By + C = 0
   % Ex + Fy + D = 0
   %
   % | A  B | | x | = | -C |
   % | E  F | | y | = | -D |
   %
   % H * v = u
   % v = inv(H)*u;

   H = [A B; E F];
   u = [-C;-D];
   v = inv(H)*u;

   x1 = v(1);
   y1 = v(2);

   % Now solve the equations:
   % Ax + By + C = 0
   % Ex + Fy + D = norm(E_ia)^2
   %
   % | A  B | | x | = | -C |
   % | E  F | | y | = | norm(E_ia)^2 - D |
   %
```

```
    % H * v = u
    % v = inv(H)*u;

    u = [-C; norm(E_ia)^2-D];
    v = inv(H)*u;

    x2 = v(1);
    y2 = v(2);

    plot3([x1 x2],[y1 y2],[THETA(k) THETA(k)]); hold on; grid on

end
xlabel('x-axis');
ylabel('y-axis');
zlabel('\theta-values');
```