Jesse Wynn
ME 575
February 17, 2018
Design Project 1

Ground Target Interception Optimization

Section I. Title Page

Problem Statement:
Maximize the field-of-view (FOV) from a downward-facing camera on a quadrotor while keeping the relative size of a ground target (in the image frame) large enough such that all target features are guaranteed to be distinguishable to a computer vision algorithm. Design variables for this problem are image sensor width (pixels), square target width (meters), height above ground (meters), and camera focal length (meters).

Main Results:
```
sensor_width = 1114.29 (pixels)
target_size = 1.49 (meters)
height = 29.99 (meters)
focal_length = 4.29e-3 (meters)
fov = 1746.05 (meters^2)
```

Note: The result above was found to be a *local maximum* (discussed further in Section III).

Section II. Procedure

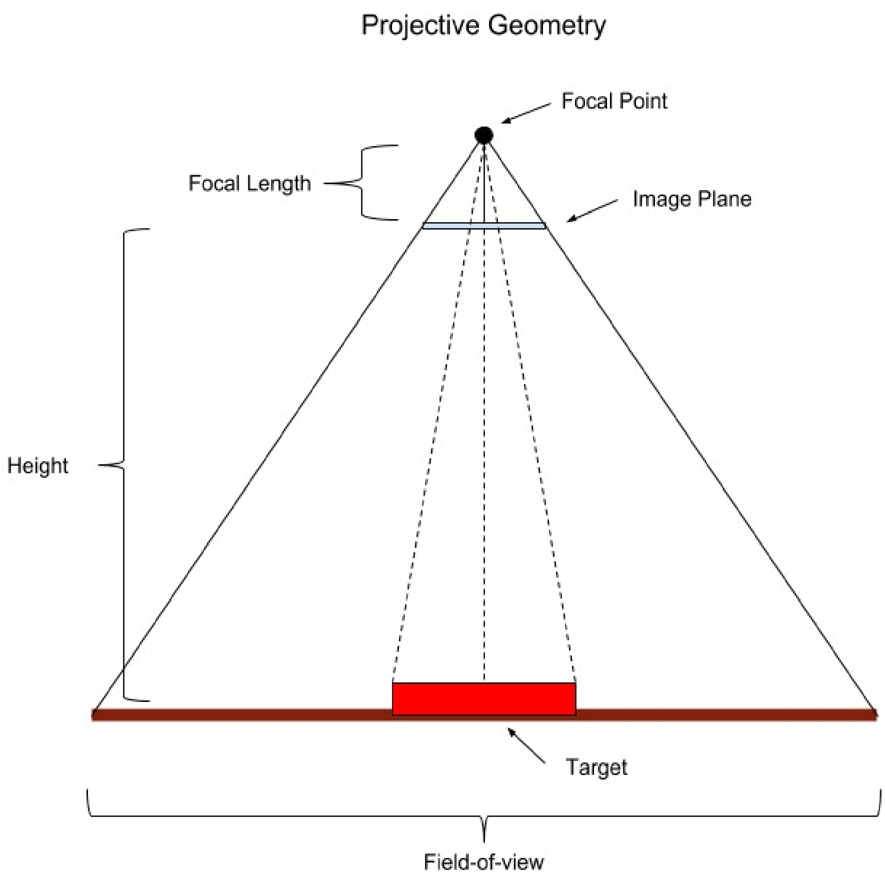
Model Development:

Model development was based off of well-known mathematical models, as well as results from actual experiments that were conducted as part of this project. The fundamental set of equations used in the model stem from projective geometry relationships and the pinhole camera model. The other relationships such as computational cost and image processing requirements (required number of pixels to adequately represent target features) are approximations based on experiments conducted as part of my research. The overall geometry of the problem is described in Figure 1.

Key Assumptions:

- Square image sensor
- Square pixels
- Pinhole camera model is sufficient

Figure 1.



Model Robustness and Testing

Robustness of the model was insured by removing any opportunities for the model to 'blow up' to infinity or 'NaN'. Since the tangent function is used in the model, and tangent is undefined for multiples of pi/2, a simple check with an 'if' statement makes sure that the value passed to tangent is less than pi/2. If the value equals or exceeds pi/2, a value of 0.99*pi/2 is passed to the function instead. Based on several tests of using the model in the optimizer, robustness has been verified through having no model failures.

Optimization Problem Functions and Equations:

Design Variables:

- Height, *h* (meters)
- Focal length, *fl* (meters)
- Sensor width, *ss_pix* (pixels)
- Target size, *ts* (meters)

Design Functions (in-loop sequence): *Compute...*

1. Angular field of view of the camera, *v* (radians)
2. Field-of-view of the downward-facing camera, *fov* (square meters)
3. Size of target in the image plane, *uv_target* (square meters)
4. Pixel size (meters)
5. Size of the target in the image plane, *uv_pix* (pixels)
6. Time to process the image, *t_proc* (seconds)
7. Image processing rate, *rate_proc* (1/seconds)

Model Constraints:

- 0.3mm <= Focal length <= 25 mm
- Image processing rate >= 10
- 0.5 m <= Target Size <= 1.5 m
- Target area (pixels) >= 1600
- Height <= 30 m

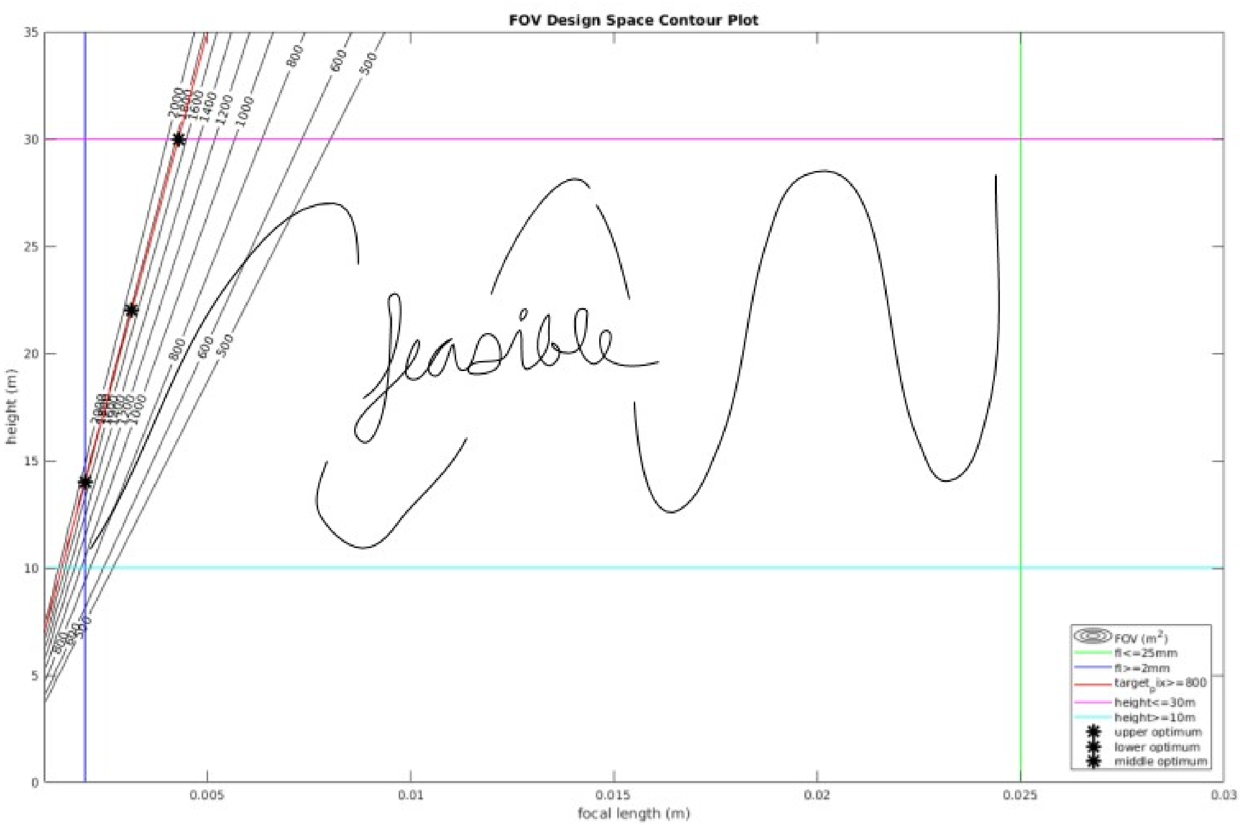
Section III. Results and Discussion of Results

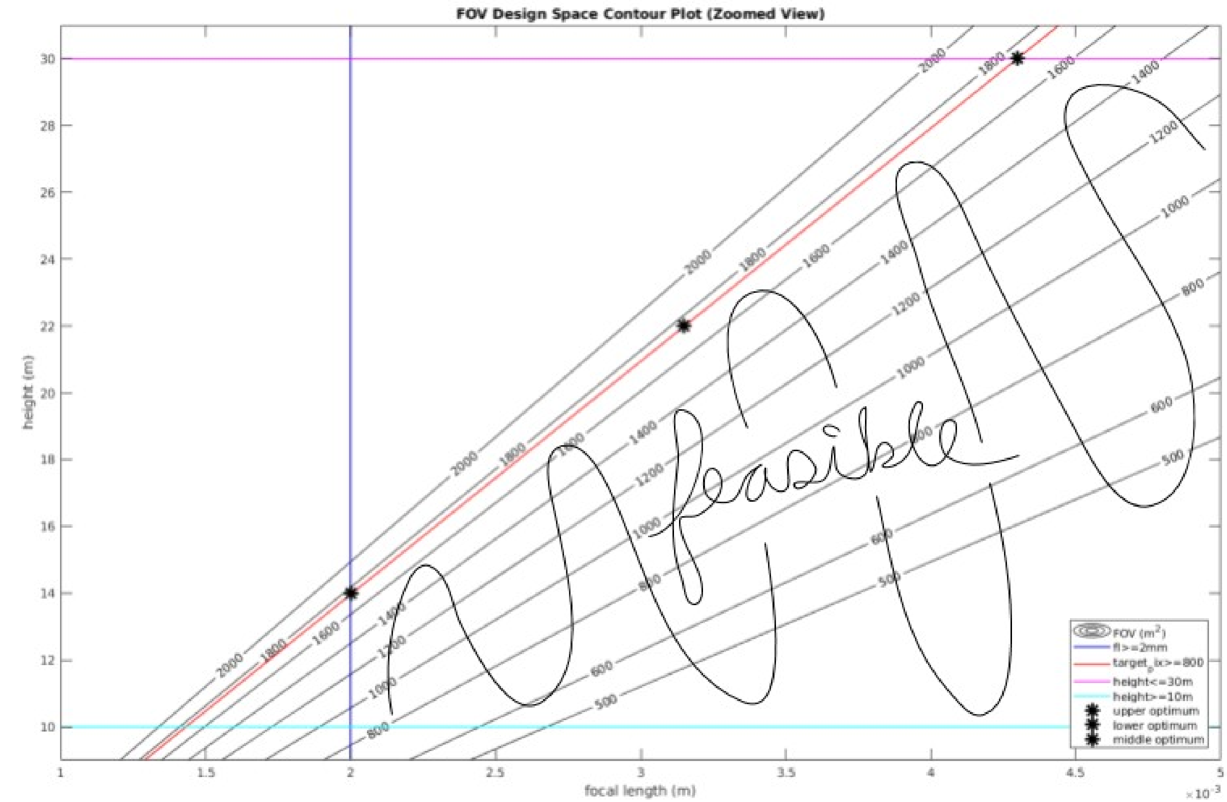
Table 1: Optimum values of variables and functions

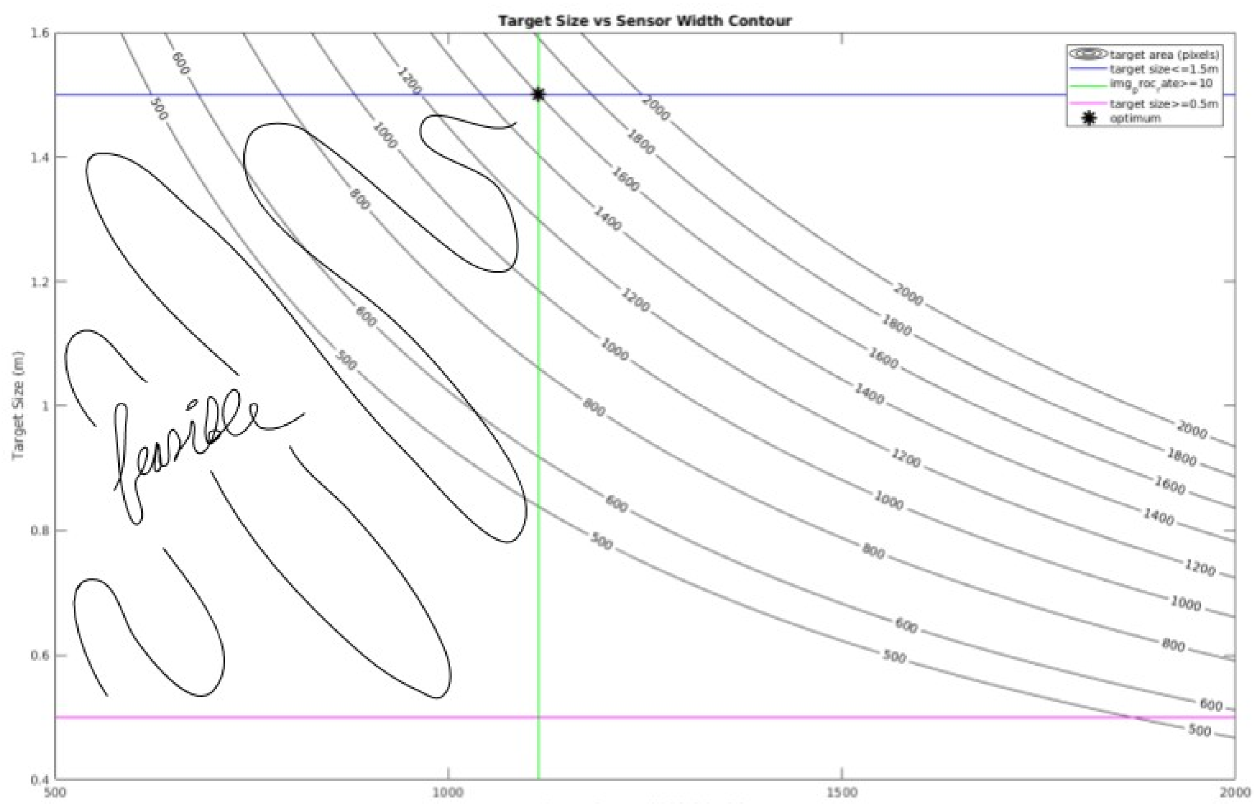
Design Variable / Function	Optimal Value	Constraint Status
Height	29.99 meters	At constraint (binding)
Focal length	4.29 millimeters	Within constraint bounds
Sensor width	1114.29 pixels	Within constraint bounds
Target Size	1.499 meters	At constraint (binding)
Angular field of view	1.217 radians	-
Field-of-view (objective)	1746.05 square meters	-
Target area (image plane)	1.155e-08 square meters	-
Pixel size	5.373e-06 meters	-
Target area (image plane--pixels)	1600 pixels	At constraint (binding)
Time to process image	0.01 seconds	-
Image processing rate	10.0	At constraint (binding)

**After optimization, objective value increased from starting value of 195 to 1746 square meters

Contour Plots




FOV Design Space Contour Plot (Zoomed View)
height (m)
focal length (m)
×10^-3
FOV (m^2)
fl>=2mm
target_pix>=800
height<=30m
height>=10m
upper optimum
lower optimum
middle optimum
feasible


Target Size vs Sensor Width Contour
Target Size (m)
Image Sensor Width (pixels)
target area (pixels)
target size<=1.5m
img_proc_rate>=10
target size>=0.5m
optimum
feasible

Discussion and Observations

As can be seen from the contour plots, the design space for this problem includes a linear region along which many equally-optimal combinations of height and focal length exist. Thus for this problem *there is an infinite number of local optimums*. After a closer inspection of the problem, this outcomes makes sense. Even though there are four design variables, the variables of focal length and height are correlated and both change how the target is perceived in the image plane. For example, the target in the image plane may appear the same with a shorter focal length at lower height as it does with a longer focal length from a greater height. The root of this correlation is because of the similar triangle relationships in the model geometry. If the variable of height or focal length is changed from a design variable to a fixed variable, then a single optimal value can be found. Although initially disappointing that the problem as posed contains no single optimum set of design variables, the results are still very useful. For most fixed-lens cameras, lenses can be purchased in only a select number of focal lengths. The results of this project allow one to select a lens between 2 and 4.3 mm, and then directly look up the optimal height for that lens. As for the two other design parameters, the optimal value for the sensor width in pixels is at the maximum that it can be while respecting the image processing rate constraint, and the physical target size is also at the maximum constraint. Both of these results make sense.

Section IV. Appendix

Fmincon optimization MATLAB script:

```
function [xopt, fopt, exitflag, output] = fov_optimizer()

    % ------------Starting point and bounds------------
    % design variables: height, focal length, sensor size, target size
    x0 = [14, 0.006, 1260, 0.707];  % starting point
    ub = [100, 0.015, 2000, 5.0];  % upper bound
    lb = [1, 0.001, 300, 0.25];  % lower bound

    % ------------Linear constraints------------
    A = [];
    b = [];
    Aeq = [];
    beq = [];

    % ------------Objective and Non-linear Constraints------------
    function [f, c, ceq] = objcon(x)

        % set objective/constraints here

        % design variables (things we'll adjust to find optimum)
        h = x(1);  % approach height or distance to target (m)
        fl = x(2);  % camera focal lenght (m)
        ss_pix = x(3);  % sensor size (width of square pixel array in pixels)
        ts = x(4); % target_size (width of square target) (m)

        % other analysis variables (constants that the optimization won't touch)
        ss_physical = 1/3;  % physical sensor size in inches (diagonal)
        min_target_pixels = 800;  % minimum number of pixels that can
                                  % represent the target and still have
```

```
                                % the target's features be
                                % distinguishable.
    F_safe = 2;  % safety factor min_target_pixels;

    % analysis functions
    ss_w = ss_physical * sqrt(2)/2;  % physical sensor width (inches)
    ss_w = ss_w * 0.0254;  % convert sensor width to meters
    v = 2 * atan(ss_w/(2*fl));  % angular field of view of the camera

    % here we don't let tan(x) blow up to infinity
    if (v/2) < 0.99*pi/2
        fov_w = 2 * (h * tan(v/2));  % width of rectangular region camera can see
    else
        fov_w = 2 * (h * tan(0.99*v/2));
    end

    fov_h = fov_w;  % height of rectangular region camera can see

    u_target = (fl/h) * (ts/2);  % projection of the target onto the image plane (meters)
    v_target = u_target;  % same as above since target is square (meters)
    pix_size = ss_w/ss_pix;  % pixel size (meters)
    target_area_pix = (2 * u_target/pix_size) * (2 * v_target/pix_size);  % area of target in pixels

    % rate at which images can be processed is directly proportional to
    % the area (in pixels) of the sensor
    tpp = 8.0539e-8;  % time it takes to process each pixel (found from experimentation)
    t_proc = tpp * ss_pix^2;  % time to process the image
    rate_proc = 1/t_proc;  % rate at which images can be processed


    % what we're optimizing
    fov = fov_w * fov_h  % area in square meters that camera can see


    % objective function (what we're trying to optimize)
    f = -fov;  % maximize Field-of-view (m^2)

    % inequality constraints (c<=0)
    c = zeros(7,1);
    c(1) = (fl - 0.025)*1;  % focal length <= 25 mm
    c(2) = (-fl + 0.002)*1;  % focal length >= 2 mm
    c(3) = -rate_proc + 10;  % image processing rate >= 10 hz
    c(4) = ts - 1.5;  % target size (width) <= 1.5 meters
    c(5) = -ts + 0.5;  % target size (width) >= 0.3 meters
    c(6) = -target_area_pix + F_safe * min_target_pixels;  % target's area in pixels >=
safety_factor * 800
    c(7) = h - 30;  % height <= 30 meters

    % equality constraints (ceq=0)
    ceq = [];  % empty when we have none

  end

  % ------------Call fmincon------------
  options = optimoptions(@fmincon, 'display', 'iter-detailed');
  options.StepTolerance = 1e-20;
  options.MaxFunctionEvaluations = 60000;
  [xopt, fopt, exitflag, output] = fmincon(@obj, x0, A, b, Aeq, beq, lb, ub, @con, options);
```

```
    % ------------Separate obj/con (do not change)------------
    function [f] = obj(x)
            [f, ~, ~] = objcon(x);
    end
    function [c, ceq] = con(x)
            [~, c, ceq] = objcon(x);
    end
end
```

Contour plot scripts:

```
% This script constructs a contour plot for the fov optimization
clc
clear
close all

% height vs. focal length
res = 100;

% limits for zoomed out view
fl_min = 0.001;
fl_max = 0.03;
h_min = 0.0;
h_max = 35.0;

[fl,height] = meshgrid(fl_min:(fl_max - fl_min)/res:fl_max, h_min:(h_max - h_min)/res:h_max);

% limits for zoomed in view
fl_min = 0.001;
fl_max = 0.005;
h_min = 9.0;
h_max = 31.0;

[fl_zoom,height_zoom] = meshgrid(fl_min:(fl_max - fl_min)/res:fl_max, h_min:(h_max -
h_min)/res:h_max);


res = 100;

fov = zeros(res+1);
target_area_pix = zeros(res+1);
target_area_pix_zoom = zeros(res+1);

% constants
ss_physical = 1/3;
F_safe = 2;
min_target_pixels = 800;

% optimums for the variables (from optimization)
ss_pix = 1114.286537365992;  % optimal pixel size
ts = 1.499999999828;  % optimal target size

h_opt_up = 29.999;
fl_opt_up = 0.004297954694;
h_opt_low = 13.99;
```

```
fl_opt_low = 0.002;
h_opt_mid = 22;
fl_opt_mid = 0.003148862;

% design variables at mesh points
% [fl,height] = meshgrid(fl_min:(fl_max - fl_min)/res:fl_max, h_min:(h_max - h_min)/res:h_max);

% equations
ss_w = ss_physical * sqrt(2)/2;  % physical sensor width (inches)
ss_w = ss_w * 0.0254;  % convert sensor width to meters
for i=1:length(fl)
    for j=1:length(fl)
        v = 2 * atan(ss_w/(2*fl(i,j)));  % angular field of view of the camera
        fov_w = 2 * (height(i,j) * tan(v/2));  % width of rectangular region camera can see
        fov_h = fov_w;  % height of rectangular region camera can see
        fov(i,j) = (fov_w * fov_h);  % area in square meters that camera can see

        u_target = (fl(i,j)/height(i,j)) * (ts/2);  % projection of the target onto the image plane (meters)
        v_target = u_target;  % same as above since target is square (meters)
        pix_size = ss_w/ss_pix;  % pixel size (meters)
        target_area_pix(i,j) = (2 * u_target/pix_size) * (2 * v_target/pix_size);  % area of target in
pixels
    end
end

for i=1:length(fl_zoom)
    for j=1:length(fl_zoom)
        v = 2 * atan(ss_w/(2*fl_zoom(i,j)));  % angular field of view of the camera
        fov_w = 2 * (height_zoom(i,j) * tan(v/2));  % width of rectangular region camera can see
        fov_h = fov_w;  % height of rectangular region camera can see
        fov_zoom(i,j) = (fov_w * fov_h);  % area in square meters that camera can see

        u_target = (fl_zoom(i,j)/height_zoom(i,j)) * (ts/2);  % projection of the target onto the image
plane (meters)
        v_target = u_target;  % same as above since target is square (meters)
        pix_size = ss_w/ss_pix;  % pixel size (meters)
        target_area_pix_zoom(i,j) = (2 * u_target/pix_size) * (2 * v_target/pix_size);  % area of target
in pixels
    end
end



figure(1)
% [C,h] = contour(fl,height,fov,[500:100:2000],'k');
[C1,h1] = contour(fl,height,fov,[500, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000],'k');
clabel(C1,h1,'Labelspacing',250);
title('FOV Design Space Contour Plot');
xlabel('focal length (m)');
ylabel('height (m)');
hold on;
% solid lines to show constraint boundaries
contour(fl,height,(fl - 0.025),[0.0,0.0],'g-','LineWidth',1);
contour(fl,height,(-fl + 0.002),[0.0,0.0],'b-','LineWidth',1);
contour(fl,height,(-target_area_pix + F_safe * min_target_pixels),[0.0,0.0],'r-','LineWidth',1);
contour(fl,height,(height - 30),[0.0,0.0],'m-','LineWidth',1);
contour(fl,height,(height - 10),[0.0,0.0],'c-','LineWidth',1);
plot(fl_opt_up,h_opt_up,'k*','MarkerSize',12,'LineWidth',2)
```

```
plot(fl_opt_low,h_opt_low,'k*','MarkerSize',12,'LineWidth',2)
plot(fl_opt_mid,h_opt_mid,'k*','MarkerSize',12,'LineWidth',2)
% show a legend
legend('FOV (m^2)','fl<=25mm','fl>=2mm','target_pix>=800','height<=30m', ...
      'height>=10m','upper optimum','lower optimum','middle optimum', 'Location','SouthEast')

figure(2)
% [C,h] = contour(fl,height,fov,[500:100:2000],'k');
[C2,h2] = contour(fl_zoom,height_zoom,fov_zoom,[500, 600, 800, 1000, 1200, 1400, 1600, 1800,
2000],'k');
clabel(C2,h2,'Labelspacing',250);
title('FOV Design Space Contour Plot (Zoomed View)');
xlabel('focal length (m)');
ylabel('height (m)');
hold on;
% solid lines to show constraint boundaries
%contour(fl,height,(fl - 0.025),[0.0,0.0],'g-','LineWidth',1);
contour(fl_zoom,height_zoom,(-fl_zoom + 0.002),[0.0,0.0],'b-','LineWidth',1);
contour(fl_zoom,height_zoom,(-target_area_pix_zoom + F_safe * min_target_pixels),
[0.0,0.0],'r-','LineWidth',1);
contour(fl_zoom,height_zoom,(height_zoom - 30),[0.0,0.0],'m-','LineWidth',1);
contour(fl_zoom,height_zoom,(height_zoom - 10),[0.0,0.0],'c-','LineWidth',1);
plot(fl_opt_up,h_opt_up,'k*','MarkerSize',12,'LineWidth',2)
plot(fl_opt_low,h_opt_low,'k*','MarkerSize',12,'LineWidth',2)
plot(fl_opt_mid,h_opt_mid,'k*','MarkerSize',12,'LineWidth',2)
% show a legend
legend('FOV (m^2)','fl>=2mm','target_pix>=800','height<=30m', ...
      'height>=10m','upper optimum','lower optimum','middle optimum', 'Location','SouthEast')
```

ME 575
Project 1: Design Project
Due Feb 14 at 11:50 p.m.

For your first project, you should select a design problem from you own discipline, develop the model and optimize it. For mechanical engineering students, some possible projects might be,

- Optimizing a mechanism
- Optimizing a control system
- Optimizing a truss
- Optimizing a heat exchanger
- Optimizing a problem from your research

Students in other disciplines should select a problem from their discipline they are interested in. Although the selection of the model is up to you, there are some requirements:

- You must be involved in developing the model; you can't just copy it from somewhere. However, this doesn't mean you have to build it from scratch.
- The model needs to include a minimum of four design variables and three design functions (more is usually better if they are realistic).
- There should be trade-offs involved with the model so we can't just guess the solution.
- The model should be continuous and differentiable, so it can be solved by a routine like `fmincon`. You may use whatever state-of-the-art routine or language you wish.
- I would expect the difficulty of the project to be a cut above the homework.

You may work on this in groups of no more than three if you wish. You are welcome to send a note to the TA indicating you'd like to find a group.

Turn in a report with the following sections (please keep to no more than five pages, single spaced, not including the Appendix or Title page):

1) Title Page with Summary. Give a brief description of the problem (less than 50 words) and your main results.
2) Procedure:
   a. Discuss the development of the model so I can understand it. This would include presenting important relationships and assumptions. Illustrations are helpful. I would appreciate having some sense of the computation sequence, as is given in Section 2.8.5 of the notes.
   b. Indicate how the model was tested and insured to be accurate and robust.
   c. Discuss what the optimization problem is by listing the design variables and functions.

3) Results and Discussion of Results:
   a. Provide a table showing the optimum values of variables and functions, with binding constraints and/or variables at bounds highlighted. How much did the objective improve over the starting design? Do you feel the optimum design is realistic? Visual representations of the optimum are encouraged.

b. Briefly discuss the optimum and the design space around the optimum. Include contour plots as appropriate, with the feasible region shaded and the optimum marked. Comment on what you learn about the design space from these. Do you feel this is a global optimum? Provide support for your conclusion.
c. Include any other observations you feel are pertinent. These may relate to the model, the results, the optimization process, the nature of the optimum, etc. This section should be a half page or less.

4) Appendix:
   a. Listing of `MATLAB or other programs`

Please turn in as a pdf on Learning Suite. Note: Include requested items (such as graphs or tables) in their respective sections as given above, and not in the Appendix. Any output from MATLAB should be integrated into the report with captions, explanatory comments, etc.