Assignment 2
Single View Metrology

Due Date: Thursday, March 14th

University of Illinois at Urbana-Champaign
Spring 2017
CEE 598
Visual Data Sensing And Analytics
For Civil Infrastructure Engineering and Management

Insturctor:
Professor Mani Golparvar-Fard

TEAM 11.2
Juan Diego Nunez Morales (jdnunez2)
Gene Kim (kim352)

PART A

- Vanishing Points (u, v) Code:

The following code was used for the purpose of calculating each one of the vanishing points:

```
function vp1 = getVanishingPoint(img)
% output vanishing point, input image

figure(1), hold off, imagesc(img)
hold on

% Allow user to input line segments; compute centers, directions, lengths
disp('Set at least two lines for vanishing point')
lines = zeros(3, 0);
line_length = zeros(1,0);
centers = zeros(3, 0);
while 1
    disp(' ')
    disp('Click first point or q to stop')
    [x1,y1,b] = ginput(1);
    if b=='q'
        break;
    end
    disp('Click second point');
    [x2,y2] = ginput(1);
    plot([x1 x2], [y1 y2], 'b')
    lines(:, end+1) = real(cross([x1 y1 1]', [x2 y2 1]')); % gives abc vector
of the line
    line_length(end+1) = sqrt((y2-y1)^2 + (x2-x1).^2);
    centers(:, end+1) = [x1+x2 y1+y2 2]/2;
end

%% Solve for vanishing point
% Insert code here to compute vp (3x1 vector in homogeneous coordinates)
% Doing Averaging of Intersection Points - 4 lines have to be selected for
% each vanishing point following a certain order (last 4 are the vertical)
inters_pts = zeros(3,0);
k = 1;
for i = 1:size(lines,2)
    for j = 1:size(lines,2)-i
        inters_pts(:,k) = cross(lines(:,i),lines(:,i+j));
        k = k + 1;
    end
end

vp1 = mean(inters_pts(:,1:4),2);

%% Display
hold on
bx1 = min(1, vp1(1)/vp1(3))-10; bx2 = max(size(img,2), vp1(1)/vp1(3))+10;
by1 = min(1, vp1(2)/vp1(3))-10; by2 = max(size(img,1), vp1(2)/vp1(3))+10;

for k = 1:size(lines, 2)
    if lines(1,k)<lines(2,k)
```

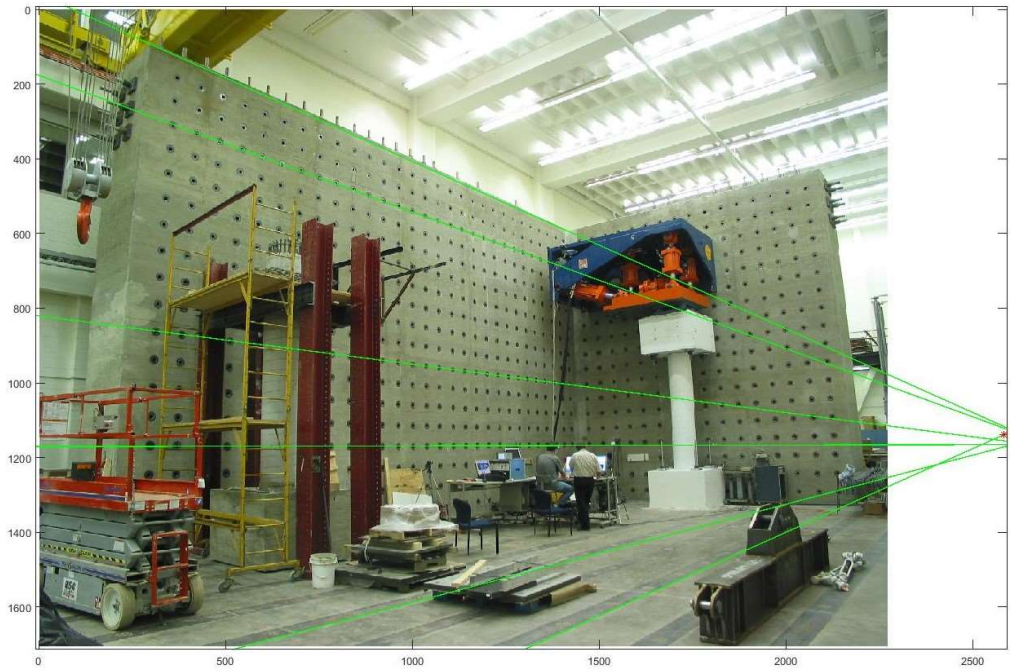
```
        pt1 = real(cross([1 0 -bx1]', lines(:, k)));
        pt2 = real(cross([1 0 -bx2]', lines(:, k)));
    else
        pt1 = real(cross([0 1 -by1]', lines(:, k)));
        pt2 = real(cross([0 1 -by2]', lines(:, k)));
    end
    pt1 = pt1/pt1(3);
    pt2 = pt2/pt2(3);
    plot([pt1(1) pt2(1)], [pt1(2) pt2(2)], 'g', 'Linewidth', 1);
end

plot(vp1(1)/vp1(3), vp1(2)/vp1(3), '*r')

axis image
axis([bx1 bx2 by1 by2]);
end
```

- Calculated Vanishing Points:

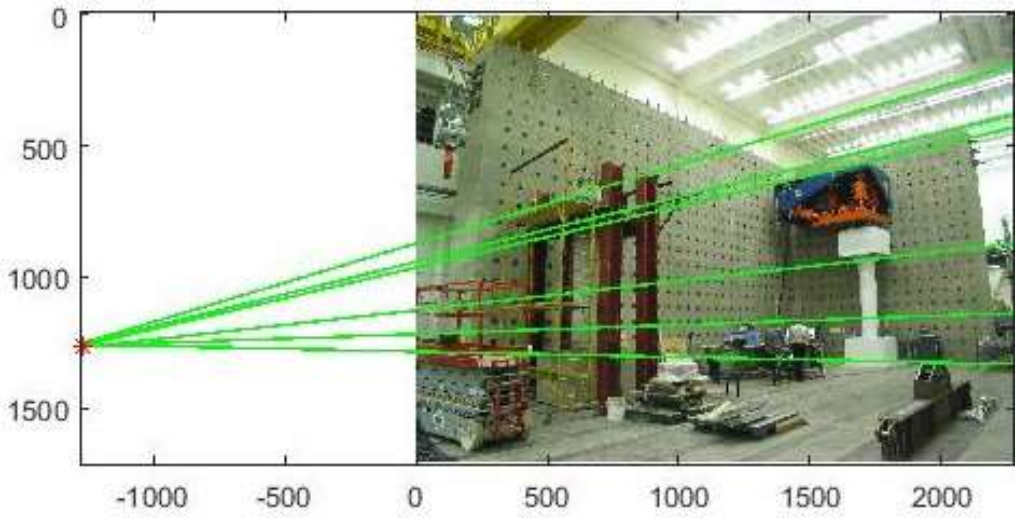
- VP1 = (2.5827e+03, 1.1392e+03)
- VP2 = (-1.2726e+03, 1.2601e+03)
- VP3 = (1.0167e+03, -8.3861e+03)

- Plotted Vanishing Points and lines for calculation:

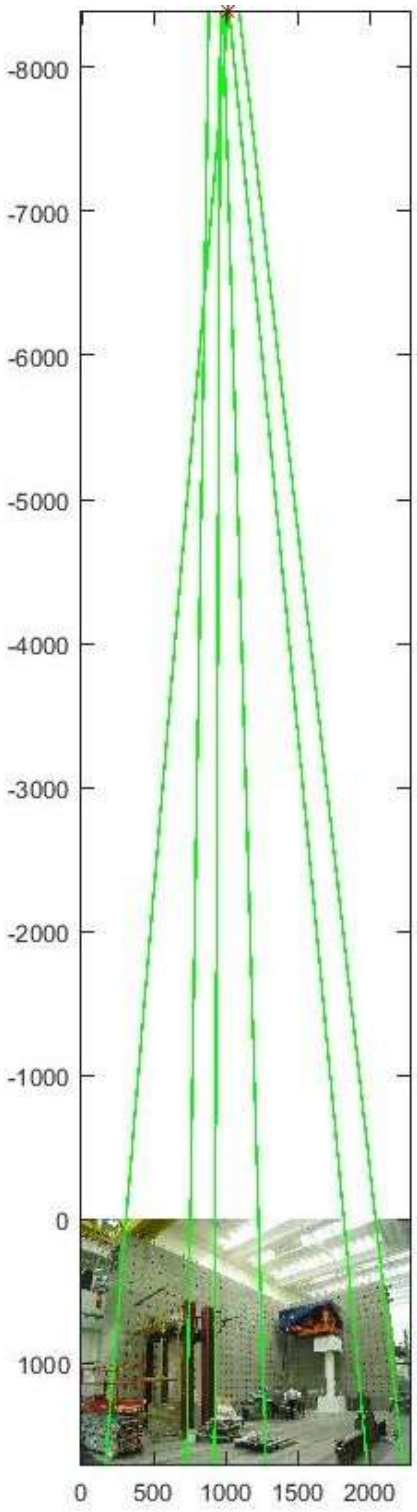
VP1:



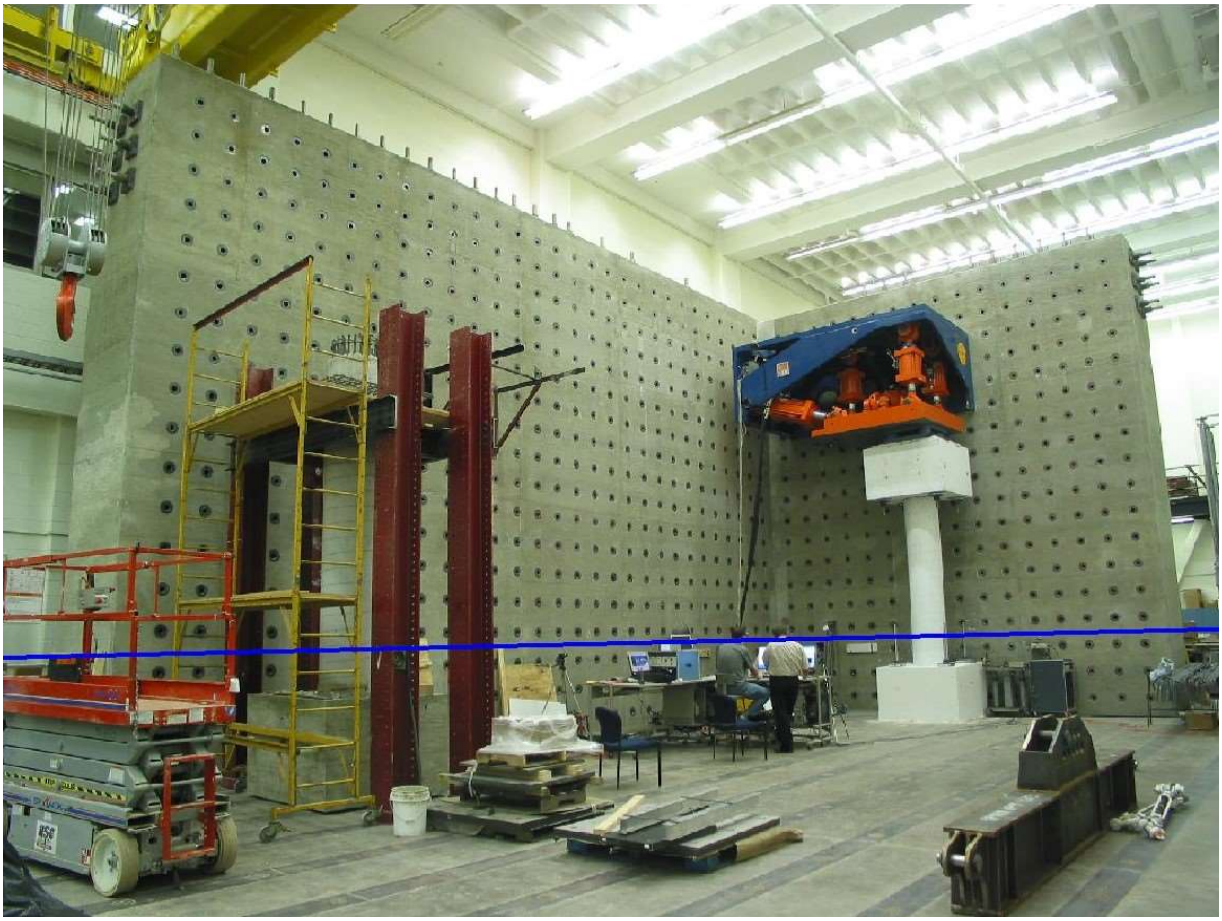
VP2:



VP3:


-8000
-7000
-6000
-5000
-4000
-3000
-2000
-1000
0
1000
0
500
1000
1500
2000

- Ground Horizon Line:



- Parameters of Ground Horizon Line:

By doing the cross product between VP1 and VP2, the parameters of the Ground Horizon Line are obtained:

GHL = cross(vp1, vp2);
GHL = [-1.3774e+12; -5.8682e+13; 7.1478e+16]
Ground Horizon Line = ***-1.3774e+12*****u** + ***-5.8682e+13*****v** + ***7.1478e+16*** **= 0**

PART B
- Method 1 (Matlab)
Calculation of Camera's Focal Length and Optical Center:

Analyzing the equation:
$pi = KRXi$ → where p*i* is the Vanishing Point *i*, K is the camera's intrinsic parameters matrix, and R is the Rotation Matrix

And understanding that:
$Xi^T Xj = 0$

We concluded that a system of 3 equations with 3 unknowns can be formed using all three Calculated Vanishing Points:

1. $p_1^T K^{-T} K^{-1} p_2 = 0$
2. $p_1^T K^{-T} K^{-1} p_3 = 0$
3. $p_2^T K^{-T} K^{-1} p_3 = 0$

Where K = $\begin{matrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{matrix}$ → where *f* is the Focal length, and (u_0, v_0) is the Optical Center.

Using MATLAB's equation solver in the following code, we stablish the Camera's Focal Length *f* and Optical Center (u_0, v_0):

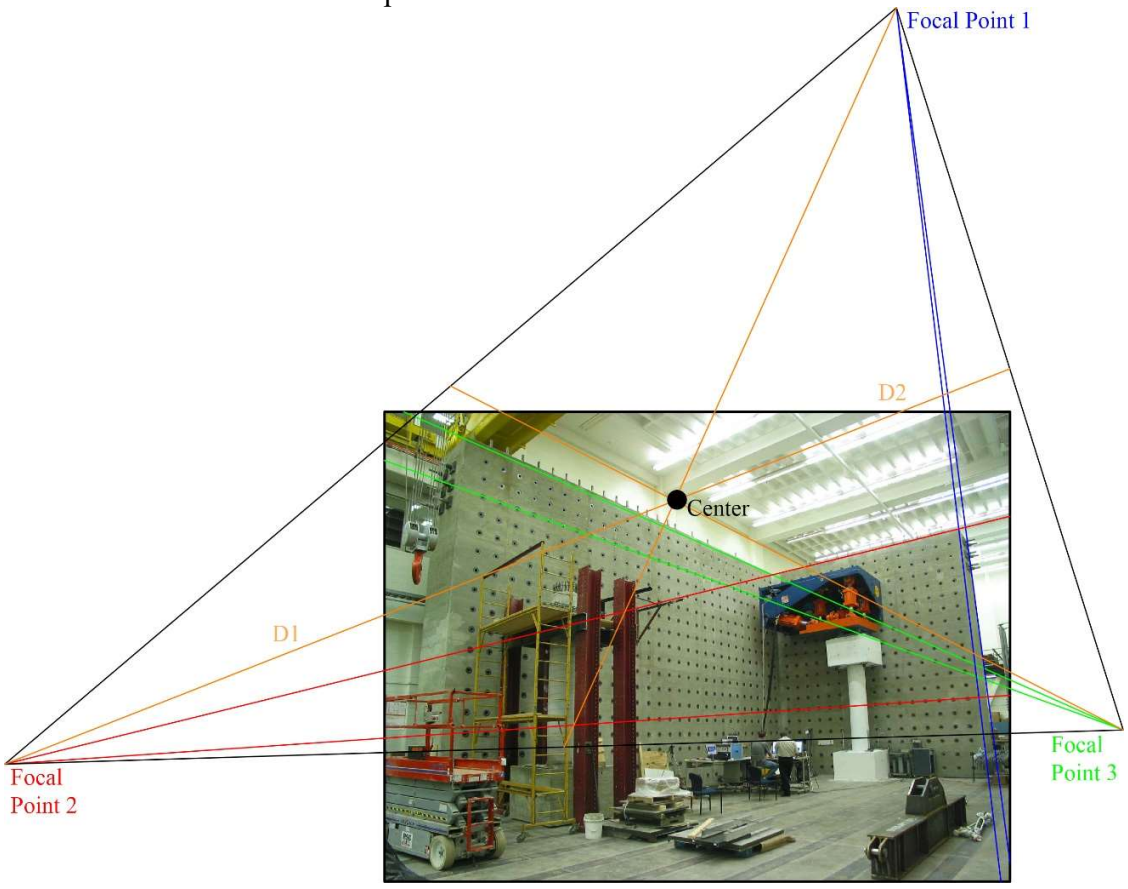
(See next page for code)

```
%% Stablishing Inputs
img = imread('NewmarkLab_UniversityofIllinois.jpg');

%% Calculating Vanishing Points
 vps = zeros(3,3);
for ii = 1:3
    vps(:,ii) = getVanishingPoint(img);
end

%% Calculating & Plotting ground Horizon Line:
 groundHline = cross(vps(:,1),vps(:,2));
% gives abc vector of the Ground Horizontal Line
figure(1),imshow(img),hold on,plot([vps(1)/vps(3),vps(4)/vps(6)],...
    [vps(2)/vps(3),vps(5)/vps(6)], 'b', 'Linewidth', 1);

%% Calculating Camera's Intrinsic Parameters
 % Using Matlab's Equation system solver
syms f u0 v0
eqns = [(vps(:,1)')*([1/f,0,-u0/f;0,1/f,-v0/f;0,0,1]')*...
    ([1/f,0,-u0/f;0,1/f,-v0/f;0,0,1])*vps(:,2) == 0,...
    (vps(:,1)')*([1/f,0,-u0/f;0,1/f,-v0/f;0,0,1]')*...
    ([1/f,0,-u0/f;0,1/f,-v0/f;0,0,1])*vps(:,3) == 0,...
    (vps(:,2)')*([1/f,0,-u0/f;0,1/f,-v0/f;0,0,1]')*...
    ([1/f,0,-u0/f;0,1/f,-v0/f;0,0,1])*vps(:,3) == 0];
K = solve(eqns,[f u0 v0]);

%% Showing Values
 % Vanishing Points
VPi = [vps(1)/vps(3),vps(2)/vps(3)];
VPj = [vps(4)/vps(6),vps(5)/vps(6)];
VPk = [vps(7)/vps(9),vps(8)/vps(9)];

focal_lenght = double(K.f);
opt_center = double([K.u0,K.v0]);
```

- Camera's Focal Length = f = 1868.1 pixels
- Camera's Optical Center = (u0, v0) = (1195.9, 785.33)

- Method 2 (Measurement and Calculation)

1. Measurements were done on **AutoCAD** by AutoDesk.
2. Pinpointing 3 vanishing points as the focal points or end points of the triangle.
3. Midpoint of each side of the triangle to pinpoint the orthocenter of the tringle. Orthocenter becomes the optical center.



- Measured

D1 = 1.1511 in. , D2 = 0.5755
Optical center
X = 0.5259 in, Y = 0.6081 in

- Calculated

Focal Point = sqrt (D1 x D2)
Focal Point = sqrt (1.1511 x 0.5755)
Focal Point = 0.8139 in. x 25.4 mm/in. = 20 mm.

PART C

Method 1 (Matlab and Calculations)

- Ground Horizon Line Illustration:

- Lines and Measurements to estimate heights:

- Estimated Heights:

Post Height = 8.5667m
Yeh Center Height = 18.8809m
Camera Height = 15.3519m

```
% Stablishing Inputs
img = imread('YehCenterUnderConstruction.jpg');
height1 = 1.346; %Height of the park meter

%% Calculating Vanishing Points
vps = zeros(3,3);
for ii = 1:3
    vps(:,ii) = getVanishingPoint2(img);
end


%% Getting required coordinates to compute height
figure, hold off, imagesc(img);
hold on
disp('Click first point or q to stop')
[X,Y] = ginput(4);
% order of selected points should be:
% 1- b0(parkmeter's base)
% 2- r0(parkmeter's top)
% 3- b(2nd element's base)
% 4- t(2nd element's top)
stop = input('Press enter to continue','s');

%% Calculating & Plotting ground Horizon Line
```

```
Horizon = cross(vps(:,1),vps(:,2));
% gives abc vector of the Ground Horizontal Line
figure(3),imshow(img),hold on,plot([vps(1)/vps(3),vps(4)/vps(6)],...
    [vps(2)/vps(3),vps(5)/vps(6)], 'b', 'Linewidth', 2);

%% Intersection between Ground Horizon Line and bottom of elements
groundLine = cross([X(1),Y(1),1]',[X(3),Y(3),1]');
V = cross(groundLine,Horizon); % intersection point (u,v,w)
Vnorm = ones(3,1); % Cartesian intersection point
Vnorm(1) = V(1)/V(3); % u/w
Vnorm(2) = V(2)/V(3); % v/w

%% Intersection onto 2nd Element
r = cross(cross(Vnorm,[X(2),Y(2),1]'),cross([X(3),Y(3),1]',...
    [X(4),Y(4),1]'));
rnorm = ones(3,1);
rnorm(1) = r(1)/r(3);
rnorm(2) = r(2)/r(3);

%% Calculation of 2nd Element's Height
height2 = ((abs(Y(3)-Y(4))*abs((vps(8)/vps(9))-rnorm(2)))/...
    (abs(Y(3)-rnorm(2))*abs((vps(8)/vps(9))-Y(4)))) * height1;

%% Plotting Lines used for calculation:
% Top line
figure(3),plot([Vnorm(1), X(2)], [Vnorm(2), Y(2)], 'g', 'Linewidth', 2);
% Bottom line
figure(3),plot([Vnorm(1), X(1)], [Vnorm(2), Y(1)], 'r', 'Linewidth', 2);
% Reference Point
figure(3),plot(rnorm(1), rnorm(2), 'r*', 'Linewidth', 1);
```

- Calculating Camera's height:

Because Yeh Center crosses through the Horizon Ground Line, we are able to estimate the camera's height by determining at what height the Ground Horizon Line intersects with Yeh Center. Knowing Yeh Center's Height = 19.7841, we use the following code to select the top and bottom part image coordinates of Yeh Center, and get the result of the difference in Y coordinates as height reference:

```
%% Estimating Camera's Height
Yeh_center_line = cross([X(3),Y(3),1]',[X(4),Y(4),1]');
% Intersection of Yeh Center and the Ground Horizon Line
camera_intersect = cross(Yeh_center_line, Horizon);
% Y Pixel difference between base and top of Yeh
deltaY_yeh = abs(Y(3) - Y(4));
% Y Pixel difference between base of Yeh and camera_intersect
deltaY_camera = abs((camera_intersect(2)/camera_intersect(3))-Y(3));

camera_height = (deltaY_camera*height2)/deltaY_yeh;
```

Method 2 (Measurement and Calculations)

- Tools and Methods used to calculate the height of YEH Center and Light Post as required by assignment 2.

1. Measurements were done on **AutoCAD** by AutoDesk. The picture was scaled to fit a 11 x 8.5 size paper, making the dimension of the picture 11 x 7.33.
2. Focal points chosen best as estimated.
3. **Cross Ratio** method used to calculate the actual heights of the required objects.
4. Numbers are rounded to the 4th decimal place in **Inches**.
5. Horizon (■) estimated by using **Focal Point Reference 1**(■)and **Focal Point Reference 2**(■).

- Known

Standard Parking Meter Height: 53 in.

- Measured

Parking Meter Height at Position 1(■): 0.3351 in.
Parking Meter Height at Position 2(■): 0.3247 in.
Parking Meter Height at Position 3(■): 0.2015 in.

Light Post Height(■): 1.9615 in.
Yeh Center Height(■): 2.9568 in.

- Calculation for Light Post Height

Parking Meter Height at Position 2 : Standard Meter Height = Measured Light Post Height : x
0.3247 : 53 = 1.9615 : x
x = (53 x 1.9615) / 0.3247
x = 320.1709 in.
x = 26.6809 ft. = (8.13 m)

- Calculations for Yeh Center (UIUC) Height

Parking Meter Height at Position 3 : Standard Meter Height = Measured Yeh Center Height : y
0.2015 : 53 = 2.9568 : y
y = (53 x 2.9568) / 0.2015
y = 777.7191 in.
y = 64.8099 ft. = (19.54 m)

- Estimated Height

Parking Meter Height = 26.6809 ft.
Yeh Center Height = 64.8099 ft.