EE 5450 Project 01: Simultaneous Monocular Calibration and Pose Estimation

David R. Mohler

February 19, 2018

1 Introduction

The following set of experiments are focused on the application of the simultaneous monocular calibration and pose estimation algorithm to a series of images of rigid objects with known dimensions and correspondence points. This algorithm is, as described by its title, capable of calibrating a single camera with a fixed focal length in the presence of known rigid objects, which enables the simultaneous discovery of the pose of the object and from this generate a three dimensional representation of that same object.

In this project we will demonstrate the generation of independent and identically distributed (IID) data samples through rejection sampling of a continuous PDF. Using this data we will then apply EM to iteratively estimate the parameters of the probability distribution function.

2 Methods and Results

The foundation of this project is based on the known characterization of a signal which is represented in the form a single continuous PDF as seen in Equation 1. It is apparent from Equation 1 that our signal is characterized as a continuous probability mixture model, consisting of three distinct mixture components. By separating these components (Figure **??**) we display the contributions of each respective distribution to the cumulative PDF.

Parameter	Value
w_1	0.5
w_2	0.3
w_3	0.2
μ_1	3.0
μ_2	10.0
σ_1^2	1.0
σ_2^2	4.0
λ	2.0

Table 1: True Distribution Parameters

Using the given values, seen in Table 1, we display the separated components of the probability mixture. Our mixture model is composed of two separate Gaussian distributions, each with their own respective means and variances, and a single exponential distribution which is characterized by the λ parameter, which can be interpreted as the inverse of the mean of the exponential distribution.

$$p(x) = w_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} e^{-\frac{1}{2}\frac{(x-\mu_1)^2}{\sigma_1^2}} + w_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} e^{-\frac{1}{2}\frac{(x-\mu_2)^2}{\sigma_2^2}} + w_3 \lambda e^{-\lambda x} \tag{1}$$

To maintain the constraint that $\int p(x)dx = 1$, each mixture component is assigned a weight such that $\sum_i w_i = 1$. The true distribution is constructed such that $w_1 = 0.5$, $w_2 = 0.3$, and $w_3 = 0.2$. The results of the weighted mixture that constitutes the continuous distribution $p(x)$ can be seen in Figure **??**.

2.1 Corrupted Correspondence Points

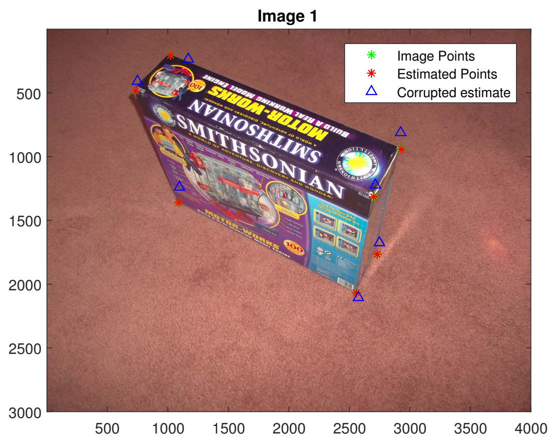

Figure 1: Estimated image coordinates

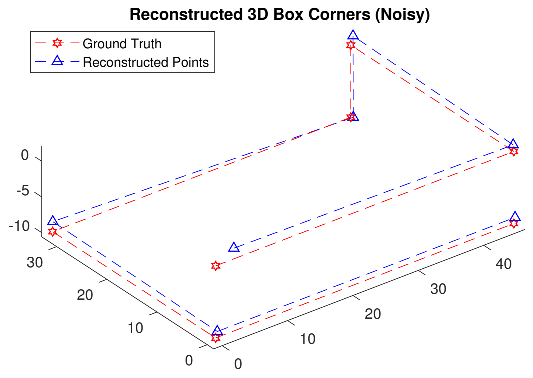

Figure 2: Reconstructed box relative to ground truth

2.2 Improper Dimensions

3 Conclusions

In this project we have demonstrated two major methods critical to estimation and uncertainty quantification. The first being a method to sample from any arbitrary closed functional form of a continuous probability distribution, and the second being the application of the Expectation Maximization algorithm to a probability mixture model. For our implementation the rejection sampling method is essential to the proper operation of the EM algorithm. Rejection sampling provides a simple method for artificial data generation. By varying the number of data points that we wish to sample from our known functional form of the distribution we are able to view the accuracy of the sampled PDF relative to the

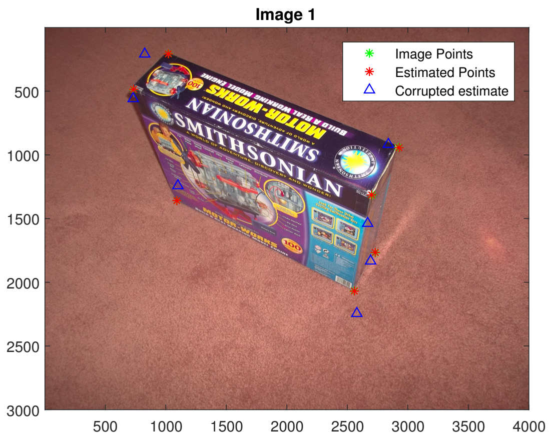

Figure 3: Estimated image coordinates (unique coordinate component noise)

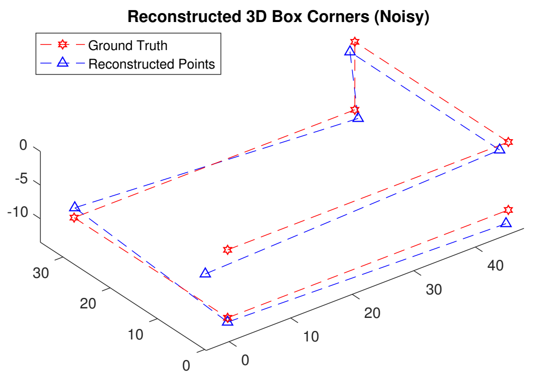

Figure 4: Reconstructed box relative to ground truth

true curve. It is noted that the number of data points required to generate an "accurate" curve is application and computational burden dependent. For the purposes of this project we found that samples ranging in the low hundreds did not provide enough data to create a decent approximation, however, beyond 1000 samples we were able to obtain good results with low margins of error. From this, we note that balancing between excessive computation and accuracy requirements lacks a singular hard threshold and should be approached with the specific design parameters in mind.

The Expectation Maximization algorithm has proven itself to be a fairly robust tool for the approximation of distribution parameters. When provided with a sufficient number of samples and reasonable initializations of the estimated parameters EM is,on average, capable of producing results with single digit error precision (percentage). However, it is not entirely immune to failure. The algorithm diverges in cases where it is initially assumed that the weight of any given component is zero, or the variance of any given Gaussian component is initialized with a zero value. Similarly, while not entirely failing, the margin of error that occurs in the impoverished case of too few samples provided to the algorithm is vastly larger on average. This error margin was observed to reach a mean of as high as nearly 41% error, roughly 8 times as large as that of the trials performed with a sufficient number of samples. We have also demonstrated that the algorithm tends to provide better performance when faced with distributions that are, for the most part distinct and separate. Whereas when tasked with a distribution that is heavily mixed and the components are largely indistinguishable from one another, the performance is much poorer, even with strong initialization parameters.

A Code Listings

Listing 1: Top level implementation for Monocular Calibration and Pose Estimation

```
%David R Mohler
%EE-5450: Topics in Robotics
%Project 1
%Spring 2018

clear
close all
%read in the images
i1=imread('Images\cvClass␣023.jpg','jpg');
i2=imread('Images\cvClass␣026.jpg','jpg');
i3=imread('Images\cvClass␣024.jpg','jpg');
i4=imread('Images\cvClass␣028.jpg','jpg');
lbox=45.6; %length of box
hbox=32.5; %height of box
wbox=10.1; %width of box
n=7;

%use either the first set of commands (to initially mark
   [+]correspondence
%points manually) or the load command  (to read in previous
   [+]correspondence
%points)
%    X1=[1 1];X2=[1 1];%must start with an initial point; this
   [+]will be removed later
%    [X1,X2]=cpselect(i1,i2,X1,X2,'Wait',true);
%     save motorBoxCorners23_26 X1 X2 %save the correspondence
   [+]points you just found

%load in correspondence points
load motorBoxCorners23_26.mat
load motorBoxCorners24_28.mat

[mc,nc]=size(X1); %need if changing # of Correspon. pts.
X1=X1(2:n+1,:); %remove initial point, it is not good data
X2=X2(2:n+1,:);
[mc,nc]=size(X1);

x1pixmat=[X1'
    ones(1,mc)];  %convert the points to homogeneous coordinates
x2pixmat=[X2'
    ones(1,mc)];  %convert the points to homogeneous coordinates
```

```
x3pixmat=[X3'
    ones(1,mc)];  %convert the points to homogeneous coordinates
x4pixmat=[X4'
    ones(1,mc)];  %convert the points to homogeneous coordinates

%Corruption of data points with gaussian noise
x1pixcor = x1pixmat;
%corruption of each coordinate in each point seperately
% for i = 1:mc
%   x1pixcor(1,i) = x1pixmat(1,i)+100*randn(1,1);
%   x1pixcor(2,i) = x1pixmat(2,i)+100*randn(1,1);
%   x1pixcor(3,i) = x1pixmat(3,i)+100*randn(1,1);
% end

%Corrupting all coordinates equally across correspondence points
%(maintains shape, but is shifted in image plane)
x1pixcor(1,:) = x1pixmat(1,:)+100*randn(1,1);
x1pixcor(2,:) = x1pixmat(2,:)+100*randn(1,1);
x1pixcor(3,:) = x1pixmat(3,:)+100*randn(1,1);

%enter object coordinates for the first 7 corners
Xomat=[0 lbox lbox lbox 0 0 lbox
       0 0 hbox hbox hbox 0 0
       0 0 0 -wbox -wbox -wbox -wbox]; %object coords of the four
          [+]corners
Xoh = [Xomat; ones(1,mc)]; %Homogeneous object coordinates
%find calibration matrices
[gest1qr,lambda1qr,K1]=monoPoseQR(Xomat,x1pixmat); %find K, depth,
   [+] and g
Rest1qr=gest1qr(1:3,1:3);Test1qr=gest1qr(1:3,4); %Extraction of R
   [+]and T

[gest1cor,lambda1cor,K1cor]=monoPoseQR(Xomat,x1pixcor); %find K,
   [+]depth, & g
Rest1cor=gest1cor(1:3,1:3);Test1cor=gest1cor(1:3,4); %Extraction
   [+]of R and T

[gest2qr,lambda2qr,K2]=monoPoseQR(Xomat,x2pixmat);
Rest2qr=gest2qr(1:3,1:3);Test2qr=gest2qr(1:3,4);

[gest3qr,lambda3qr,K3]=monoPoseQR(Xomat,x3pixmat);
Rest3qr=gest3qr(1:3,1:3);Test3qr=gest3qr(1:3,4);

[gest4qr,lambda4qr,K4]=monoPoseQR(Xomat,x4pixmat);
Rest4qr=gest4qr(1:3,1:3);Test4qr=gest4qr(1:3,4);
```

```
PI1 = [K1*Rest1qr K1*Test1qr];
PI1cor = [K1cor*Rest1cor K1cor*Test1cor];
PI2 = [K2*Rest2qr K2*Test2qr];
PI3 = [K3*Rest3qr K3*Test3qr];
PI4 = [K4*Rest4qr K4*Test4qr];

%Estimated pixel coordinates of correspondence points
x1pixest = zeros(size(x1pixmat));
x1pixestcor = zeros(size(x1pixmat));
x2pixest = zeros(size(x2pixmat));
x3pixest = zeros(size(x3pixmat));
x4pixest = zeros(size(x4pixmat));
for i = 1:mc
    x1pixest(:,i) = (PI1*Xoh(:,i))/lambda1qr(i);
    x1pixestcor(:,i) = (PI1cor*Xoh(:,i))/lambda1cor(i);
    x2pixest(:,i) = (PI2*Xoh(:,i))/lambda2qr(i);
    x3pixest(:,i) = (PI3*Xoh(:,i))/lambda3qr(i);
    x4pixest(:,i) = (PI4*Xoh(:,i))/lambda4qr(i);
end

%Reconstruction of Object Coordinates from estimates
Xoest1 = zeros(size(Xomat));
Xoest1cor = zeros(size(Xomat));
Xoest2 = zeros(size(Xomat));
Xoest3 = zeros(size(Xomat));
Xoest4 = zeros(size(Xomat));
for i = 1:mc
   Xoest1(:,i) = Rest1qr'*inv(K1)*(lambda1qr(i)*x1pixest(:,i)-K1*
      [+]Test1qr);
   Xoest1cor(:,i) = Rest1qr'*inv(K1)*...
                        (lambda1qr(i)*x1pixestcor(:,i)-K1*Test1qr)
                           [+];
   Xoest2(:,i) = Rest2qr'*inv(K2)*(lambda2qr(i)*x2pixest(:,i)-K2*
      [+]Test2qr);
   Xoest3(:,i) = Rest3qr'*inv(K3)*(lambda3qr(i)*x3pixest(:,i)-K3*
      [+]Test3qr);
   Xoest4(:,i) = Rest4qr'*inv(K4)*(lambda4qr(i)*x4pixest(:,i)-K4*
      [+]Test4qr);
end

%Images and plotting
figure(1)
clf
image(i1);
```

```
hold on
plot(X1(:,1)+j*X1(:,2),'g*')
scatter(x1pixest(1,:),x1pixest(2,:),'r*') %Pure pixel estimates
scatter(x1pixestcor(1,:),x1pixestcor(2,:),'b^') %Noise corrupted
   [+]pixel est.
legend('Image␣Points','Estimated␣Points','Corrupted␣estimate')
title('Image␣1')
hold off

figure(2)
clf
image(i2)
hold on
plot(X2(:,1)+j*X2(:,2),'g*')
scatter(x2pixest(1,:),x2pixest(2,:),'r*')
legend('Image␣Points','Estimated␣Points')
title('Image␣2')
hold off


figure(3)
clf
image(i3)
hold on
plot(X3(:,1)+j*X3(:,2),'g*')
scatter(x3pixest(1,:),x3pixest(2,:),'r*')
legend('Image␣Points','Estimated␣Points')
title('Image␣3')
hold off

figure(4)
clf
image(i4)
hold on
plot(X4(:,1)+j*X4(:,2),'g*')
scatter(x4pixest(1,:),x4pixest(2,:),'r*')
legend('Image␣Points','Estimated␣Points')
title('Image␣4')
hold off

figure(5)
hold on
clf
pts=1:7;
plot3(Xomat(1,pts),Xomat(2,pts),Xomat(3,pts),'--rh')
```

```
axis equal
title('3D␣Box␣Corners')
hold off

figure(6)
hold on
clf
pts=1:7;
plot3(Xoest1(1,pts),Xoest1(2,pts),Xoest1(3,pts),'--g*')
axis equal
title('Reconstructed␣3D␣Box␣Corners')
hold off

%Reconstruction of truth and noise boxes
figure(7)
clf
pts=1:7;
hold on
plot3(Xomat(1,pts),Xomat(2,pts),Xomat(3,pts),'--rh')
plot3(Xoest1cor(1,pts),Xoest1cor(2,pts),Xoest1cor(3,pts),'--b^')
view(3)
axis equal
title('Reconstructed␣3D␣Box␣Corners␣(Noisy)')
legend('Ground␣Truth','Reconstructed␣Points')
hold off

figure(1)
```

Listing 2: QR Decomp Based Algorithm Implementation

```
function [g,lambda,K]=monoPoseQR(Xomat,xpixh)
%Function to find the camera pose (g), depth to object points (
   [+]lambda), and
%camera calibration matrix (K)

n = size(Xomat,2);

%convert object coords to homogeneous coordinates
Xoh = [Xomat; ones(1,n)];
I3 = eye(3);
e3 = [0 0 1]';

N = zeros(3*n,12);
j = 1;
for i  = 1:n
    N(j:j+2,:) = kron(Xoh(:,i)',I3)-kron(Xoh(:,i)',(xpixh(:,i)*e3
        [+]'));
    j = j+3;
end

[~,~,V] = svd(N);
PIs = V(:,end); %Extract LSE of stacked PI
PI = reshape(PIs,[3 4]); %Unstack PI vector
[alf,K,R] = qrCommute(PI(:,1:3)); %use QR decomp to find K,R, and
   [+]scale
T = (inv(K)/alf)*PI(:,4); %Calculate translation
PIest = [K*R K*T];
g = [R T;0 0 0 1]; %Contruct pose matrix
lambda = e3'*PIest*Xoh; %Extract depth
```

Listing 3: Rearranged QR Decomposition (Credit: Dr. John McInroy)

```
function [alf, K,R ] = qrCommute( M )
%function [alf, K,R ] = qrCommute( M )

%modified qr decomposition.  It commutes the order
%of the rotation matrix and the upper triangular matrix.
%
%The decomposition is returned such that alf*K*R = M, where
%alf is a scalar, K is upper triangular, and R is a special
   [+]orthogonal
%matrix.
%
%In addition, it makes the diagonal of K positive, with the last
   [+]entry
%of K, K(n,n), equal to one.  This factorization is ideal for
   [+]uncalibrated
%camera factorization problems.  If it is not possible, an error
   [+]is
%returned.
%
%J. McInroy, 2/4/10

[m,n]=size(M);
E=zeros(n,n);
E2=zeros(n,n);
for i=1:n,
    E(i,n+1-i)=1;
end
[Q,Rp]=qr(M'*E);
Kn=E*Rp';
Rn=Q';
```

```
alf=Kn(n,1);
for i=1:n,
    if Kn(i,n+1-i)<0,
        E2(n+1-i,i)=-1;
    else
       E2(n+1-i,i)=1;
    end
end
R=E2'*Rn;
K=Kn*E2/alf;
if det(R)<0,
    R=-R;
    alf=-alf;
end
if K(n,n)<0,
    K=-K;
    alf=-alf;
end
end
```

References