Advanced Control Laboratory (085705)
ROS #4: Triangulation & 3D Reconstruction

Tom Smadar 203374061

Daniel Engelsman 300546173

1 Repeat Lab 3

1.1 Matlab Code + Explanations

(a) Weve selected image number 96 and loaded it into matlab:

```
%% Read image from ROSbag
ImgNr = 28;
bag = rosbag('rosbag_ac.bag')
bagselect1 = select(bag,'Topic','/camera/rgb/image_raw/compressed')
imgMsg = readMessages(bagselect1);
C = imgMsg{ImgNr}.Data;
fileID = fopen('test.jpg','w');
fwrite(fileID,C);
fclose(fileID);
imshow('test.jpg')
```

(b) Weve not used Ground Truth, but odometry reading instead.
Weve defined the Transformations between Global and Robot frame,
and between Robot and Camera frame.

```
%% Calculate landmark location
bagselect2 = select(bag,'Topic','/RosAria/pose');
poseMsg = readMessages(bagselect2);
P = poseMsg{ImgNr}.Pose.Pose.Position;
alpha = poseMsg{ImgNr}.Pose.Pose.Orientation;
quat = [alpha.X alpha.Y alpha.Z alpha.W];
alpha = quat2eul(quat);
alpha = alpha(3);
Rob = [P.X P.Y 0]'; % Robot Position (Global)
Tg2r = angle2dcm(0,0,alpha,'XYZ');
Tg2r(:,4) = Tg2r*(-Rob);
Tg2r(4,:) = [0 0 0 1];
Tr2c = angle2dcm(0,0.5*pi,-0.5*pi,'XYZ'); % Rotation from Global to Camera
Tr2c(:,4) = [0.08 0.08 -0.15]'; % Translation from Camera to Robot in
    Camera Frame
```

```
Tr2c(4,:) = [0 0 0 1];
% Landmarks:
Landmark1 = [2.656 1.83 0.3]';
Landmark2 = [3.794 1.495 0.3]';

(c) And derived their 2D coordinates from the image:
%% Derive coordinates:
figure(1)
imshow('test.jpg')
hold on
[Px,Py]= ginput;
tru = scatter(Px,Py,80,'+','r');

(d) Projecting the 3D landmarks into the camera frame:
%% Project Coordinates:
K = [529.777 0 322;0 532.012 246;0 0 1]; % Calibration Matrix

% Projection Matrix:
Tg2c =Tr2c*Tg2r;
T = Tg2c(1:3,:);
M = K*T;

% Homogenous:
Landmark1(4) = 1;
Landmark2(4) = 1;
L(:,1) = M*Landmark1;
L(:,1) = L(:,1)./L(3,1);
L(:,2) = M*Landmark2;
L(:,2) = L(:,2)./L(3,2);
scatter(L(1,:),L(2,:),80,'+','b');
set(gca,'Xlim',[0 900])

%% Re-Projection Error:
errx = Px' - L(1,:);
erry = Py' - L(2,:);
landErr1 = norm([errx(1) erry(1)])
landErr2 = norm([errx(2) erry(2)])
```

1.2 Results - Fixed Images

Chosen Image where both landmarks are clearly visible, and then launching the algorithm :

Figure 1: Image # 96

We can see that the projected points (marked blue) are quite close to the manually selected ones (marked red). There is a height difference that might pertain to the camera calibration matrix.

Figure 2: Image # 96 after Projected crosses

The Re-projection error for the landmarks :

Landmark	Error [px]
Landmark 1	16
Landmark 2	6

Figure 3: Error Table

2 Image correspondence for the entire database

2.1 Landmarks Visibility

#	L1	L2	#	L1	L2	#	L1	L2	#	L1	L2
1	0	1	31	1	1	61	0	0	91	0	0
2	0	1	32	1	1	62	0	0	92	0	0
3	0	1	33	1	1	63	0	0	93	0	0
4	0	1	34	1	1	64	0	0	94	0	0
5	0	1	35	0	1	65	0	0	95	0	1
6	0	1	36	0	1	66	0	0	96	1	1
7	0	1	37	0	0	67	0	0	97	1	1
8	0	1	38	0	0	68	0	0	98	1	1
9	0	1	39	0	0	69	0	0	99	1	1
10	0	1	40	0	0	70	0	0	100	1	1
11	0	1	41	0	0	71	0	0	101	1	1
12	0	1	42	0	0	72	0	0	102	1	1
13	0	1	43	0	0	73	0	0	103	1	1
14	0	1	44	0	0	74	0	0	104	1	1
15	0	1	45	0	0	75	0	0	105	1	1
16	0	1	46	0	0	76	0	0	106	1	1
17	0	1	47	0	0	77	0	0	107	1	1
18	0	1	48	0	0	78	0	0	108	1	1
19	0	1	49	0	0	79	0	0	109	1	1
20	0	1	50	0	0	80	0	0	110	1	1
21	0	0	51	0	0	81	0	0	111	1	1
22	0	1	52	0	0	82	0	0	112	1	1
23	1	1	53	0	0	83	0	0	113	1	1
24	1	1	54	0	0	84	0	0	114	1	1
25	1	1	55	0	0	85	0	0	115	1	1
26	1	1	56	0	0	86	0	0	116	1	1
27	1	1	57	0	0	87	0	0	117	1	1
28	1	1	58	0	0	88	0	0	118	1	1
29	1	1	59	0	0	89	0	0	119	1	1
30	1	1	60	0	0	90	0	0	120	1	1
									121	1	1

Figure 4: Landmarks Visibility (1 for visible and 0) from Robot Position

2.2 Graph Presentation

```
load('Img_Corr'); N_msgs = 121;
% ---------- 2.b Estimated Path of the Robot ---------- %
bag = rosbag('rosbag_ac.bag');
bag_GT  = select(bag,'Topic','/Robot_1/pose');
all_GT_Msgs = readMessages(bag_GT);
bag_est = select(bag,'Topic','/RosAria/pose');
all_est_Msgs = readMessages(bag_est);

%% ---------- Extract Paths out of Files ---------- %
for i=1:N_msgs
    c_est = all_est_Msgs{i}.Pose.Pose.Position;
    [P_est(i,1), P_est(i,2)] = deal( c_est.X, c_est.Y );
    c_GT = all_GT_Msgs{i}.Pose.Position;
    [P_GT(i,1), P_GT(i,2)] = deal( c_GT.X, c_GT.Y );
end

% ---------- Sorting and fitting Plots ---------- %
```

```
hold on; pbaspect([1 1 1]);
plot(P_est(:,1), P_est(:,2), 'linewidth', 2.5 );
pos_Deviat = ones(N_msgs,1)*P_GT(1,1:2);
P_GT_fix = P_GT - pos_Deviat;
Rot_mocap = angle2dcm( 0, 0, -deg2rad(90.6), 'XYZ');
P_GT_fix = transpose(Rot_mocap*[P_GT_fix zeros(N_msgs,1)]');
plot(P_GT_fix(:,1), P_GT_fix(:,2), 'linewidth', 2.5 )

scatter(Lnd_1(1), Lnd_1(2), 300, 'k*');
scatter(Lnd_2(1), Lnd_2(2), 300, 'k*');

for i=1:N_msgs
    if Img_Corr(i,1) == 1 && Img_Corr(i,2) == 1
        plot([P_est(i,1) Lnd_1(1)], [P_est(i,2) Lnd_1(2)], 'g--', '
            linewidth', 1 );
        plot([P_est(i,1) Lnd_2(1)], [P_est(i,2) Lnd_2(2)], 'g--', '
            linewidth', 1 );
    elseif Img_Corr(i,1) == 1 && Img_Corr(i,2) == 0
        plot([P_est(i,1) Lnd_1(1)], [P_est(i,2) Lnd_1(2)], 'g--', '
            linewidth', 1 );
    elseif Img_Corr(i,2) == 1 && Img_Corr(i,1) == 0
        plot([P_est(i,1) Lnd_2(1)], [P_est(i,2) Lnd_2(2)], 'g--', '
            linewidth', 1 );
    end
end

legend('Estimated Pioneer Path','Ground Truth','Landmark #1','Landmark #2')
title('Estimated Vs. Ground-Truth of Robot Path','FontSize','20')
```

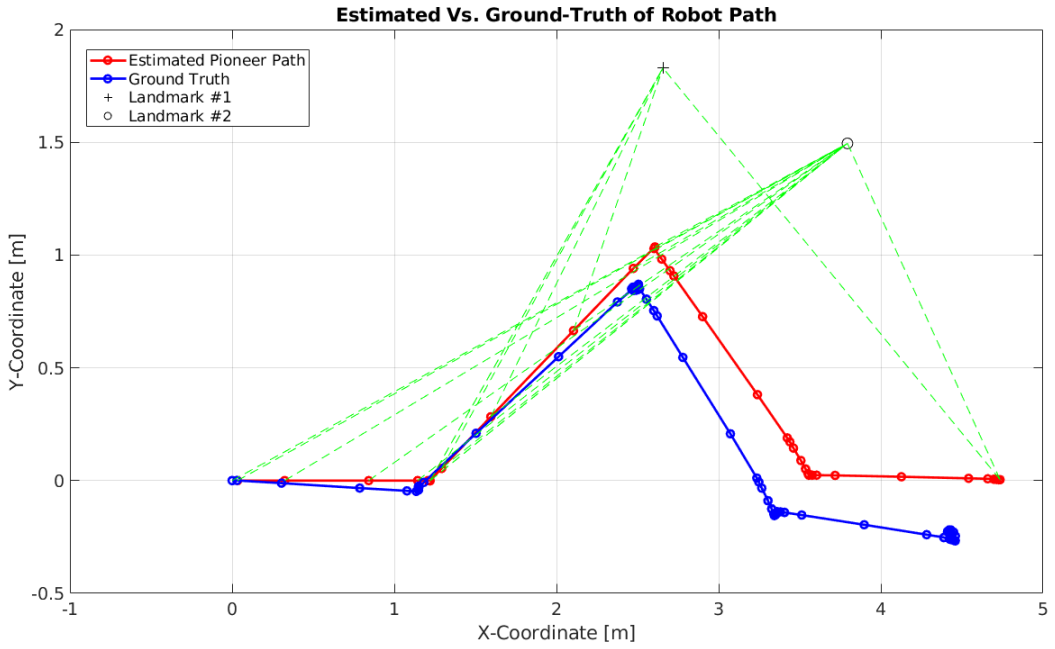

Figure 5: Landmarks Correspondence to Robot Position

3 Auxiliary Transformation Matrix

```
function T_global2cam = GetTranslationMat(FrameIndx, Database, odomORmocap)
global pos_Deviat

if odomORmocap == 1
    % ------- Fitting odometry to Robot Co. system -------- %
    bag_pos = select(Database, 'Topic', '/RosAria/pose');
    pos_msgs = readMessages(bag_pos);
    R_i = pos_msgs{FrameIndx}.Pose.Pose.Position;
    R = [R_i.X R_i.Y 0]; R = R';
    Or = pos_msgs{FrameIndx}.Pose.Pose.Orientation;
    angle = quat2eul([Or.X Or.Y Or.Z Or.W]);
    psi = angle(3);
else % odomORmocap == 0
    % --------- Fitting mocap to Robot Co. system --------- %
    bag_pos = select(Database, 'Topic', '/Robot_1/pose');
    pos_msgs = readMessages(bag_pos);
    R_i = pos_msgs{FrameIndx}.Pose.Position;
    R_i = [R_i.X R_i.Y] - pos_Deviat;          % Initial Postion Deviation
    Rot_mocap = angle2dcm( 0, 0, -deg2rad(90.6), 'XYZ' );
    R = Rot_mocap*[R_i(1) R_i(2) 0]';
    Or = pos_msgs{FrameIndx}.Pose.Orientation;
    angle = quat2eul([Or.X Or.Y Or.Z Or.W]);
    psi = angle(3) + deg2rad(90.6);            % Initial Angular Deviation
end

% ------------------ Project on Camera Frame ------------------ %
% ----------------- Transform : Camera to Robot ----------------- %
R_r2c = angle2dcm(pi/2,0,0)*angle2dcm(0,0,pi/2);
t_r2c = [0.08 0.08 -0.15]';
T_r2c = [R_r2c' t_r2c; 0 0 0 1];
% ----------------- Transform : World to Camera ----------------- %
R_g2r = angle2dcm(0, 0, psi,'xyz');
t_g2r = R_g2r*(-R);
T_g2r = [R_g2r t_g2r; 0 0 0 1];
% -------- Output : Transformation Product -------- %
T_global2cam = T_r2c*T_g2r;
```

Notes :

- the mo-cap is turned by +90.6 ° around zaxis, in reference to the robots coordinate system.
- The function uses few global variables for simplicity reasons.

4 Triangulate a landmark

Matlab Code

```
%% ---------------------- 4. Triangulate a Landmark ---------------------- %
% ---------------- Extracting Images from Rosbag ---------------- %
bag = rosbag('rosbag_ac.bag');
bag_imgs  = select(bag,'Topic','/camera/rgb/image_raw/compressed');
all_imgs = readMessages(bag_imgs);

% ------------- Triangulation - 3 Images with L ---------------- %
N_i = [1  32  96];

for i=1:length(N_i)
    figure(i); imshow( readImage( all_imgs{N_i(i)} ) );
    [pix_i(i,1), pix_i(i,2)]  = ginput;
end

[pix_1  pix_2  pix_3] = deal( pix_i(1,:), pix_i(2,:), pix_i(3,:) );
u = pix_i(:,1); v = pix_i(:,2);
```

4.1 Choose 3 different images in which L2 visible

Figure 6: Set of Images [1 32 96]

4.2 Manual Extraction

```
1     pix_1
2     pix_2
3     pix_3
```

```
Command Window
  pix_1 =
    101.0000  208.0000
  pix_2 =
    399.0000  202.0000
  pix_3 =
    420.0000  163.0000
```

Figure 7: Obtained Pixels of the manual Extraction

4.3 Reconstruction at both Methods

Matlab Code

```
%% ---------- Reconstruction using Odometry / Mocap Triangulation ----------
   %
flag = 1;                                   % Set for both methods
for j=1:2                                   % 0 - Mo-cap, 1 - Odometry
    clear A b M
    for i=1:length(N_i)
        % ---------- Obtain SPECIFIC Transformation Matrix ---------- %
        T_w2c = GetTranslationMat(N_i(i), bag, j-flag);
        M = K*T_w2c(1:3,:);                 % M : Projection Matrix
        % ---------- Form A x = b linear system ---------- %
        b((2*i-1):(2*i),1)=[-u(i)*M(3,4) + M(1,4); -v(i)*M(3,4) + M(2,4)];

        A( (2*i-1):(2*i),:) = [u(i)*M(3,1)-M(1,1), u(i)*M(3,2)-M(1,2), u(i)
            *M(3,3)-M(1,3);
             v(i)*M(3,1)-M(2,1),   v(i)*M(3,2)-M(2,2),   v(i)*M(3,3)-M(2,3)];
    end

    if j-flag == 0
        X_Mo_cap = inv(A'*A)*A'*b
        err_Mo_cap = norm( Lnd_2 - X_Mo_cap )
    else
        X_Odom = inv(A'*A)*A'*b
        err_Odom    = norm( Lnd_2 - X_Odom    )
    end
end
```

Note :

- The code refers to both methods (odometry / mo-cap) depends on the flag value.
- The u and v pixels as seen above, have been extracted from Pix1, Pix2, Pix3 matrix arrays.

4.4 3D Presentation of Triangulation points

```
%% ---------- 3D Presentation of Triangulation points ---------- %
hold on; grid on; view([-39 34]);
plot3([3.794 3.794], [1.495 1.495], [0 0.5], 'k--', 'LineWidth', 1.5);
scatter3(X_Mo_cap(1), X_Mo_cap(2), X_Mo_cap(3), 'MarkerEdgeColor','k','
    MarkerFaceColor',[0 .75 .75] );
scatter3(X_Odom(1), X_Odom(2), X_Odom(3), 'MarkerEdgeColor','k','
    MarkerFaceColor', 'r' );
scatter3(Lnd_2(1), Lnd_2(2), Lnd_2(3), 'MarkerEdgeColor','k','
    MarkerFaceColor', 'b' );

ind(1) = xlabel('X axis'); ind(2) = ylabel('Y axis');
xlim([0 5]); ylim([0 3]); zlim([0 0.5]);
ind(4) = title('Triangulation of 3D points in ANPL');
ind(5) = legend('Mo-cap', 'Odometry', 'True');
set(ind, 'Interpreter', 'latex', 'fontsize', 16 );
```

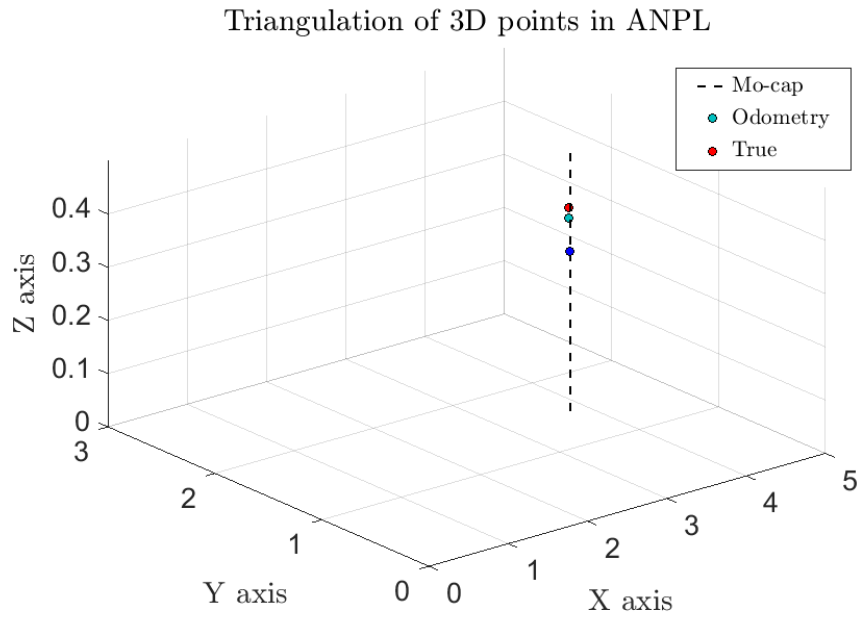

Figure 8: 3D Scatter of Triangulation points after Reconstruction

4.5 3D location error of L2 for both Methods

```
X_Mo_cap =
    3.9288
    1.6008
    0.3480
err_Mo_cap =
    0.1780

X_Odom =
    4.0871
    1.7201
    0.3509
err_Odom =
    0.3731
```

Figure 9: Absolute Error between Reconstructed Landmarks to Ground Truth

5 The importance of diversity

5.1 3D location error of L2 for both Methods

```
%% ---------------- 5.a.b The importance of diversity ---------------- %
% ------------------------ Load Data Automatically ------------------------ %
r1 = [5 18 20]; r2 = [24 26 27]; r3 = [30 32 34]; r4 = [96 100 115];
N_i = [r1; r2; r3; r4];
load('u_5'); load('v_5');

flag = 0;                                   % 0 - Mo-cap, 1 - Odometry
for k=1:4
    for i=1:3
        % ---------- Manual pixel input : used only once ---------- %
%         figure(i); imshow( readImage( allMsgs{N_i(k,i)} ) );
%         [u(k,i), v(k,i)] = ginput;
        % ---------- Obtain SPECIFIC Transformation Matrix -------- %
        T_w2c = GetTranslationMat(N_i(k,i), bag, flag);
        M = K*T_w2c(1:3,:);                 % M : Projection Matrix
        run_i(3*(k-1)+i) = N_i(k,i);
        % ---------------- Form A x = b linear system ---------------- %
        b( (2*(i+k-1)-1):(2*(i+k-1)),1) = [-u(k,i)*M(3,4) + M(1,4); -v(k,i)
            *M(3,4) + M(2,4) ];

        A( (2*(i+k-1)-1):(2*(i+k-1)),:) = [u(k,i)*M(3,1)-M(1,1),  u(k,i)*M
            (3,2)-M(1,2),  u(k,i)*M(3,3)-M(1,3);
             v(k,i)*M(3,1)-M(2,1),  v(k,i)*M(3,2)-M(2,2),  v(k,i)*M(3,3)-M
                 (2,3)];

        if flag == 0
            X_Mo_cap_i(:,3*(k-1)+i) = inv(A'*A)*A'*b;
            err_Mo_cap(3*(k-1)+i) = norm( Lnd_2 - X_Mo_cap_i );
        else
            X_Odom_i(:,3*(k-1)+i) = inv(A'*A)*A'*b;
            err_Odom(3*(k-1)+i)   = norm( Lnd_2 - X_Odom_i   );
        end
    end

    if flag == 0
        X_Mo_cap = inv(A'*A)*A'*b;
        err_Mo_cap(k) = norm( Lnd_2 - X_Mo_cap );
    else
        X_Odom = inv(A'*A)*A'*b;
        err_Odom(k) = norm( Lnd_2 - X_Odom );
    end
end
```

```
hold on; X_norm_O = zeros(1,12); X_norm_M = X_norm_O;
for i=1:length(run_i)
    if flag == 0
        X_norm_M(i)=(sum(run_i(1:i))*sum(err_Mo_cap(1:i)))/sum(run_i(1:i));
    else
        X_norm_O(i)=(sum(run_i(1:i))*sum(err_Mo_cap(1:i)))/sum(run_i(1:i));
    end
end
```

Note :

- As requested, frames increase monotonously along time.
- After first execution, pixels values were saved aside.

5.2 Impact of Poses' Variety

```
%% ------- Reconstruction using Odometry / Mocap Triangulation -------- %
P_ref = P_est(22,:);
for a=22:36
    WND(a) = norm(P_ref - P_est(a));
    % ---------- Obtain SPECIFIC Transformation Matrix -------- %
    T_w2c = GetTranslationMat(a, bag, 1);
    M = K*T_w2c(1:3,:);                 % M : Projection Matrix
    % ---------------- Form A x = b linear system ---------------- %
    b((2*a-1):(2*a),1) = [-u(a)*M(3,4) + M(1,4); -v(a)*M(3,4) + M(2,4) ];

    A( (2*a-1):(2*a),:) = [u(a)*M(3,1)-M(1,1), u(a)*M(3,2)-M(1,2), u(a)*M
        (3,3)-M(1,3);
        v(a)*M(3,1)-M(2,1), v(a)*M(3,2)-M(2,2), v(a)*M(3,3)-M(2,3)];
    if a>22
        X_Odom(:,a) = inv(A'*A)*A'*b;
    end
end

for a=22:36
    err_Odom(a) = norm( Lnd_2 - X_Odom(:,a));
end

figure(1); hold on; grid on; box on;
plot(WND(23:36), err_Odom(23:36), 'b-', 'LineWidth', 2);
ind_1(2) = xlabel('Weighted Normalized Difference');
ind_1(3) = ylabel('3D Triangulation Error [m]');
set(gca,'FontSize', 16 );
ind_1(4) = title('3D Triangulation Error Vs. WND','FontSize',20);ind_1(1) =
    title('Geographic Coordinates vs. Step');
set(ind_1, 'Interpreter', 'latex', 'fontsize', 14 );
```

In this section we're requested to present a correlation between variety in poses and and 3D triangulation error. In order to do so we need to choose a compatible segment that fulfils 3 conditions :

1. **Line Of Sight :** The robot "sees" Landmark 2 along the segment (**Figure 4**).

2. **Motion :** Triangulation occurs while the robot moves along route.

3. **Variance :** The error differs between the images in the segment, and is not constant.

The chosen segment that satisfies these 3 requirement, is the marked **Window of opportunity** :

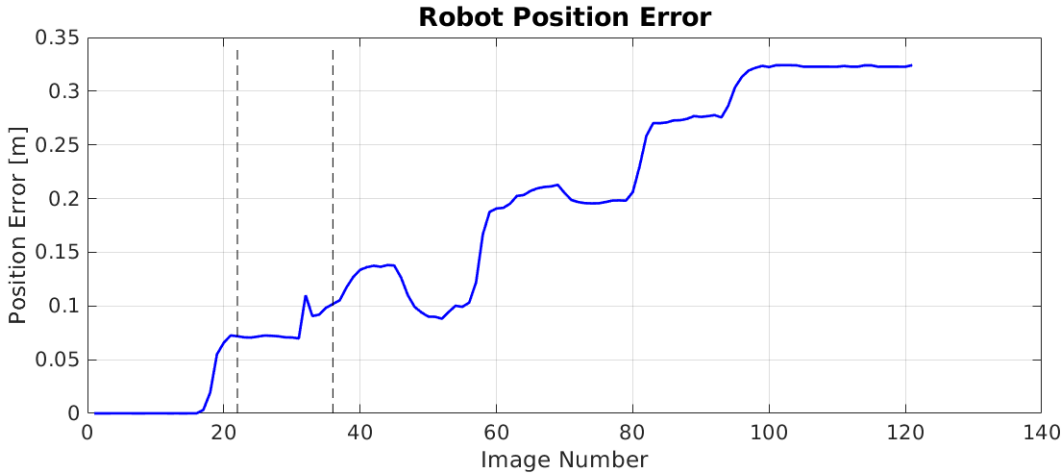

Figure 10: Images 22:36 domain

And now we'll calculate the 3D triangulation error vs. the Weighted Normalized Difference :

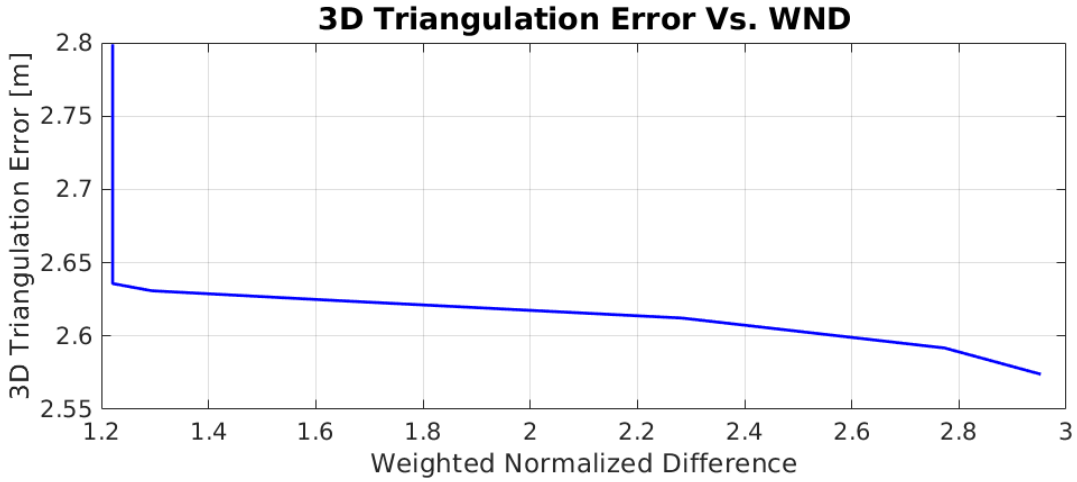

Figure 11: 3D Triangulation Vs WND

One can see the 3D error declination as function of the robot progression. However, the sharp asymptote at the beginning is explained by the robot's sharp twist while taking the 1st picture.

5.3 Results Presentation

```
%% ------------------------ 5.c Plot a graph ------------------------ %
hold on; grid on;
scatter(P_est(:,1), P_est(:,2), 'MarkerEdgeColor', 'k','MarkerFaceColor', '
    b');
scatter(P_GT_fix(:,1), P_GT_fix(:,2), 'MarkerEdgeColor', 'k','
    MarkerFaceColor','r' );
% ------------------ LandMarks ------------------ %
scatter(Lnd_1(1), Lnd_1(2), 300, 'k*');
scatter(Lnd_2(1), Lnd_2(2), 300, 'k*');
% ---------- 3D Triangulations ---------- %
plot(P_est(:,1), P_est(:,2), 'b--', 'linewidth', 1.5  );
plot(P_GT_fix(:,1), P_GT_fix(:,2), 'r--', 'linewidth', 1.5  );
ind(1) = xlabel('Iteration [$\#$]');
ind(2) = ylabel('3D Triangulation Error');
ind(3) = title('3D Triangulation Error Vs. WND');
```

Hereby presented a graph showing the the results :

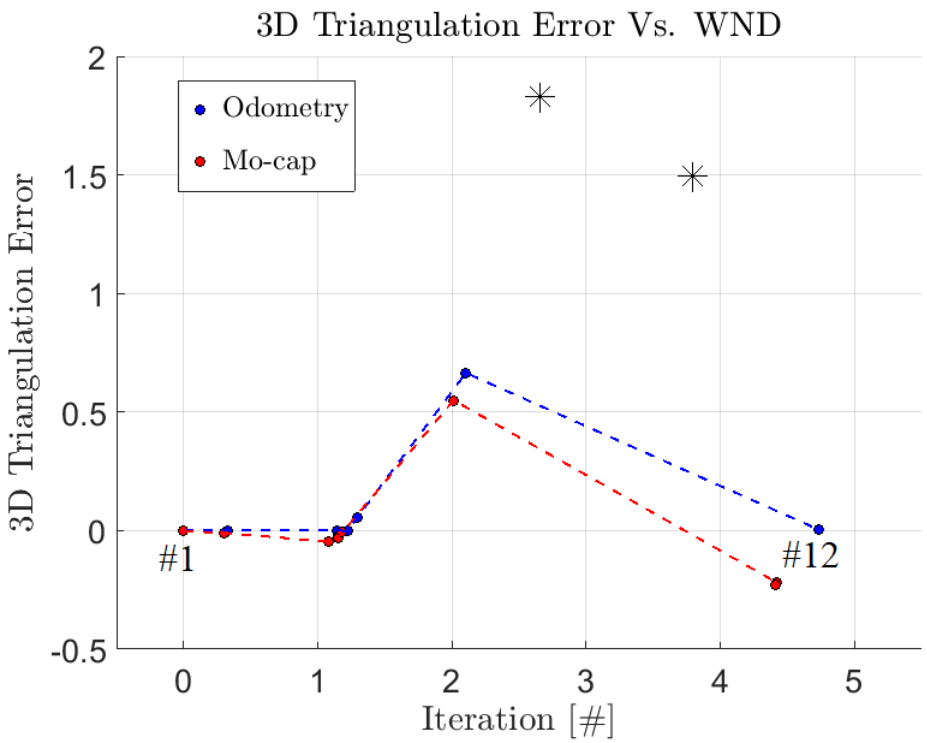

Figure 12: Results Presentation

5.4 Used Frames

Figure 13: 1st Set of Images [5 18 20]

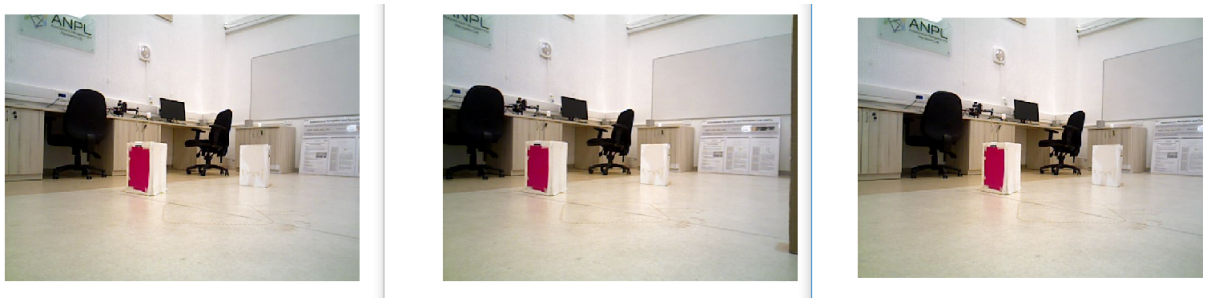

Figure 14: 2nd Set of Images 24 26 27]

Figure 15: 3rd Set of Images [30 32 34]

Figure 16: 4th Set of Images [96 100 115]