```
% Hier soll Bildtransformation durchgeführt werden. Bird's Eye View soll berechnet werden
% Zuerst ros_bags.m ausführen!

%% Neigungswinkel in x-, y- und z-Richtungen
g_vec = sqrt(imu_data{10}.LinearAcceleration.Z.^2 + ...
    imu_data{10}.LinearAcceleration.Y.^2 + imu_data{10}.LinearAcceleration.X.^2);
theta = acos(imu_data{10}.LinearAcceleration.Z/g_vec); % Drehung um x
psi = acos(imu_data{10}.LinearAcceleration.X/g_vec); % Drehung um z
phi = 0; % Drehung um y

%% Matrizen Aufstellen
Rx = [1 0 0;
      0 cos(theta) -sin(theta);
      0 sin(theta) cos(theta)];

Ry = [cos(phi) 0 sin(phi);
      0 1 0;
      -sin(phi) 0 cos(phi)];

Rz = [cos(psi) -sin(psi) 0;
      sin(psi) cos(psi) 0;
      0 0 1];

Intr = [1396.34485420153 0 size(img,2)/2;
        0 1396.34485420153 size(img,1)/2;
        0 0 1];

pixel = 0.00135; % Größe eines Pixels in mm

T_vec = [0;0;350]; % Kamera befindet sich in 330 mm Höhe

% Dritte Spalte der R-Matrix durch T-Vektor ersetzen.
R = Rx*Ry*Rz;
R(:,3) = 0;
R(:,3) = T_vec;
```

```
% H-Matrix ist die um Translation ergänzte Rotationsmatrix multipliziert
% mit der intrinsischen Matrix
H = Intr*R;

A = H^-1; % Inverse Matrix berechnen.
%% Bildtransformation
img_gray = rgb2gray(img); % Bild in Grayscale Image umwandeln!
undist_gimg = undistortImage(img_gray,cameraParams_realsense);

% % Bild Auswählen
% [file,path] = uigetfile('*.*');
% f1 = fullfile(path,file);
% if prod(double(file)) == 0 && prod(double(path)) == 0
%     return
% end

a = undist_gimg;%imread(f1); % ausgewählte Datei laden
%a_gray = rgb2gray(a);
b = size(a);

% Wenn das ausgewählte Bild ein RGB Bild ist, dann setze a1 auf 1. Dies ist
% für die Berechnung der neuen Pixelkoordinaten notwendig.
a1 = 0;
if a(3) == 3
    a1 = 1;
end

% Translation Matrix. Verschiebt das Zentrum von (0,0) in die Mitte des
% Bildes
trans = [1 0 -b(2)/2;
         0 1 -b(1)/2;
         0 0 1];

% Berechnung neuer Koordinaten
```

```
outx = zeros(b(1),b(2));
outy = zeros(b(1),b(2));

for i = 1:b(1)
    for j = 1:b(2)
        new = A*trans*[j;i;1]; % Neue Koordinate für jedes Pixel berechnen
        outx(i,j) = round(new(2)/new(3));
        outy(i,j) = round(new(1)/new(3));
    end
end

% Erzeugen des transformierten Bildes
minoutx = min(outx,[],'all');
minouty = min(outy,[],'all');

maxoutx = max(outx,[],'all');
maxouty = max(outy,[],'all');

% abs() bedeutet Absolutwert oder Betrag. Da wir das Bild verschoben haben,
% entstehen Negativwerte. In Matlab sind negative Koordinaten nicht
% erlaubt. Daher müssen die Koordinaten in positive umgerechnet werden.
f = zeros(maxouty+abs(minouty)+1, maxoutx+abs(minoutx)+1);

for i = 1:b(1)
    for j = 1:b(2)
        f(outy(i,j)+abs(minouty)+1,outx(i,j)+abs(minoutx)+1,1) = a(i,j,1);
        if a1 == 1
            f(outy(i,j)+abs(minouty)+1,outx(i,j)+abs(minoutx)+1,2) = a(i,j,2);
            f(outy(i,j)+abs(minouty)+1,outx(i,j)+abs(minoutx)+1,3) = a(i,j,3);
        end
    end
end

%Die Lücken mit einem Median Filter füllen
b1 = size(f);
```

```
for i = 2:b1(1)-2
    for j = 2:b1(2)-2
        if f(i,j)==0
       f(i,j) = median([f(i-1,j-1),f(i-1,j),f(i-1,j+1),f(i,j-1),f(i,j),f(i,j+1),f(i+1,j-1),f(i+1,j),f(i+1,j+1)]);
       if a1 == 1
       f(i,j,2) = median([f(i-1,j-1,2),f(i-1,j,2),f(i-1,j+1,2),f(i,j-1,2),f(i,j,2),f(i,j+1,2),f(i+1,j-1,2),f(i+1,j,2),f(i+1
j+1,2)]);
       f(i,j,3) = median([f(i-1,j-1,3),f(i-1,j,3),f(i-1,j+1,3),f(i,j-1,3),f(i,j,3),f(i,j+1,3),f(i+1,j-1,3),f(i+1,j,3),f(i+1
j+1,3)]);
       end
        end
    end
end


%% Display the Images
figure;
imshow(uint8(a));
title('Original Image')
figure;
imshow(uint8(f));
title('Transformed Image')
```