**Lidar and Camera Calibration**

**Load Data**

**Load the Velodyne HDL-64 sensor data**

imagePath = fullfile(toolboxdir('lidar'),'lidardata','lcc','HDL64','images');

ptCloudPath = fullfile(toolboxdir('lidar'),'lidardata','lcc','HDL64','pointCloud');

cameraParamsPath = fullfile(imagePath,'calibration.mat');

intrinsic = load(cameraParamsPath);

imds = imageDatastore(imagePath);

imageFileNames = imds.Files;

pcds = fileDatastore(ptCloudPath,'ReadFcn',@pcread);

ptCloudFileNames = pcds.Files;

squareSize = 200;

rng('default')

**Detect Checkerboard Corners**

[imageCorners3d,checkerboardDimension,dataUsed] = estimateCheckerboardCorners3d(imageFileNames,intrinsic.cameraParams,squareSize);

imageFileNames = imageFileNames(dataUsed);

helperShowImageCorners(imageCorners3d,imageFileNames,intrinsic.cameraParams)

**Detect Checkerboard Plane**

roi = helperComputeROI(imageCorners3d,5);

ptCloudFileNames = ptCloudFileNames(dataUsed);

[lidarCheckerboardPlanes,framesUsed,indices] =detectRectangularPlanePoints(ptCloudFileNames,checkerboardDimension,ROI=roi);

ptCloudFileNames = ptCloudFileNames(framesUsed);

imageFileNames = imageFileNames(framesUsed);

imageCorners3d = imageCorners3d(:,:,framesUsed);

helperShowCheckerboardPlanes(ptCloudFileNames,indices)

**Calibrate Lidar and Camera**

[tform,errors] = estimateLidarCameraTransform(lidarCheckerboardPlanes,imageCorners3d,intrinsic.cameraParams);

helperFuseLidarCamera(imageFileNames,ptCloudFileNames,indices,intrinsic.cameraParams,tform);

helperShowError(errors)

function roi = helperComputeROI(imageCorners3d, tolerance)

xCamera = reshape(imageCorners3d(:, 1, :), [], 1);

yCamera = reshape(imageCorners3d(:, 2, :), [], 1);

zCamera = reshape(imageCorners3d(:, 3, :), [], 1);

xMaxLidar = max(zCamera) + tolerance;

xMinLidar = min(zCamera) - tolerance;

yMaxLidar = max(xCamera) + tolerance;

yMinLidar = min(xCamera) - tolerance;

zMaxLidar = max(yCamera) + tolerance;

zMinLidar = min(yCamera) - tolerance;

roi = [xMinLidar, xMaxLidar, yMinLidar, yMaxLidar, zMinLidar, zMaxLidar];

end

function helperFuseLidarCamera(imageFileNames, ptCloudFileNames, indices, intrinsic, tform)

pc = pcread(ptCloudFileNames{1});

indices = indices(~cellfun('isempty',indices));

figureH = figure('Position', [0, 0, 640, 480]);

panel = uipanel('Parent',figureH,'Title','Colored Lidar Points', 'Position',[0.01,0,1,0.55]);

ax = axes('Parent',panel,'Color',[0,0,0],'Position',[0 0 1 1],'NextPlot','add');

ax.XLim = [0, 3];

ax.YLim = [-5, 5];

ax.ZLim = [-4, 2];

axis(ax,'equal');

zoom on;

scatterH = scatter3(ax, nan, nan, nan, 20, '.');

view(ax,3);

campos([-117.6074 -48.5841 53.3789]);

imagepanel = uipanel('Parent',figureH,'Title','Projected Lidar Points', 'Position',[0.01,0.55,1,0.45]);

imax = axes('Parent',imagepanel,'Color',[0,0,0],'Position',[0 0 1 1],'NextPlot','add');

imH = imshow([],'Parent',imax);

imax.Position = [0,0,1,1];

axis(imax, 'equal');

h = [];

numFrames = numel(ptCloudFileNames);

for i = 1:numFrames

im = imread(imageFileNames{i});

J = undistortImage(im, intrinsic);

if ~isempty(h)

delete(h);

end

imH.CData = J;

ptCloud = pcread(ptCloudFileNames{i});

xyzPts = ptCloud.Location;

if ~ismatrix(xyzPts)

x = reshape(xyzPts(:, :, 1), [], 1);

y = reshape(xyzPts(:, :, 2), [], 1);

z = reshape(xyzPts(:, :, 3), [], 1);

else

x = xyzPts(:, 1);

y = xyzPts(:, 2);

z = xyzPts(:, 3);

end

ptCloud = pointCloud([x, y, z]);

plane = select(ptCloud, indices{i});

[~, indice]= projectLidarPointsOnImage(ptCloud, intrinsic, tform);

subpc = select(ptCloud, indice);

ptCloud = fuseCameraToLidar(J, subpc, intrinsic, invert(tform));

x = ptCloud.Location(:, 1);

y = ptCloud.Location(:, 2);

z = ptCloud.Location(:, 3);

projectedPtCloud = projectLidarPointsOnImage(plane, intrinsic, tform);

hold(imax,'on');

h = plot(projectedPtCloud(:, 1), projectedPtCloud(:, 2), '.r');

hold(imax,'off');

set(scatterH,'XData',x,'YData',y,'ZData',z, 'CData', ptCloud.Color);

pause(1);

end

end

function helperShowCheckerboardPlanes(ptCloudFileNames, indices)

pc = pcread(ptCloudFileNames{1});

indices = indices(~cellfun('isempty',indices));

figureH = figure('Position', [0, 0, 640, 480]);

panel = uipanel('Parent',figureH,'Title','Lidar Features');

ax = axes('Parent',panel,'Color',[0,0,0],'Position',[0 0 1 1],'NextPlot','add');

ax.XLim = pc.XLimits;

ax.YLim = pc.YLimits;

ax.ZLim = pc.ZLimits + [-1, 1];

axis(ax,'equal');

zoom on;

scatterH = scatter3(ax, nan, nan, nan, 7, '.');

view(ax,3);

campos([-117.6074 -48.5841 53.3789]);

numFrames = numel(ptCloudFileNames);

for i = 1:numFrames

ptCloud = pcread(ptCloudFileNames{i});

xyzPts = ptCloud.Location;

if ~ismatrix(xyzPts)

x = reshape(xyzPts(:, :, 1), [], 1);

y = reshape(xyzPts(:, :, 2), [], 1);

z = reshape(xyzPts(:, :, 3), [], 1);

xyzPts = [x, y, z];

else

x = xyzPts(:, 1);

y = xyzPts(:, 2);

z = xyzPts(:, 3);

end

ptCloud = pointCloud(xyzPts);

color = uint8([255\*ones(size(ptCloud.Location, 1), 1),...

zeros(size(ptCloud.Location, 1), 1), 255\*ones(size(ptCloud.Location, 1), 1)]);

x = ptCloud.Location(:, 1);

y = ptCloud.Location(:, 2);

z = ptCloud.Location(:, 3);

r = color(:, 1);

g = color(:, 2);

b = color(:, 3);

ptCloud.Color = color;

plane = select(ptCloud, indices{i});

r(indices{i}) = zeros(plane.Count, 1);

g(indices{i}) = 255\*ones(plane.Count, 1);

b(indices{i}) = zeros(plane.Count, 1);

planeSize = plane.Count;

planeColor = uint8([zeros(planeSize, 1),...

255\*ones(planeSize, 1), zeros(planeSize, 1)]);

plane.Color = planeColor;

set(scatterH,'XData',x,'YData',y,'ZData',z, 'CData', [r,g,b]);

pause(1);

end

end

function helperShowImageCorners(imageCorners3d, imageFileNames, intrinsic)

figureH = figure('Position', [0, 0, 480, 320]);

panel = uipanel('Parent', figureH, 'Position', [0 0 1 1], 'Title', 'Image Features');

ax = axes('Parent',panel);

numImages = numel(imageFileNames);

for i = 1:numImages

im = imread(imageFileNames{i});

J = undistortImage(im, intrinsic);

imCorners3d = imageCorners3d(:, :, i);

imCorners2d = projectLidarPointsOnImage(imCorners3d, intrinsic, rigid3d());

edge = [imCorners2d(4, 1:2); imCorners2d(1, 1:2)];

imshow(J, 'Parent', ax);

hold on;

plot(imCorners2d(1:2, 1), imCorners2d(1:2, 2), '-', 'Color','green', 'LineWidth' ,4, 'Parent' ,ax);

plot(imCorners2d(2:3, 1), imCorners2d(2:3, 2), '-', 'Color','blue', 'LineWidth', 4, 'Parent', ax);

plot(imCorners2d(3:4, 1), imCorners2d(3:4, 2), '-', 'Color','magenta', 'LineWidth', 4,'Parent', ax);

plot(edge(:, 1), edge(:, 2), '-', 'Color','red', 'LineWidth', 4, 'Parent', ax);

plot(imCorners2d(1, 1), imCorners2d(1, 2), 'og', 'MarkerSize', 10, 'Parent', ax);

plot(imCorners2d(2, 1), imCorners2d(2, 2), 'or', 'MarkerSize', 10, 'Parent', ax);

plot(imCorners2d(3, 1), imCorners2d(3, 2), 'ob', 'MarkerSize', 10, 'Parent', ax);

plot(imCorners2d(4, 1), imCorners2d(4, 2), 'om', 'MarkerSize', 10, 'Parent', ax);

hold off;

pause(1);

end

function helperShowError(errors)

figureH = figure('Visible','off','Position',[0, 0, 1200, 640],...

'Name','Error Plots');

panel1 = uipanel('Parent',figureH,'Position',[0.04,0.58,0.42,0.38],...

'Title','Translation Error','FontSize',15,'TitlePosition','centertop');

axes1 = axes('Parent',panel1,'Position',[0.1 0.1 0.8 0.8],'NextPlot','add');

axes1.Toolbar.Visible = 'off';

axis(axes1,'tight');

disableDefaultInteractivity(axes1)

panel2 = uipanel('Parent',figureH,'Position',[0.55,0.58,0.40,0.38],'Title','Rotation Error','FontSize',15,'TitlePosition','centertop');

axes2 = axes('Parent',panel2,'Position',[0.1 0.1 0.8 0.8],'NextPlot','add');

axes2.Toolbar.Visible = 'off';

axis(axes2,'tight');

disableDefaultInteractivity(axes2)

panel3 = uipanel('Parent',figureH,'Position',[0.25,0.03,0.40,0.38],'Title','Reprojection Error','FontSize',15,'TitlePosition','centertop');

axes3 = axes('Parent',panel3,'Position',[0.1 0.1 0.8 0.8],'NextPlot','add');

axes3.Toolbar.Visible = 'off';

axis(axes3,'tight');

disableDefaultInteractivity(axes3)

set(figureH,'Visible','on');

numFrames = size(errors.TranslationError,1);

bar(axes1,errors.TranslationError);

line1H = plot(axes1,[1,numFrames],[mean(errors.TranslationError),mean(errors.TranslationError)],'--','Color','b');

legend(line1H,strcat('Overall Mean Translation Error: ', num2str(mean(errors.TranslationError)),' in m'),'Location','southeast');

bar(axes2,errors.RotationError);

line2H=plot(axes2,[1,numFrames],[mean(errors.RotationError),mean(errors.RotationError)],'--','Color','b');

legend(line2H,strcat('Overall Mean Rotation Error: ', num2str(mean(errors.RotationError)),' in deg'),'Location','southeast');

bar(axes3,errors.ReprojectionError);

line3H=plot(axes3,[1,numFrames],[mean(errors.ReprojectionError),mean(errors.ReprojectionError)],'--','Color','b');

legend(line3H,strcat('Overall Mean Reprojection Error: ', num2str(mean(errors.ReprojectionError)),' in pixel'),'Location','southeast');

end