Contents

List of Figures

List of Tables

Chapter 1

Methodology for Image Registration

This chapter describes the methodology for image registration, that is, the process that colorizes (defines the color) the point cloud based on the images taken in the acquisitions. This method can be split into two parts: the Color Registration (Section 1.1), where the process is described per-image and each image colorizes a portion of the point cloud, and the Color Fusion (Section 1.2), where all the colorized point cloud partials are merged into the final colorized point cloud. Also, the pixel registration relies on a camera calibration, both the intrinsic calibration and also the extrinsic, so two methods are shown to obtain this calibration (Sections 1.3 and 1.4).

1.1 Color Registration

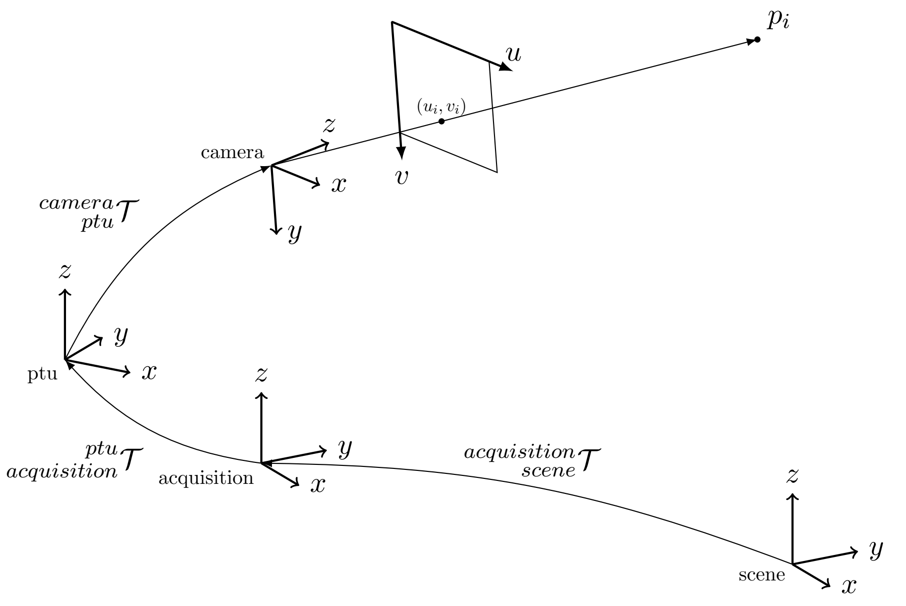
This method describes how to colorize a point cloud based on a single image, using a working principle similar to the ray tracing used in computer graphics. As an overview, each point in the point cloud can be transformed as a ray in the camera perspective, which is basically the path from the eye point to the point. This ray can be used to retrieve the original color of the point from the image. However, this process is not so straightforward, because the position and orientation of the camera has to be very precise and occluded points need to be rejected.

1.1.1 Point to pixel coordinates transformation

To start, each point has to be transformed, because the original point is registered in the scene coordinate frame (p_{scene}) and has to be registered into the camera coordinate frame (p_{camera}). So, the transformations ${}^{acquisition}_{scene}\mathcal{T}$, ${}^{ptu}_{acquisition}\mathcal{T}$, ${}^{camera}_{ptu}\mathcal{T}$ can be used according to Equation (1.1). The ${}^{acquisition}_{scene}\mathcal{T}$ transformation is obtained in **??**, ${}^{ptu}_{acquisition}\mathcal{T}$ is the transformation of the PTU and ${}^{camera}_{ptu}\mathcal{T}$ is the extrinsic calibration of the camera and the method to obtain it is in Section 1.4. The transformation graph can be seen in Figure 1.1.

$$p_{scene} = {}^{acquisition}_{scene}\mathcal{T} \cdot {}^{ptu}_{acquisition}\mathcal{T} \cdot {}^{camera}_{ptu}\mathcal{T} \cdot p_{camera} \tag{1.1}$$

Next, each point was transformed into pixel coordinates (u, v), using the pinhole camera model. This model defined how a light ray in projected in the image sensor of a camera and has two parameters: the focal length $f = (f_x, f_y)$ and optical center $(c = (c_x, c_y))$. This



p_i
u
(u_i, v_i)
v
z
camera
x
y
${}^{camera}_{ptu}\mathcal{T}$
z
y
ptu
x
z
y
${}^{ptu}_{acquisition}\mathcal{T}$
acquisition
x
${}^{acquisition}_{scene}\mathcal{T}$
z
y
scene
x

Figure 1.1: Color registration for a single point

parameters are obtained in the intrinsic calibration of the camera (Section 1.3). According to this model, each point is projected as pixel coordinates (u, v) to a plane located a unit distance from the camera eye point, using the perspective projection matrix in Equation (1.2), according to Equation (1.3).

$$\mathcal{P} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

$$\begin{pmatrix} uz \\ vz \\ z \end{pmatrix} = \mathcal{P} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{1.3}$$

1.1.2 Camera Distortion

The pinhole camera model does not regard the distortion caused by the lens, which is not negligible for most cameras. The two sources of distortion are radial and tangential distortion. Radial distortion makes straight lines appear curved, known as the barrel distortion and pincushion distortion. This distortion is highly noticed in images taken with fish-eye lenses, as seen in Figure 1.2. This distortion can be solved by transforming the (u, v) with Equation (1.4). Similarly, tangential distortion is caused by a misalignment of the lens to the imaging plane, which causes areas in the image to appear closer than expected. This deformation can be solved with the Equation (1.5). In brief, to undistort the image five parameters need to be determined, also known as the distortion coefficients: $\{k_1, k_2, p_1, p_2, k_3\}$, which are obtained in the camera intrinsic calibration method, described in Section 1.3.

$$\begin{pmatrix} u \\ v \end{pmatrix}_{calibrated} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{pmatrix} u \\ v \end{pmatrix} \tag{1.4}$$

$$\begin{pmatrix} u \\ v \end{pmatrix}_{calibrated} = \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} 2p_1 uv + p_2(r^2 + 2u^2) \\ p_2(r^2 + 2v^2) + 2p_2 uv \end{pmatrix} \tag{1.5}$$

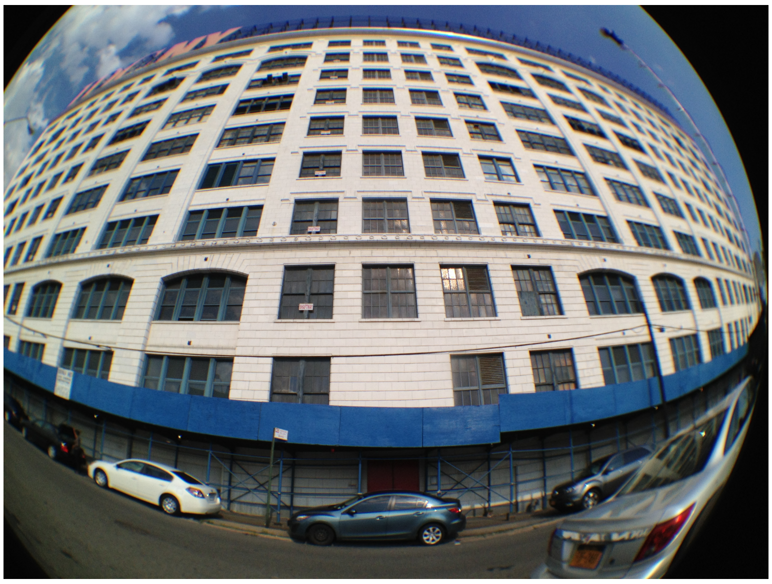


Figure 1.2: Barrel distortion in fish eye lens

1.1.3 Point filtering

Not all points are eligible for the color registration, based on it's location and camera properties, so two filtering steps were used: the first filter removes the points outside the view frustum and the second removes the hidden points.

View Frustrum Removal Filter

The view frustum is defined as the region of space that is captured by the camera sensor, which for pinhole cameras is a pyramid truncated by two parallel planes (the near and far clipping planes), as seen in Figure 1.3. The sides of the frustum are limited by the size of the sensor, so the points that lie outside the bounding box defined by the points $(0, 0)$ and $(width, height)$ is excluded.

The near and far clipping planes should reject the points based on the depth of field of the camera. The depth of field, or DOF, is the distance from the camera to the objects range, so that this objects are in focus. If the object falls out of this range, it starts to loose focus incrementally the farther out it is. There is a point where the object is sharper, which is called point of optimal focus. Two factor influences the DOP: the aperture size and the focal length. The aperture influences the amplitude of the DOF, so a bigger aperture results in a narrower DOF, and the focal length defines the point of optimal focus, so a bigger focal

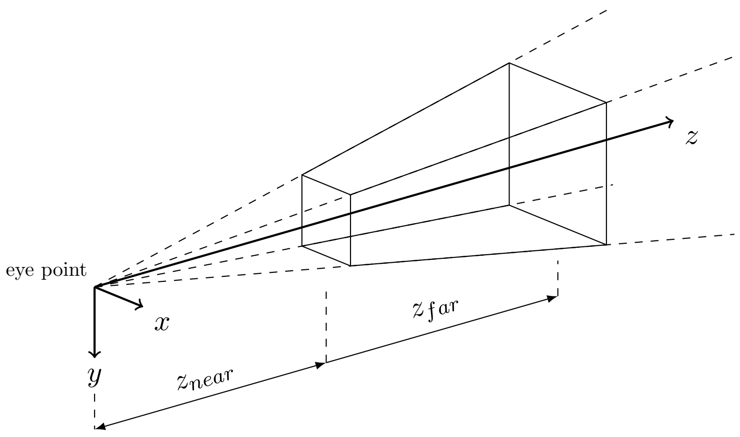


Figure 1.3: Representation of the visual frustum of the camera

length moves the DOF farther away from the camera. The near and far clipping plane should defined to be the boundaries of the DOF, so that all points are in focus.

Based on the frustum of the camera defined above, the points that do not respect Equation (1.6) should be rejected.

$$\begin{aligned} 0 &< u < width \\ \wedge\, 0 &< v < height \\ \wedge\, z_{near} &< z < z_{far} \end{aligned} \tag{1.6}$$

Hidden Point Removal Filter

Not all points that lie on the frustum of the camera are seen by the camera, because some of this points are occluded by nearer objects, so they need to be removed.

In [2], an simple and fast operator, the Hidden Point Remove, or HPR, determines the visibility of point sets, viewed from a given viewport. This method is easily implemented and has a asymptotic complexity of $O(n \log n)$, where n is the number of points in the point cloud. Moreover, this method work well for both sparse and dense point clouds.

The HPR operator operates on a set of points $\mathcal{P} = \{p_i | i = 1 \ldots n\}$, and the goal is to determine whether p_i is visible from a viewpoint $\mathcal{C}$. In this application, C is the origin of the point cloud. The algorithm consists of two steps: the inversion and the convex hull construction.

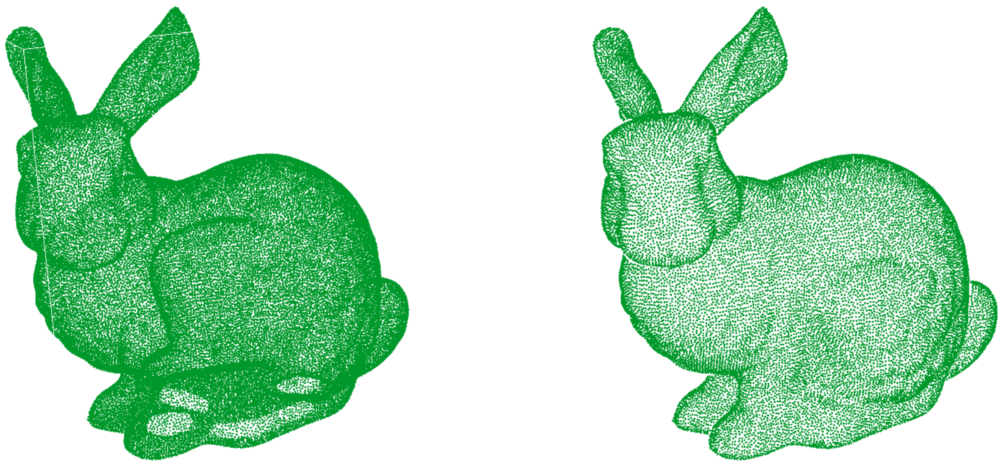
The inversion step maps each point p_i along the ray from $\mathcal{C}$ to p_i, such that $|p_i|$ is monotonically decreasing. There are multiple ways to perform the inversion, but in [2] the *spherical flipping* was used. Spherical flipping reflects a point p_i with respect to a sphere of radius R to the new point $\hat{p_i}$ by applying the Equation (1.7).

$$\hat{p_i} = p_i + 2(R - |p_i|)\frac{p_i}{|p_i|} \tag{1.7}$$

Afterwards, the convex hull of $\hat{\mathcal{P}} \bigcup \{\mathcal{C}\}$, where $\hat{\mathcal{P}}$ is the transformed point set and $\mathcal{C}$ is the center of the sphere, is computed. Finally, the points that lie in the complex hull are the visible points of the point set.

This algorithm only has a parameter, which is the radius R of the sphere used for the spherical flipping, which influences the amount of false positives of the algorithm. In general, R is determined based on the maximum point length $max(|p_i|)$ and a exponential factor α, such that $R = max(|p_i|) \times 10^{\alpha}$. In this application, a factor of $\alpha = 3$ was adequate.

As an example, the HPR operator was used in the Stanford Bunny point cloud, as seen in Figure 1.4 and, as seen, Figure 1.4b only presents the points that are visible, as opposed to Figure 1.4a.



(a) Before the HPR (b) After the HPR

Figure 1.4: Result of the HPR operator in the Bunny point cloud

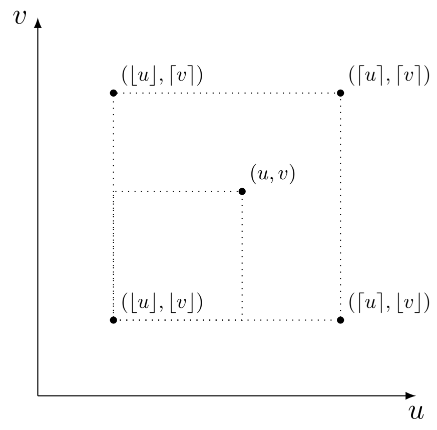
1.1.4 Color Attribution

Finally, the color is extracted from the image at the pixel coordinates (u, v) and saved for the correspondent pixel. Because images are discrete, the color is interpolated using a bilinear interpolation, which uses the neighbor pixels to interpolate the color C at (u, v) in an image I according to Equation (1.8) (the ceil and floor operators are, respectively, $\lceil\cdot\rceil$ and $\lfloor\cdot\rfloor$). The interpolation can be visualized in Figure 1.5.

$$\begin{aligned} C(u, v) = &\, (u - \lceil u\rceil)\ (v - \lceil v\rceil)\ I_{\lfloor u\rfloor,\lfloor v\rfloor} \\ &+ (u - \lceil u\rceil)\ (v - \lfloor v\rfloor)\ I_{\lfloor u\rfloor,\lceil v\rceil} \\ &+ (u - \lfloor u\rfloor)\ (v - \lceil v\rceil)\ I_{\lceil u\rceil,\lfloor v\rfloor} \\ &+ (u - \lfloor u\rfloor)\ (v - \lfloor v\rfloor)\ I_{\lceil u\rceil,\lceil v\rceil} \end{aligned} \tag{1.8}$$

1.2 Color Fusion

In an capture with N_a acquisitions, each one with N_i images, the total number of images account to $N_a \times N_i$. Each one of this images will yield a partial colorized point cloud, according to Section 1.1, and now the point clouds need to be merged into a final point cloud. More specifically, each point p_i has multiple correspondent colors, one for each registered image. The method here described determines the final color in a point-wise fashion and does not account for the neighbor points.



v
u
$(\lfloor u \rfloor, \lceil v \rceil)$
$(\lceil u \rceil, \lceil v \rceil)$
(u, v)
$(\lfloor u \rfloor, \lfloor v \rfloor)$
$(\lceil u \rceil, \lfloor v \rfloor)$

Figure 1.5: Bilinear interpolation in an image

Let us admit that the point p has a set $C = \{c_i | i = 1 \dots k\}$ of k registered colors. The final color of this point c should be a combination of the colors in C. The first and easier approach is to average the colors to obtain the color c, as seen in Equation (1.9). However, this is a poor heuristic as it considers that all colors have the same error, which is not true. For example, an image taken closer to an object is more precise than one taken away from it.

$$c = \frac{1}{k} \sum_{i}^{k} c_i \tag{1.9}$$

A common solution for this mean limitation is to use an weighted mean, shown in Equation (1.10). The w_i are the weights for each color and should reflect the quality of each color, because colors with bigger weight have a bigger influence in the final color.

$$c = \frac{\sum_i^k w_i c_i}{\sum_i^k w_i} \tag{1.10}$$

In this work, the quality measurement was determined based on an heuristic that depends on two factors, that are obtained in the color registration phase (Section 1.1).

The first factor f_1 depends on the distance d from the camera to the point and on the optimal focus point d_f. f_1 is smaller the bigger the distance between d and d_f. The function used was the gaussian centered on d_f. The second factor f_2 depends on the distance from the pixel coordinates (u, v) to the center of the optical center (c_x, c_y). Again, a gaussian distribution was used to calculate f_2, and a bigger distance also yields a smaller f_2. In brief, both factors f_1 and f_2 are calculated according to Equations (1.11) and (1.12). The parameters α and β determine how wide the gaussian function is, so points farther from the peak point influence more or less.

$$f_1 = e^{-\frac{(d-d_f)^2}{2\alpha^2}} \tag{1.11}$$

$$f_2 = e^{-\frac{(u-c_x)^2+(v-c_y)^2}{2\beta^2}} \tag{1.12}$$

$$\tag{1.13}$$

Figure 1.6: Interface for the *cameracalibrator* node

The two factors are then combined into the weight w factor of the color, based on a linear combination, dependent on a parameter s, which determines the influence of each factor, as seen on Equation (1.14).

$$w = sf_1 + (1 - s)f_2 \tag{1.14}$$

In conclusion, for each point p_i the color c_i is attributed, based on the registered colors of each image. The fusion of all this colors is based on a weighted mean, where the weight of each color is determined by an heuristic that considers the location of the color in pixel coordinates and the distance of the point to the camera, in order to benefit points that have a better quality in the measurement, for example, points that are in focus or points that are closer to the camera center. This process is repeated for all the points of the point cloud until every point has a color (however, some points have no color registered, because no color was registered before).

1.3 Camera Intrinsic Calibration

The intrinsic calibration determines the intrinsic parameters of the camera, such as the focal lenght ($f = (f_x, f_y)$), the optical center ($c = (c_x, c_y)$) and the distortion coefficients to model the lens distortion. This parameters are required to create the distortion matrix, as well as the undistortion function, which is essencial for the point projection (as described in **??**).

The calibration procedure used in this work is a standard procedure for cameras with low distortion and is known as the chessboard camera calibration. This method calibrates a monocular camera with fixed focus using a sequence of images taken from a chessboard with known dimensions. In order to improve the calibration results, the chessboard should rotate and move, in order to occupy the entire image size.

After all the images are obtained, the corners of the chessboard are extracted and the re-projection error is minimized to obtain the the intrinsic parameters. The results of this calibration are more accurate if the corners of the chessboard are well defined in the image, so the chessboard should have an appropriate size. Also, the chessboard poses should be enough and should be well distributed spatially.

In the end, the accuracy of the calibration should be measured for new images, with the re-projection error. This value should be as low as possible and, as a rule of thumb, a value less than 0.01 is acceptable.

In ROS, this calibration is easily obtained with the *cameracalibrator.py*, which includes a graphical interface, and provides feedback about the corner detection and the state of the calibration. The interface is shown on Figure 1.6. In this system, this data is first saved into a ROS *camera_info* file. Then, this file is also saved in each capture in the parameters file.

1.4 Camera Extrinsic Calibration

The extrinsic parameters of the camera the position and orientation of the camera in the robot. In this case, the camera was mounted statically on top of the PTU, just like the laser scanner. This calibration that determines this extrinsic parameters is known as the eye-in-hand calibration, described in [1].

Bibliography

[1] Radu Horaud and Fadi Dornaika. “Hand-Eye Calibration”. In: 14 (June 1995), pp. 195–210.

[2] Sagi Katz, Ayellet Tal, and Ronen Basri. “Direct visibility of point sets”. In: 26 (July 2007).