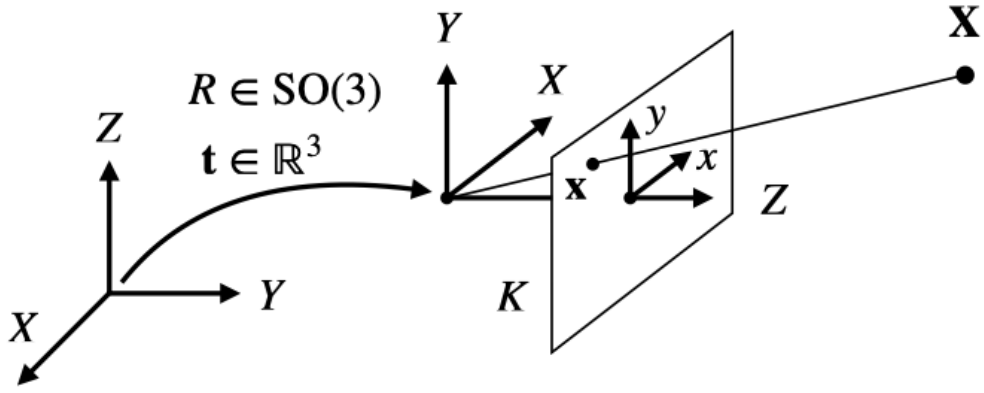
🚙 03 - Camera Calibration

In this activity we will calibrate the parameters of the camera projection matrix for our robots. This involves two procedures:

1. **Intrinsic Calibration**: Estimate the intrinsic parameters of your camera, including the focal length(s), center of projection, and lens distortion parameters (which we didn't cover in this course).
2. **Extrinsic Calibration**: Estimate the homography that transforms points between the ground plane and the image plane.

You will need the Duckietown hardware to proceed. If you do not have a Duckiebot and Duckietown, proceed to the next activity: the image filtering tutorial.

Approach



Pinhole camera geometry.

As we have seen, the camera matrix models the transformation of 3D scene points, expressed relative to a Cartesian world frame, to their corresponding projection on the image plane, expressed in terms of pixels. Assuming that we express the points in homogeneous coordinates, we have shown that we can decompose the transformation as the product of two matrices:

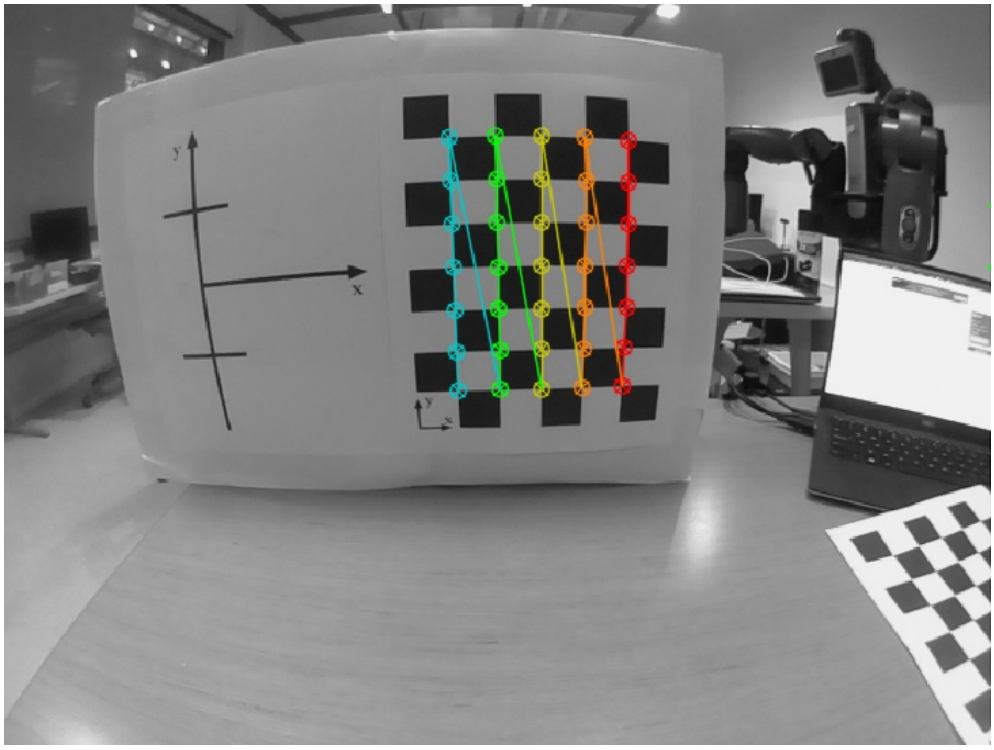
$$\mathbf{x}_i = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} [\,\mathbf{R} \mid \mathbf{t}\,] \mathbf{X}_i \tag{1}$$

$$= K [\,\mathbf{R} \mid \mathbf{t}\,] \mathbf{X}_i \tag{2}$$

where $\mathbf{X}_i$ is the four-vector that specifies the homogeneous coordinates of a point in the world frame and $\mathbf{x}_i$ is the three-vector that specifies the homogeneous coordinates in the image.

In the following, we will estimate the parameters of these two matrices, starting with the camera's intrinsic matrix K.

Intrinsic Calibration



The checkerboard used for calibration annotated with the detected points.

Calibration involves optimizing some objective (e.g., reprojection error) given known correspondences between coordinates in the scene $\mathbf{X}_i$ and their image-space projections $\mathbf{x}_i$. Checkerboard patterns are often used for this purpose since they provide points that are easy to detect and well defined (i.e., we can easily define their coordinates). Because checkerboards are planar (i.e., $Z_i = 0 \; \forall i$) there is a homography that relates the coordinates of points on the checkerboard to their image-space coordinates.

$$\mathbf{x}_i = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 0 \\ 1 \end{bmatrix} \quad \text{since } Z_i = 0 \tag{4}$$

$$= \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & t_x \\ R_{21} & R_{22} & t_y \\ R_{31} & R_{32} & t_z \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{5}$$

$$= \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix} \tag{6}$$

The vectors $\mathbf{X}_i$ and $\mathbf{x}_i$ specify the coordinates of the world and image points in *homogeneous coordinates*.

Letting $\mathbf{r}_1$ and $\mathbf{r}_2$ be the first and second columns of the (unknown) rotation matrix associated with the world (checkerboard)-to-camera transformation, and (u_i, v_i) be the pixel coordinates of point $\mathbf{x}_i$ (i.e., the non-homogeneous coordinates), we have

$$\alpha \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = K \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ 1 \end{bmatrix}$$

where α is a scale term (since $\mathbf{x}_i$ is in homogeneous coordinates). Letting $\mathbf{h}_j$ be the j^{th} column of H, we can relate the above expression to the homography matrix

$$\begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda K \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}$$

where λ is a non-zero constant (since the projection is only defined up-to-scale).

As columns of the rotation matrix, $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal (i.e., they are orthogonal and they have unit norm). In other words, the columns $\mathbf{r}_i$ of any rotation matrix R exhibit the following properties

$$\mathbf{r}_i^\top \mathbf{r}_j = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases}$$

In order to see why this is the case, let's start with the fact that the inverse of any rotation matrix is its transpose, i.e., $R^{-1} = R^\top$. Let's consider two reference frames A and B that share the same origin. Let R be the rotation matrix that transforms points expressed relative to frame A in terms of their coordinates in frame B (since the reference frames share the same origin, the transformation does not include a translation or, more precisely, the transformation is the zero-vector). Consider a vector

$\mathbf{X}$ that specifies the coordinates of a point with respect to frame A. We can express the coordinates for the same point with respect to frame B as $\mathbf{X}' = R\mathbf{X}$. Similarly, we can transform $\mathbf{X}'$, and any other vector in frame B, to frame A as

$$\begin{aligned} \mathbf{X} &= R^{-1}\mathbf{X}' && (9)\\ &= R^{-1}R\mathbf{X} && (10)\\ &= \mathbf{X} && (11) \end{aligned}$$

This, together with the fact that $R^{-1} = R^{\top}$, implies that

$$R^{\top}R = \begin{bmatrix} \mathbf{r}_1^{\top}\mathbf{r}_1 & \mathbf{r}_1^{\top}\mathbf{r}_2 & \mathbf{r}_1^{\top}\mathbf{r}_3 \\ \mathbf{r}_2^{\top}\mathbf{r}_1 & \mathbf{r}_2^{\top}\mathbf{r}_2 & \mathbf{r}_2^{\top}\mathbf{r}_3 \\ \mathbf{r}_3^{\top}\mathbf{r}_1 & \mathbf{r}_3^{\top}\mathbf{r}_2 & \mathbf{r}_3^{\top}\mathbf{r}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and, in turn, that the columns of any rotation matrix R are orthonomral.

One can show that we can exploit this property to get the following two constraints on K :

$$\begin{aligned} \mathbf{h}_1^{\top}K^{-\top}K^{-1}\mathbf{h}_2 &= 0 && (12)\\ \mathbf{h}_1^{\top}K^{-\top}K^{-1}\mathbf{h}_1 &= \mathbf{h}_2^{\top}K^{-\top}K^{-1}\mathbf{h}_2 && (13) \end{aligned}$$

Having estimated the homography associated with a particular image of the checkerboard, the above equations constitute two constraints on the intrinsic parameters (homographies have eight degrees-of-freedom, while the transformation has six degrees-of-freedom, allowing us to only impose two constraints on the intrinsics from a particular image).

We will estimate the intrinsic parameters of the matrix by collecting several images of the same checkerboard, either by moving the camera around in the scene or by keeping the camera fixed and changing the position and orientation of the checkerboard. There is a homography associated with each image of the checkerboard (since either the camera or checkerboard move between images, these homographies will be different for different images), and after estimating the homography, we can use the two equations above in computing the intrinsic matrix.

For more information on this procedure, see the following paper, which proposed the idea:

Zhengyou Zhang, A Flexible New Technique for Camera Calibration, Technical Report MSR-TR-98-71, Microsoft Research, 1998.

🚙 Intrinsic calibration procedure

During this procedure we identify the instrinsic parameters of the camera. These are only a function of the camera specifications (lens, focus, pixel array, etc.), and not of the

placement of the camera in the world.

This procedure can be completed either by keeping the Duckiebot still and moving the calibration pattern, or viceversa. To ensure the calibration pattern is flat at all times, we suggest to keep the camera calibration pattern on the ground and move the Duckiebot.

0. Make sure your Duckiebot is powered-on, its camera is working, and you have a camera calibration pattern available. Also, make sure to focus the camera by gently rotating the lens of the camera.

   Once the calibration is complete, do not touch the focus anymore, ever, as it will invalidate calibration.

1. Place your camera calibration pattern on a flat surface (e.g., a table, your Duckietown, or the floor). Make sure it stays flat on the ground and no wrinkles are present. Tape it down if needed.

2. Open a terminal on your computer and type:

   ```
   dts duckiebot calibrate_intrinsics ROBOTNAME
   ```

3. When the window opens you will need to move the Duckiebot around (grab it with your hands and move it as you wish; it doesn't need to be on the ground.) so that is sees the calibration pattern.

4. Move the Duckiebot in front of the pattern until you see colored lines overlaying the checkerboard. You will only see the colored lines if the entire checkerboard is within the field of view of the camera.

   You should also see colored bars in the sidebar of the display window. These bars indicate the current range of the checkerboard in the camera's field of view:

   - X bar: the observed horizontal range (left - right)
   - Y bar: the observed vertical range (top - bottom)
   - Size bar: the observed range in the checkerboard size (forward - backward from the camera direction)
   - Skew bar: the relative tilt between the checkerboard and the camera direction

   Move the robot such that the checkerboard is located at different locations, orientations, and scales in the image (note that you can alternatively keep the camera stationary and move the checkerboard instead, but if you do so make sure that you keep the checkerboard flat since, afterall, the whole procedure relies on the assumption that the target is planar). After each movement, make sure to pause long enough for the checkerboard to become highlighted. Once you have collected enough data, all four indicator bars will turn green. Press the `CALIBRATE` button in the sidebar.

Calibration will take a few moments. Note that the screen may dim and / or become irresponsive. Don't worry, the calibration is working.

Save the calibration results

Make sure to save your new intrinsic calibration parameters! You can save the results by pressing the `COMMIT` button in the side bar. (You never need to click the `SAVE` button.)

Final check to make sure it's stored

You can verify that the new gain value has been saved on your Duckiebot by opening the Dashboard > File Manager > config > calibrations > camera_intrinsic page.

You should find a file named `ROBOTNAME.yaml` in addition to the `default.yaml`. Double click on it and verify that the gain value is indeed the one you chose.

Lens Distortion

A pinhole camera is an idealized camera model. As discussed in lecture, we made the aperture sufficiently small that light reflecting off different locations in the scene impact different points on the image (sensor) plane. Here, the goal was to minimize the size of the "circle of confusion", regions of the image that record intensity from multiple points in the scene. However, by making the aperture so small, we are, by design, limiting the amount of light that reaches the sensor. This results in a low signal-to-noise ratio, meaning that it is difficult to tease apart valid signal from noise.

In practice, cameras mitigate this effect by increasing the aperture to sizes that result in significant confusion (i.e., aliasing) and then placing a lens in front of the aperture to control light on the sensor. In this way, a greater amount of light is exposed to the sensor, while being focused on the corresponding pixels (i.e., reducing aliasing).

However, lenses themselves are imperfect and cause some distortion in the image. There are several parametric models that describe the effects of different types of distortion and standard calibration procedures, including the one above, estimate these parameters along with the other intrinsic parameters of the camera.

Example: Exploring lens distortion in simulation

In this example, we will visualize the ability to use the results of intrinsic calibration to remove distortions introduced by the lens. This procedure is often referred to as *rectification*.

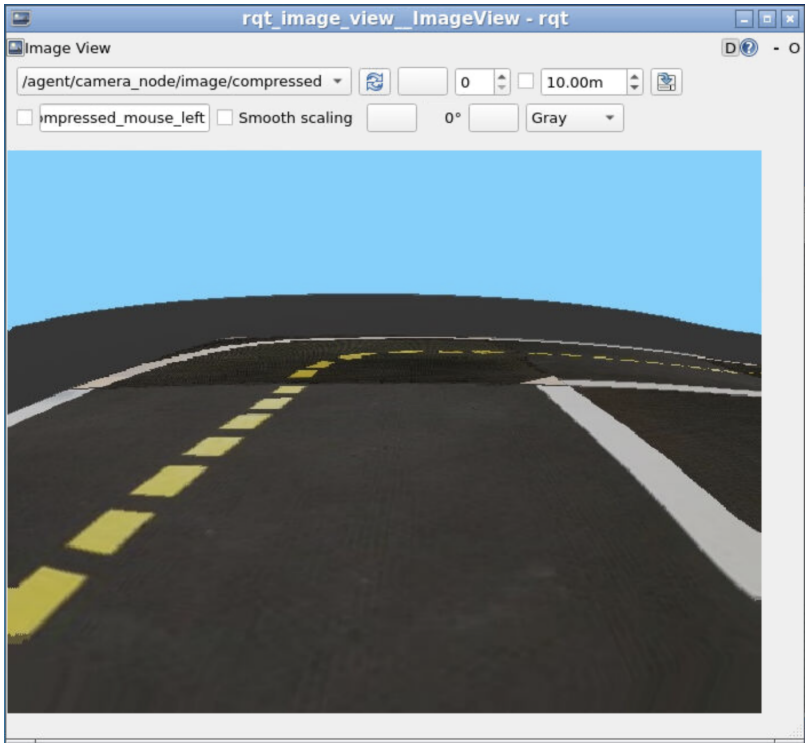
1. Open a terminal on your computer, and type

   ```
   dts code build
   ```
2. Wait for the build to finish, then type:

`dts code workbench --sim`

3. Open VNC on you browser and click on the `VLS - Visual Lane Servoing Exercise` icon on your desktop. Minimize any windows that are opened as a result.

4. Click on the `RQT Image View` icon on your desktop. This will bring up a window like the one below (though you likely won't see an image yet).



RQT Image View GUI.

5. The `RQT Image View` GUI includes a drop-down menu at the top. If you click on this menu, you should see two entries

    - `agent/camera_node/image/compressed`: The original (compressed) image from the Duckiebot's camera
    - `agent/rectifier_node/image/compressed`: The image from the Duckiebot's camera after rectification

6. Compare the appearance of the two images, particularly as it pertains to lines in the scene.

Note: If you don't see an image, try sending a single command to the Duckiebot using the `Joystick` GUI (there is an icon on the desktop) as well as clicking the refresh button to the right of the drop-down menu on the `RQT Image View` GUI.

🚙 Example: Exploring lens distortion on your Duckiebot

In this example, we will visualize the ability to use the results of intrinsic calibration to remove distortions introduced by the lens. This procedure is often referred to as *rectification*.

1. Open a terminal on your computer, and type

   ```
   dts code build
   ```
2. Wait for the build to finish, then type:

   ```
   dts code workbench -b ROBOTNAME
   ```
3. Place your Duckiebot in a lane facing in the direction of travel.

4. Follow the same instructions at point 3 for the simulation-based example above.

Note: Consider placing images with straight lines (e.g., the extrinsic calibration target mentioned below) in front of your Duckiebot's camera to see the effects of distortion.

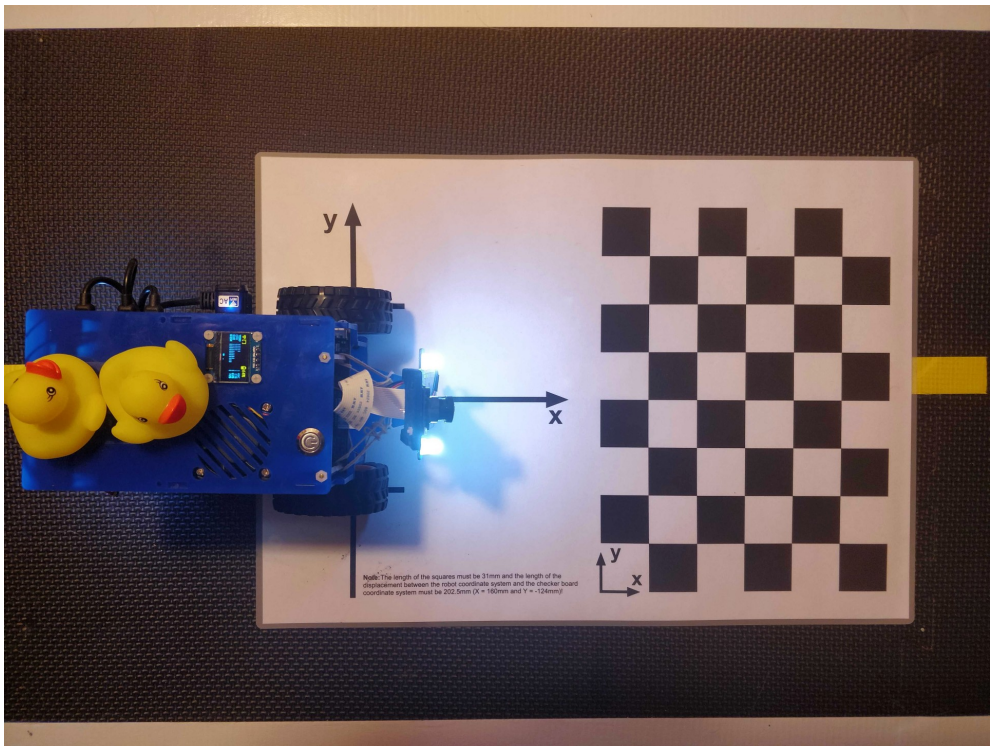
Note: If you don't see an image, try sending a single command to the Duckiebot using the `Joystick` GUI (there is an icon on the desktop) as well as clicking the refresh button to the right of the drop-down menu on the `RQT Image View` GUI.

Extrinsic Calibration

Having estimated the intrinsic parameters of the camera, we are now ready to calulate the extrinsic parameters, namely the homography that relates points in a planar world frame to the image plane. In our case, the world plane will be the ground plane with the origin below the midpoint between the Duckiebot's drive wheels with the positive x-axis pointing forward and the positive y-axis pointing to the left. In this way, world frame moves with the robot.

Much like the intrinsic calibration procedure, we estimate the homography using known correspondences between points expressed in the ground frame and the image, and we will again use a checkerboard pattern for this purpose. Unlike intrinsic calibration, however, we don't need to move the camera or target around since four point-pairs are sufficient given the eight degrees-of-freedom associated with the homography.

🚙 Extrinsic calibration procedure



We calibrate the extrinsic parameters (homography) by placing the Duckiebot on a known checkerboard pattern.

1. Place your powered-on Duckiebot and checkerboard with the wheels aligned with the y-axis on the target and the vehicle centered laterally on the origin, as shown above. Try and keep the field-of-view of the camera as clutter-free as possible.

2. Open a terminal on your computer and type:

```
dts duckiebot calibrate_extrinsics ROBOTNAME
```

Follow the instructions on the screen. If successful, the calibration results will be automatically saved to your Duckiebot.

Final check to make sure it's stored

You can verify that the new gain value has been saved on your Duckiebot by opening the Dashboard > File Manager > config > calibrations > camera_extrinsic page.

You should find a file named `ROBOTNAME.yaml` in addition to the `default.yaml`. Double click on it and verify that the gain value is indeed the one you chose.

You can now move to the image filtering tutorial.