Proposed solution to the robocup object detection problem

Elorm Avevor

December 2022

1 The problem

Detect and **locate** different objects of interest in a maze. This capability has to be an autonomous background service for teleop or autonomous robots. Some items will be in the maze multiple times (e.g. hazmat signs), others rather rare (e.g. door). Show the detection live in the GUI and identify the object by name, e.g. "Oxidizer". Points will be adjusted accordingly (for example, hazmats might get 1 point, doors 5 points). The points are doubled if the identified object is marked at the correct location in the map (within 1 m of the true position). Points will be deducted for wrongly detected or located objects. The identified objects in the map have to be numbered and colored according to the standard (see Mapping Capabilities and 2D Map Format). There has to be a corresponding text file with the object number, the object type, the time they were found and location. Objects to be recognized in 2022:

- Fire Extinguisher
- Door
- Hazmat Signs
- QR codes (read the QR code)
- Valves
- Baby dolls
- Exit signs; green (according to the standard of the country the competition is held in; 2019 for example: Austrailian exit sign)
- Fire Extinguisher signs; red (according to the standard of the country the competition is held in; 2019 for example Austrailian fire extinguisher sign)
- Heat source

2 Proposed solution

1. Detect the object locations and size in the image using YOLOv5 network + a QR code detector.
2. Find the positions in 3D space of the of the centres of the bounding boxes of the objects. Also estimate measurement noise covariance.
3. For planar objects also record their normals.
4. Make a list of known object locations.
5. For a new observation, if there's a high probability that it's already been seen, update that object's state (position, orientation, etc.) using a Kalman filter. Otherwise, add the new observation to the list of known objects.

6. Objects that are only seen for a few frames then not for a long time get deleted.

7. Once an object has been seen for enough frames, its position is recorded in a file.

Here's a video of the current detector in action.

2.1 Yolo detection

Yolo detection is done with OpenCV using Intel's OpenVino library. This allows for fast detection on the integrated GPU while the CPU can be used for other tasks.

The subtasks are as follows:

- Load the net.
- Pre-process the data – converting the colour images from 640×480 to the network's input size by adding a border and/or scalint the image.
- Detection

First 2 subtasks are fairly boring and well documented on the OpenCV website.

Detection

OpenCV passes the data to the iGPU which then preforms the inference and returns an array of bytes – these data then need to be interpreted correctly to recover the bounding boxes.
The data look like the following:

$$\left\{x^{(i)}, y^{(i)}, w^{(i)}, h^{(i)}, \text{conf}^{(i)}, p_0^{(i)}, \ldots, p_{M-1}^{(i)}\right\}_{i=0}^{N-1}$$

Where N is some large number set by the YOLO model and M is the number of classes and p_j is the probability that the object is class j given there is an object with the given bounding box. All values are 32 bit floating point numbers.

To perform detection look at the first set of data $(M + 5)$ floats. Extract the position (x, y), size(w, h) and confidence of that detection. If the confidence (probability that there's an object with that bounding box) is too low, go to the next set of $(M + 5)$ floats and repeat. If the confidence is high enough, find the class (fire extinguisher, person, explosive sign etc.). Each of these classes were given a unique index (between 0 and M - 1) during training. The class id is given by the class with the highest probability:

$$\text{idx} = \arg\max_j p_j$$

If the conditional probability p_{idx} is too low, that means it is likely there is an object there but its type is unclear. If the probability is big enough, the object is recorded.

For hazmat signs, the distribution $\{p_j | j \in \text{hazmat indexes}\}$ is recorded. [1].

After doing this N times, you'll have a list of bounding boxes but some of them may be approx duplicates. To remove duplicates, non-maximum suppression is carried out (if 2 bounding boxes look similar i.e. they have areas of intersection close to their areas of union, the one with the highest confidence is used). This is done in a class-agnostic way so a fire extinguisher may be deleted if a flammable sign is detected at the same place with the same size.

At this point the detection is complete.

2.2 QR code detection

Nothing much of note – it uses a library called Zbar.

[1] Note this distribution likely won't sum to 1 as there's a probability that the bounding box doesn't contain a hazmat sign

2.3 Finding the object positions

Once the centre of the object is found, the depth of that pixel is easily found using the depth image. The point's centre can be found relative to the depth camera using a function provided by Intel's RealSense library. There's then a rigid body transform (rotation and translation) to find the point relative to the tracking camera's frame. The translation and rotation of the tracking camera relative to its initial position can be used to find the object's position relative to the world.

2.4 Finding the covariance matrix

A measurement is pretty useless without any estimate of its accuracy. The sources of measurement noise are:

- Errors in depth measurement (assumed negligible) as depths are limited to 2m.
- Errors in the tracking camera orientation (assumed negligible for ease of calculation)
- Errors in the the tracking camera position
- The centre of the bounding box won't be the centre of the object (The centre may not even be a well defined concept).

The tracking camera's position covariance is provided with every measurement from the tracking camera.

The errors in the centre of the object are given by hand-tuned covariance matrices. E.g. The error in a door's centre is large vertically but relatively small normal to the door.

The total covariance is the sum of the covariances from each error source.

2.5 Normals to surfaces

Objects come with a bounding box. Sample N points in the bounding box and find their position relative to the camera. Normals are assumed to have a strictly negative z coordinate relative to the camera (i.e. the normals have a component facing the camera).

The equation of the plane can be written as:

$$c = a_1 x + a_2 y - z$$

However the plane's location isn't relevant – just the normal. Ordinary least squares will be used to estimate the plane. That estimator is unbiased, meaning that if all xs, ys, and zs have their averages subtracted, the new value of c will be 0 and the plane equation can be simplified to:

$$z = a_1 x + a_2 y$$

A simple solution to this plane fitting is least squares fitting minimising $\sum |z_i - a_1 x_i - a_2 y_i|^2$. The solution is:

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \mathrm{Cov}\,(x,x) & \mathrm{Cov}\,(x,y) \\ \mathrm{Cov}\,(y,x) & \mathrm{Cov}\,(y,y) \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{Cov}\,(x,z) \\ \mathrm{Cov}\,(y,z) \end{bmatrix}$$

Where $\mathrm{Cov}\,(A,B) = \frac{1}{N}\sum a_i b_i$ in the case where A and B have 0 mean.

With sensible sampling and pre-processing, x and y would be 0 mean and independent in which case $\mathrm{Cov}\,(x,y) = \mathrm{Cov}\,(y,x) = 0$ which would simplify the problem to:

$$a_1 \approx \frac{\mathrm{Cov}\,(x,z)}{\sigma_x^2}$$
$$a_2 \approx \frac{\mathrm{Cov}\,(y,z)}{\sigma_y^2}$$

But doing the full calculation isn't computationally burdensome so the simplified fitting isn't used.

The normal is then given by:

$$\frac{1}{\sqrt{a_1^2 + a_2^2 + 1}} \begin{bmatrix} a_1 \\ a_2 \\ -1 \end{bmatrix}$$

2.5.1 Kalman filter

For static objects, the Kalman filter is relatively easy to compute: The state transition equation is given by:

$$X_{n+1} = X_n$$

Where X is the unknown real position of the object.

The observation equation is given by:

$$O_n = X_n + N_n$$

Where N is the measurement noise which has covariance R and O is the observed position of the object.

The Kalman update step is then:

$$\hat{X}_{n+1} = \hat{X}_n + \hat{S}_n \left(\hat{S}_n + R_n\right)^{-1} (O_n - X_n)$$
$$\hat{S}_{n+1} = \hat{S}_n - \hat{S}_n \left(\hat{S}_n + R_n\right)^{-1} \hat{S}_n$$

Where $\hat{X}_n$ is the best guess of the object's position and $\hat{S}_n$ represents the covariance in that best guess.

2.6 Matching objects over frames

Assuming everything is Gaussian, the correct metric for matching objects over frames is the Mahalanobis distance:

$$d^2 = \left(\hat{X}_n - O_n\right)^T \left(\hat{S}_n + R_n\right)^{-1} \left(\hat{X}_n - O_n\right)$$

The probability that O and $\hat{X}$ have the same mean is a monotonically decreasing function of d^2. By looking at tables for χ^2 distributions (for 3 dof), a value of $d^2 = 21$ means that the false negative rate is very small.

Sanity check:
For the tracking camera noise is almost always $(10\text{cm})^2 I$ (unless it gets lost). The noise in the centre of a hazmat sign is small compared to 10 cm so a good approximation is $R = (10\text{cm})^2 I$. If the hazmat sign has been seen for a lot of frames, $\hat{S}_n$ will go to 0 leaving: $d^2 = \left|\frac{\hat{X}_n - O_n}{10\text{cm}}\right|^2$. So the condition for rejecting the object is, $\left|\hat{X}_n - O_n\right| > \sqrt{21} \times 10\text{cm} \approx 46\text{cm}$. So if an observed hazmat is found 46 cm away from all known hazmats, it will be assumed to be a new hazmat. Redoing the calculation for $\hat{S}_n = R$ (when the object is first observed) gives $\left|\hat{X}_n - O_n\right| > \sqrt{42} \times 10\text{cm} = 65\text{cm}$. This is rather conservative but the noise isn't Gaussian (e.g. the tracking camera position is susceptible to small drift) and it was decided that double counting an object is worse than missing one.

Matching is done by finding the pair with the smallest Mahalanobis distance and matching them if their distance is below the threshold. That object and observation is no longer allowed to be matched again. Keep repeating until the closest pairs are above the threshold. All remaining observations are added to the list of known objects.

Experiments show that there's a significant probability of hazmat signs being misclassified as other classes of hazmat sign e.g. flammable sign being misclassified as spontaneously combustible. Fortunately, the YOLO model provides the conditional probabilities of each class so if a hazmat sign is matched over N different images, the distribution of that hazmat sign can estimated as the average of the N distributions.

If an object was only seen for a few frames then not again for a lot of frames, it is deleted as it was probably found in error.

3 Not yet implemented

The following improvements haven't been implemented either because they're unlikely to significantly improve detection or the hardware isn't available to test them.

3.1 Using normals

It's possible that 2 hazmat signs are on opposite corners of a wall. They could be close to each other but never be in frame at the same time. This would cause the algorithm to think those were the same hazmat sign and get the detection wrong. To rectify this, objects can be rejected if the dot product between the observation's normal and the known object's normal is less than a threshold.

3.2 Finding the axis of fire extinguishers (approximated as a cylinder)

This isn't necessary for the task but is an interesting problem given the constraints. Currently, the bounding boxes aren't very tight, so only a small patch of pixels near the fire extinguisher's centre is used. Take that patch and split it into 16 smaller patches. Find the normals to the patch at each of those 16 locations. Use least squares to find the axis which minimises the sum of the dot products between each normal and the axis. This isn't very good if the patch is small and noisy as if all normals are in the same plane, the axis is easily corrupted by noise.

It is possible to find calculate the best fit cylinder by minimising the sum of square distances to that cylinder however, as far as I can tell, that function is non-linear and can have multiple local minima so optimising would be difficult.

3.3 Covariance in tracking camera orientation

The tracking camera pose messages come with estimates of the covariance of the angles relative to each axis $(\theta_x, \theta_y, \theta_z)$. It's unclear how these angles are defined (e.g. are the rotations applied in a certain order? Which order). If you could figure that out, rather than using a quaternion to describe the camera's pose, you could use those 3 angles. There would then be a function $(X, Y, Z) = f(\theta_x, \theta_y, \theta_z)$ that finds the position of the pixel in question relative to the 3 angles. With that, a jacobian matrix can be calculated:

$$J = \begin{pmatrix} \frac{\partial X}{\partial \theta_x} & \frac{\partial X}{\partial \theta_y} & \frac{\partial X}{\partial \theta_z} \\ \frac{\partial Y}{\partial \theta_y} & \frac{\partial Y}{\partial \theta_y} & \frac{\partial Y}{\partial \theta_z} \\ \frac{\partial Z}{\partial \theta_x} & \frac{\partial Z}{\partial \theta_y} & \frac{\partial Z}{\partial \theta_z} \end{pmatrix}$$

For small errors in angle, the corresponding error in position is given by:

$$\begin{pmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{pmatrix} = J \begin{pmatrix} \Delta\theta_x \\ \Delta\theta_y \\ \Delta\theta_z \end{pmatrix}$$

The covariance is given by

$$\mathbb{E}\left[\begin{pmatrix}\Delta X & \Delta Y & \Delta Z\end{pmatrix}^T \begin{pmatrix}\Delta X & \Delta Y & \Delta Z\end{pmatrix}\right] = J\mathbb{E}\left[\begin{pmatrix}\Delta\theta_x & \Delta\theta_y & \Delta\theta_z\end{pmatrix}^T \begin{pmatrix}\Delta\theta_x & \Delta\theta_y & \Delta\theta_z\end{pmatrix}\right] J^T$$

The inner expectation is just the covariance matrix that comes with the tracking camera pose message.

Note that the Jacobian matrix will be singular or ill-conditioned (you can rotate the camera while pointing in the same direction) so it must be used alongside other sources of noise.

3.4 Covariance from depth

The depth camera has errors which vary with distance. Previously, they were assumed to be negligible however for a perhaps more accurate model, a model of $\sigma_d(d)$ could be made to estimate depth accuracy. There would be a ray of direction $\mathbf{r}$ from the camera to the centre of the object. The error in (X, Y, Z) due to the depth inaccuracy is $\mathbf{r}\Delta d$ so the covariance is then $\sigma_d^2(d)\mathbf{r}\mathbf{r}^T$.

Again, this is a singular (rank 1) or ill-conditioned matrix so other sources of error must also be considered.

3.5 Tracking move-able people

So far all objects have been static. For a person moving, it may be sufficient to slightly modify the Kalman state transition matrix to:

$$\begin{aligned} X_{n+1} &= X_n + \Delta X_n \\ O_n &= X_n + N_n \end{aligned}$$

Where X_n is a random change in position with 0 mean and covariance (Q_n) estimated from walking humans. Hopefully that's a sufficiently accurate model for tracking and the person's velocity needn't be included in the state X. However, by not tracking the velocity, the position covariance will never tend to 0.

The Kalman update step would be:

$$\begin{aligned} \hat{X}_{n+1} &= \hat{X}_n + \left(\hat{S}_n + Q_n\right)\left(\hat{S}_n + Q_n + R_n\right)^{-1}(O_n - X_n) \\ \hat{S}_{n+1} &= \hat{S}_n + Q_n - \left(\hat{S}_n + Q_n\right)\left(\hat{S}_n + Q_n + R_n\right)^{-1}\left(\hat{S}_n + Q_n\right) \end{aligned}$$

These people would then need to be deleted if they're out of frame for a few seconds. As far as I can tell, double counting would be inevitable.

3.6 Detecting the remaining object types

Many object types aren't being detected. It is quite a time consuming process to prepare the data and train the detector.