CHAPTER 15

Registration

Computing a transformation that aligns an image or a depth map or a set of images with another such is generally known as *registration*. One approach to registration is to abstract the image (etc.) as a set of points, yielding the following general problem. Assume we know a set of N reference points in d dimensions. We observe M points in d dimensions, and these observed M points are obtained by transforming the reference points with some transformation and adding noise, then dropping some points and including some pure noise points. The two sets of points are often referred to as *point clouds*. We want to determine the transformation from the two point clouds.

This problem occurs in a wide range of practical applications. As we shall see, calibrating a camera involves solving a version of this problem (Section 32.2). Determining where you are in a known map very often involves solving a version of this problem. Imagine, for example, a camera looking directly downwards from an aircraft flying at fixed height. The image in the camera translates and rotates as the aircraft moves. If we can compute the transformation from image i to image $i+1$, we can tell how the aircraft has moved. Another useful case occurs when we have a depth map of a known object and want to compute the *pose* of the object (its position and orientation in the frame of the depth sensor). We could do so by having reference points on the object, finding interest points in the depth map, then solving for a transformation that maps reference points to depth points.

How one approaches this class of problem depends on three important factors.

- **Correspondence:** if it is known *which* observation corresponds to *which* reference point, the problem is relatively straightforward to solve (unless there are unusual noise effects). This case is uncommon, but does occur. In robotics, *beacons* are objects that identify themselves (perhaps by wearing a barcode; by transmitting some code; by a characteristic pattern) and can be localized. They are useful, precisely because they yield correspondence and so simplify computing the transformation. If correspondence is not known, which is the usual case, computing the transformation becomes rather harder.

- **Transformation:** there are closed form solutions for known correspondence and Euclidean or affine transformations. Homographies (and higher dimensional analogs) do not admit closed form transformations.

- **Noise:** computing a transformation can become very hard if many of the observations do not come from reference points, if many of the reference points are dropped, or if some observations are subject to very large noise effects.

15.1 REGISTRATION WITH KNOWN CORRESPONDENCE AND GAUSSIAN NOISE

15.1.1 Affine Transformations and Gaussian Noise

In the simplest case, the correspondence is known – perhaps the reference points are beacons – and the only noise is Gaussian (so $N = M$). Write $\mathbf{x}_i$ for the i'th observation and $\mathbf{y}_i$ for the i'th reference point. We will assume the noise is isotropic, which is by far the most usual case. Once you have followed this derivation, you will find it easy to incorporate a known covariance matrix. We have

$$\mathbf{x}_i = \mathcal{M}\mathbf{y}_i + \mathbf{t} + \xi_i$$

where ξ_i is the value of a normal random variable with mean $\mathbf{0}$ and covariance matrix $\Sigma = \sigma^2\mathcal{I}$. A natural procedure to estimate $\mathcal{M}$ and $\mathbf{t}$ is to maximize the likelihood of the noise. Because it will be useful later, we assume that there is a weight w_i for each pair, so the negative log-likelihood we must minimize is proportional to

$$\sum_i w_i \left(\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right)^T \left(\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right)$$

(the constant of proportionality is σ^2, which doesn't affect the optimization problem). The gradient of this cost with respect to $\mathbf{t}$ is

$$-2\sum_i w_i \left(\mathbf{x}_i - \mathcal{M}\mathbf{y}_i - \mathbf{t}\right)$$

which vanishes at the solution. In turn, if $\sum_i w_i\mathbf{x}_i = \sum_i w_i\mathcal{M}\mathbf{y}_i$, $\mathbf{t} = \mathbf{0}$. One straightforward way to achieve this is to ensure that both the observations and the reference points have a center of gravity at the origins. Write

$$\mathbf{c}_x = \frac{\sum_i w_i\mathbf{x}_i}{\sum_i w_i}$$

for the center of gravity of the observations (etc.) Now form

$$\mathbf{u}_i = \mathbf{x}_i - \mathbf{c}_x \text{ and } \mathbf{v}_i = \mathbf{y}_i - \mathbf{c}_y$$

and if we use $\mathbf{u}_i$ as observations and $\mathbf{v}_i$ as reference points, then the translation will be zero. In turn, the translation from the original reference points to the original observations is $\mathbf{c}_x - \mathbf{c}_y$.

We obtain $\mathcal{M}$ by minimizing

$$\sum_i w_i \left(\mathbf{u}_i - \mathcal{M}\mathbf{v}_i\right)^T \left(\mathbf{u}_i - \mathcal{M}\mathbf{v}_i\right).$$

Now write $\mathcal{W} = \mathsf{diag}\left([w_1, \ldots, w_N]\right)$, $\mathcal{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]$ (and so on). You should check that the objective can be rewritten as

$$\mathsf{Tr}\left(\mathcal{W}(\mathcal{U} - \mathcal{M}\mathcal{V})^T(\mathcal{U} - \mathcal{M}\mathcal{V})\right).$$

Now the trace is linear; $\mathcal{U}^T\mathcal{U}$ is constant; and we can rotate matrices through the trace (Section 32.2). This means the cost is equivalent to

$$\mathsf{Tr}\left(-2\mathcal{M}\mathcal{V}\mathcal{U}^T + \mathcal{M}^T\mathcal{M}\mathcal{V}\mathcal{V}^T\right)$$

which will be minimized when

$$\mathcal{M}\mathcal{V}\mathcal{W}\mathcal{V}^T = \mathcal{V}\mathcal{W}\mathcal{U}^T$$

(which you should check). Many readers will recognize a least squares solution here. The trace isn't necessary here, but it's helpful to see an example using the trace, because it will be important in the next case.

15.1.2 Euclidean Motion and Gaussian Noise

One encounters affine transformations relatively seldom in practice, though they do occur. Much more interesting is the case where the transformation is Euclidean. The least squares solution above isn't good enough, because the $\mathcal{M}$ obtained that way won't be a rotation matrix. But we can obtain a least squares solution with a rotation matrix, using a neat trick. We adopt the notation of the previous section, and change coordinates from $\mathbf{x}_i$ to $\mathbf{u}_i$ as above to remove the need to estimate translation.

We must choose $\mathcal{R}$ to minimize

$$\sum_i w_i(\mathbf{u}_i - \mathcal{R}\mathbf{v}_i)^T(\mathbf{u}_i - \mathcal{R}\mathbf{v}_i).$$

This can be done in closed form (a fact you should memorize). Equivalently, we must minimize

$$\begin{aligned}\sum_i w_i(\mathbf{u}_i - \mathcal{R}\mathbf{v}_i)^T(\mathbf{u}_i - \mathcal{R}\mathbf{v}_i) &= \mathsf{Tr}\left(\mathcal{W}(\mathcal{U} - \mathcal{R}\mathcal{V})(\mathcal{U} - \mathcal{R}\mathcal{V})^T\right) \\ &= \mathsf{Tr}\left(-2\mathcal{U}\mathcal{W}\mathcal{V}^T\mathcal{R}^T\right) + K \\ &\quad \text{(because } \mathcal{R}^T\mathcal{R} = \mathcal{I}\text{)} \\ &= -2\mathsf{Tr}\left(\mathcal{R}\mathcal{U}\mathcal{W}\mathcal{V}^T\right)\end{aligned}$$

Now we compute an SVD of $\mathcal{U}\mathcal{V}^T$ to obtain $\mathcal{U}\mathcal{W}\mathcal{V}^T = \mathcal{A}\mathcal{S}\mathcal{B}^T$ (where $\mathcal{A}$, $\mathcal{B}$ are orthonormal, and $\mathcal{S}$ is diagonal – Section 32.2 if you're not sure). Now $\mathcal{B}^T\mathcal{R}\mathcal{A}$ is orthonormal, and we must maximize $\mathsf{Tr}\left(\mathcal{B}^T\mathcal{R}\mathcal{A}\mathcal{S}\right)$, meaning $\mathcal{B}^T\mathcal{R}\mathcal{A} = \mathcal{I}$ (check this if you're not certain), and so $\mathcal{R} = \mathcal{B}\mathcal{A}^T$.

Procedure: 15.1 *Weighted Least Squares for Euclidean Transformations*

We have N reference points $\mathbf{x}_i$ whose location is measured in the agent's coordinate system. Each corresponds to a point in the world coordinate system with known coordinates $\mathbf{y}_i$, and the change of coordinates is a Euclidean transformation (rotation $\mathcal{R}$, translation $\mathbf{t}$). For each $(\mathbf{x}_i, \mathbf{y}_i)$ pair, we have a weight w_i. We wish to minimize

$$\sum_i w_i(\mathbf{x}_i - \mathcal{R}\mathbf{y}_i - \mathbf{t})^T(\mathbf{x}_i - \mathcal{R}\mathbf{y}_i - \mathbf{t})$$

Write

$$\begin{aligned}\mathbf{c}_x &= \frac{\sum_i w_i\mathbf{x}_i}{\sum_i w_i} \\ \mathbf{c}_y &= \frac{\sum_i w_i\mathbf{y}_i}{\sum_i w_i} \\ \mathbf{u}_i &= \mathbf{x}_i - \mathbf{c}_x \\ \mathbf{v}_i &= \mathbf{y}_i - \mathbf{c}_y\end{aligned}$$

Then the least squares estimate $\hat{\mathbf{t}}$ of $\mathbf{t}$ is

$$\hat{\mathbf{t}} = \mathbf{c}_x - \mathbf{c}_y$$

Write $\mathcal{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N]$ (etc); $\mathcal{W} = \text{diag}(w_1, \ldots, w_N)$; and $\text{SVD}(\mathcal{U}\mathcal{S}\mathcal{V}) = \mathcal{A}\Sigma\mathcal{B}^T$. The least squares estimate $\hat{\mathcal{R}}$ is

$$\hat{\mathcal{R}} = \mathcal{B}\mathcal{A}^T$$

15.1.3 Homographies and Gaussian Noise

We now work with $d = 2$, and allow the transformation to be a homography. Solving for a homography requires solving an optimization problem, but estimating a homography from data is useful, and relatively easy to do. Furthermore, we can't recover the translation component from centers of gravity (exercises **TODO:** homography exercise). In all cases of interest, the points $\mathbf{x}_i$ and $\mathbf{y}_i$ will be supplied in affine coordinates, rather than homogeneous coordinates, and we convert to homogeneous coordinates by attaching a 1, as before. Write m_{ij} for the i, j'th element of matrix $\mathcal{M}$. In affine coordinates, a homography $\mathcal{M}$ will map $\mathbf{y}_i = (y_{i,x}, y_{i,y})$ to $\mathbf{x}_i = (x_{i,x}, x_{i,y})$ where

$$x_{i,x} = \frac{m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13}}{m_{31}x_{i,x} + m_{32}x_{i,y} + m_{33}} \text{ and } x_{i,y} = \frac{m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23}}{m_{31}x_{i,x} + m_{32}x_{i,y} + m_{33}}$$

Write $\mathcal{M}(\mathbf{y})$ for the result of applying the homography to $\mathbf{y}$, *in affine coordinates*. In most cases of interest, the coordinates of the points are not measured precisely, so

we observe $\mathbf{x}_i = \mathcal{M}(\mathbf{y}_i) + \xi_i$, where ξ_i is some noise vector drawn from an isotropic normal distribution with mean $\mathbf{0}$ and covariance Σ.

The error will be in affine coordinates – for example, in the image plane – which justifies working in affine rather than homogeneous coordinates. Again, we assume that the noise is isotropic, and so that $\Sigma = \sigma^2\mathcal{I}$. The homography can be estimated by minimizing the negative log-likelihood of the noise, so we must minimize

$$\sum_i w_i \xi_i^T \xi_i$$

where

$$\xi_i = \begin{bmatrix} x_{i,x} - \frac{m_{11}y_{i,x}+m_{12}y_{i,y}+m_{13}}{m_{31}y_{i,x}+m_{32}y_{i,y}+m_{33}} \\ x_{i,y} - \frac{m_{21}y_{i,x}+m_{22}y_{i,y}+m_{23}}{m_{31}y_{i,x}+m_{32}y_{i,y}+m_{33}} \end{bmatrix}$$

using standard methods (Levenberg-Marquardt is favored; Chapter 32.2). This approach is sometimes known as *maximum likelihood* . Experience teaches that this optimization is not well behaved without a strong start point.

There is an easy construction for a good start point. Notice that the equations for the homography mean that

$$x_{i,x}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13} = 0$$

and

$$x_{i,y}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23} = 0$$

so each corresponding pair of points $\mathbf{x}_i$, $\mathbf{y}_i$ yields two *homogeneous* linear equations in the coefficients of the homography. They are homogeneous because scaling $\mathcal{M}$ doesn't change what it does to points (check this if you're uncertain). If we obtain sufficient points, we can solve the resulting system of homogeneous linear equations. Four point correspondences yields an unambiguous solution; more than four – which is better – can be dealt with by least squares (exercises **TODO:** fourpoint homography). The resulting estimate of $\mathcal{M}$ has a good reputation as a start point for a full optimization.

Procedure: 15.2 *Estimating a Homography from Data*

Given N known source points $\mathbf{y}_i = (y_{i,x}, y_{i,y})$ in affine coordinates and N corresponding target points $\mathbf{x}_i$ with measured locations $(x_{i,x}, x_{i,y})$ and where measurement noise has zero mean and covariance $\Sigma = \sigma^2\mathcal{I}$, estimate the homography $\mathcal{M}$ with i, j'th element m_{ij} by minimizing:

$$\sum_i \xi_i^T \xi_i$$

where

$$\xi = \begin{bmatrix} x_{i,x} - \frac{m_{11}y_{i,x}+m_{12}y_{i,y}+m_{13}}{m_{31}y_{i,x}+m_{32}y_{i,y}+m_{33}} \\ x_{i,y} - \frac{m_{21}y_{i,x}+m_{22}y_{i,y}+m_{23}}{m_{31}y_{i,x}+m_{32}y_{i,y}+m_{33}} \end{bmatrix}$$

Obtain a start point by as a least-squares solution to the set of homogeneous linear equations

$$x_{i,x}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{11}y_{i,x} + m_{12}y_{i,y} + m_{13} = 0$$

and

$$x_{i,y}(m_{31}y_{i,x} + m_{32}y_{i,y} + m_{33}) - m_{21}y_{i,x} + m_{22}y_{i,y} + m_{23} = 0.$$

15.1.4 Projective Transformations and Gaussian Noise

A *projective transformation* is the analogue of a homography for higher dimensions. In affine coordinates, a projective transformation $\mathcal{M}$ will map $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,d})$ to $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$ where

$$x_{i,1} = \frac{m_{11}y_{i,1} + \ldots + m_{1d}y_{i,d} + m_{1(d+1)}}{m_{(d+1)1}y_{i,1} + \ldots + m_{(d+1)d}y_{i,d} + m_{(d+1)(d+1)}}$$

and

$$x_{i,d} = \frac{m_{d1}y_{i,1} + \ldots + m_{dd}y_{i,d} + m_{d(d+1)}}{m_{(d+1)1}y_{i,1} + \ldots + m_{(d+1)d}y_{i,d} + m_{(d+1)(d+1)}}$$

Estimating this transformation follows the recipe for a homography, but there are now more parameters. I have put the result in a box, below.

Procedure: 15.3 *Estimating a Projective Transformation from Data*

Given N known source points $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,d})$ in affine coordinates and N corresponding target points $\mathbf{x}_i$ with measured locations $(x_{i,1}, \ldots, x_{i,d})$ and where measurement noise has zero mean and is isotropic, the homography $\mathcal{M}$ with i, j'th element m_{ij} by minimizing:

$$\sum_i \xi_i^T \Sigma^{-1} \xi_i$$

where

$$\xi_i = \begin{bmatrix} x_{i,1} - \frac{m_{11}y_{i,1}+\ldots+m_{1d}y_{i,d}+m_{1(d+1)}}{m_{(d+1)1}x_{i,1}+\ldots+m_{(d+1)d}x_{i,d}+m_{(d+1)(d+1)}} \\ \ldots \\ x_{i,d} - \frac{m_{d1}y_{i,1}+\ldots+m_{dd}y_{i,d}+m_{d(d+1)}}{m_{(d+1)1}x_{i,1}+\ldots+m_{(d+1)d}x_{i,d}+m_{(d+1)(d+1)}} \end{bmatrix}$$

Obtain a start point by as a least squares solution to the set of homogeneous linear equations

$$\begin{aligned} 0 &= x_{i,1}(m_{(d+1)1}y_{i,1} + \ldots + m_{(d+1)d}y_{i,d} + m_{(d+1)(d+1)}) - \\ &\quad m_{11}y_{i,1} + \ldots + m_{1d}y_{i,d} + m_{1(d+1)} \\ &\ldots \\ 0 &= x_{i,d}(m_{(d+1)1}y_{i,1} + \ldots + m_{(d+1)d}y_{i,d} + m_{(d+1)(d+1)}) - \\ &\quad m_{(d+1)1}x_{i,1} + \ldots + m_{(d+1)d}x_{i,d} + m_{(d+1)(d+1)} \end{aligned}$$

15.2 UNKNOWN CORRESPONDENCE

15.2.1 Unknown Correspondence and ICP

Now assume correspondences are not known, and some reference (resp. observed) points may not even have corresponding observed (resp. reference) points. We have N reference points $\mathbf{y}_i$ and M observed points $\mathbf{x}_i$. For the moment, we will assume that all weights w_i are 1. A straightforward, and very effective, recipe for registering the points is *iterative closest points* or *ICP*. The key insight here is that, if the transformation is very close to the identity, then the $\mathbf{y}_{c(i)}$ that corresponds to $\mathbf{x}_i$ should be the closest reference point to $\mathbf{x}_i$. This finding the closest reference point to each measurement and computing the transformation using that correspondence. But the transformation might not be close to the identity, and so the correspondences might change. We could repeat the process until they stop changing.

Formally, start with a transformation estimate $\mathcal{T}_1$, a set of $\mathbf{m}_i^{(1)} = \mathcal{T}^{(1)}(\mathbf{y}_i)$ and then repeat two steps:

- **Estimate correspondences** using the transformation estimate. Then, for each $\mathbf{x}_i$, we find the closest $\mathbf{m}^{(n)}$ (say $\mathbf{m}_c^{(n)}$); then $\mathbf{x}_i$ corresponds to $\mathbf{m}_{c(i)}^{(n)}$.

- **Estimate a transformation** $\mathcal{T}^{(n+1)}$ using the corresponding pairs. This maps each $\mathbf{m}_{c(i)}^{(n)}$ to its corresponding $\mathbf{x}_i$.

These steps are repeated until convergence, which can be tested by checking if the correspondences don't change or if $\mathcal{T}^{(n+1)}$ is very similar to the identity. The require transformation is then

$$\mathcal{T}^{(n+1)} \circ \mathcal{T}^{(n)} \circ \ldots \mathcal{T}^{(1)}$$

There are a number of ways in which this very useful and very general recipe can be adapted. First, if there is any description of the points available, it can be used to cut down on correspondences (so, for example, we match only red points to red points, green points to green points, and so on). Second, finding an exact nearest neighbor in a large point cloud is hard and slow, and we might need to subsample the point clouds or pass to approximate nearest neighbors (more details below). Third, points that are very far from the nearest neighbor might cause problems, and we might omit them (again, more details below).

15.2.2 ICP and Sampling

One particularly useful application of ICP occurs when one wishes to register a mesh to a set of points. For example, you might want to register a cloud of measured points to a mesh model of an object built using a CAD modelling system. A natural procedure is to sample points on the mesh model to get a point cloud, then treat the problem using ICP. Another useful application is when one has two mesh models, where the triangulation of the meshes might not be the same. In this case, you could sample both meshes to end up with two point clouds, then register the point clouds. How one samples the mesh or meshes is important.

The ICP recipe becomes difficult to apply to point clouds when M or N are very large. One obvious strategy to control this problem applies when something else – say, a color measurement – is known about each point. For example, we might get such data by using a range camera aligned with a conventional camera, so that every point in the depth map comes with a color. When extra information is available, one searches only compatible pairs for correspondences.

Large point clouds are fairly common in autonomous vehicle applications. For example, the measurements might be LIDAR measurements of some geometry. It is quite usual now to represent that geometry with another, perhaps enormous, point cloud, which you could think of as a map. Registration would then tell the vehicle where it was in the map. Notice that in this application, there is unlikely to be a measurement that exactly corresponds to each reference point. Instead, when the registration is correct, every $\mathbf{x}_i$ is very close to some transformed $\mathbf{y}_i$, so a least squares estimate is entirely justified. In cases like this, one can subsample the reference point cloud, the measurement point cloud, or both.

The sampling procedure depends on the application, and can have significant effects. For example, imagine we are working with LIDAR on a vehicle which is currently in an open space next to a wall (Figure **??**). There will be many returns from the wall, and likely few from the open space. Uniformly sampled measurements would still have many returns from the wall, and few from the open space. This

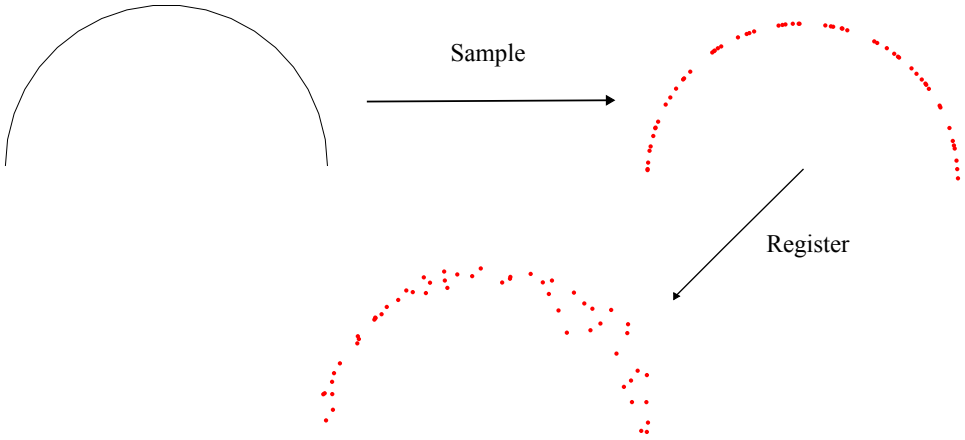

FIGURE 15.1: *In many problems, one has to register a mesh – which might come from a CAD model – to a set of measurements – which might come from LIDAR or from a range camera.* **Top left:** *shows a view of a very simple 1D mesh, in 2D. Registering this mesh to a set of measurements* **bottom** *is a straightforward application of ICP. One samples the mesh* **top left**, *then registers this set of points to the measurements.*

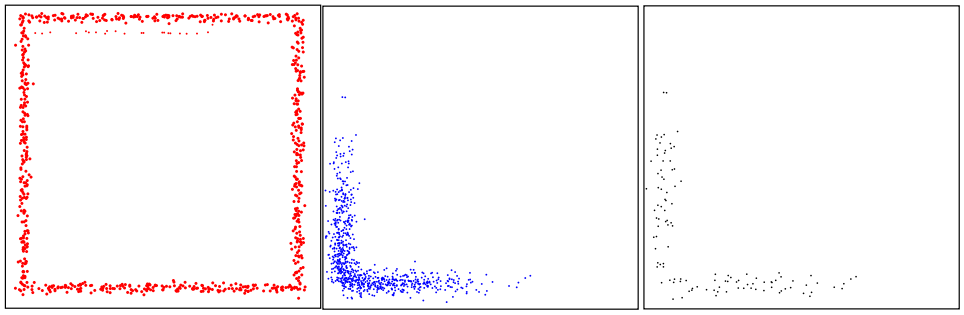

FIGURE 15.2: *On the* **left** *a map of a simple arena, represented as a point cloud. Such a map could be obtained by registering LIDAR measurements to one another. A LIDAR or depth sensor produces measurements in the sensor's coordinate system, and registering these measurements to the map will reveal where the sensor is. However, the sensor may measure points more densely at some positions than at others.* **Left** *shows such a measurement; note the heavy sampling of points near the corner and the light sampling on the edges. This can bias the registration, because the large number of points near the corner mean that the registration error consists mostly of errors from these points. It can also create significant computational problems, because finding the closest points will become slower as the number of points increases. A stratified sample of the measurements (***right***) is obtained by dividing the plane (in this case) into cells of equal area (usually a grid), then resampling the measurements at random* so there are no more than a fixed number of samples in each box. *Such a sample can both reduce bias and improve the speed of registration.* **TODO:** Source, Credit, Permission

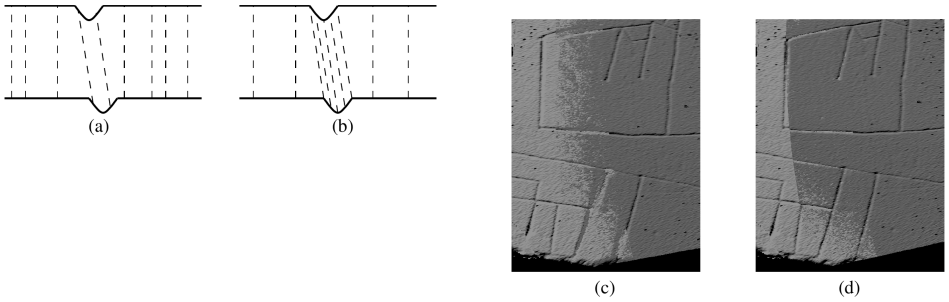

FIGURE 15.3: *The sample of points used in registration can be biased in useful ways. For example, (a) shows a cross section of a flat surface with a small groove (***above***) which needs to be registered to a similar surface (***below***). If point samples are drawn on the surface at random, then there will be few samples in the groove; the dashed lines indicate correspondences. In turn, the registration will be poor, because the surfaces can slide on one another. In (b), the samples have been drawn so that normal directions are evenly represented in the samples. Notice this means more samples concentrated in the groove, and fewer on the flat part. As a result, the surface is less free to slide, and the registration improves.*
TODO: what do c and d show? **TODO:** Source, Credit, Permission

could bias the estimate of the vehicle's pose. A better alternative would be to build a *stratified sample* by breaking the space around the vehicle into blocks of fixed size, then choosing uniformly at random a fixed number of samples in each block. In this scheme, the wall would be undersampled, and the open space would be oversampled, somewhat resolving the bias.

Another stratified sampling strategy is to ensure that surface normal directions are evenly represented in the samples. Make an estimate of a surface normal at each point (for example, by fitting a plane to the point and some of its nearest neighbors). Now break the unit sphere, which encodes the surface normals, into even cells, and sample the points so that each cell has the same number of samples. This approach is particularly useful when we are trying to register flat surfaces with small relief details on them (Figure **??**).

15.2.3 ICP: Finding Nearest Neighbors

Finding the exact nearest neighbor of a query point in a large collection of reference points is more difficult than most people realize (one can beat linear search, but by only a very small factor []). However, finding a point that has high probability of being almost as close as the nearest neighbor (an *approximate nearest neighbor*) can be done rather fast using a variety of approximation schemes []. It is usual to substitute an approximate nearest neighbor, found using a k-d tree (eg []).

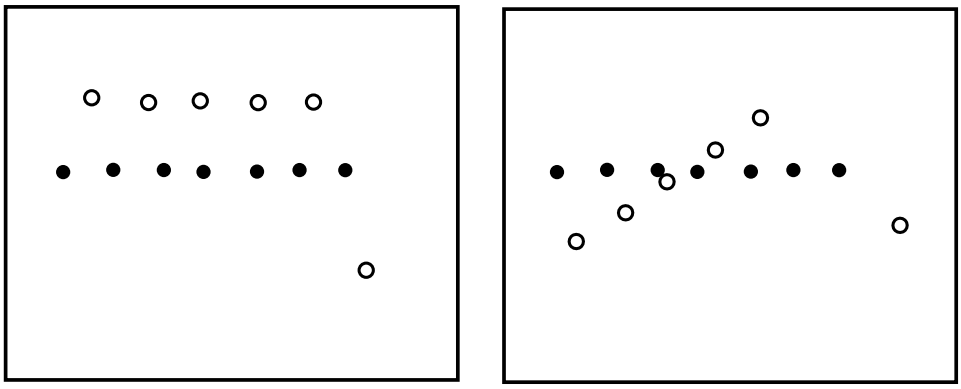

FIGURE 15.4: *Significant registration errors can be caused by just one point that is in the wrong place. On the* **left***, a set of empty points must be registered to a set of filled points. Notice that one empty point is badly out of place. An ideal registration would ignore this, and put the approximate line of empty points on the line of filled points. On the* **right***, the registration that actually results. The square of a large number is very large, meaning the minimum of the squared error isn't where you might think; reducing the large offset entirely justifies a set of medium sized errors. Points that lie significantly far from their "natural" positions are often known as* outliers.

Resources: ICP **TODO:** *ICP Resources*

15.3 NOISE THAT ISN'T GAUSSIAN: ROBUSTNESS AND IRLS

In our examples, if we assume the noise is normal and isotropic, the squared error is reasonably described as negative log-likelihood. But in some cases, even when the measurements and the reference points are properly aligned, some measurement points may lie quite far from the closest reference point. One reason is pure error. Effects like scattering from rain or translucency can cause LIDAR or depth sensors to report measurements that are quite different from the actual geometry. Another is overhangs, which occur when either the reference or measured set contains points representing geometry that isn't in the other set. In this case, some points from one set should be far away from the closest point in the other set. Each of these effects (Figure 32.2) means that modelling noise as Gaussian may not be justified.

Large distances between some point pairs could have a significant effect on the estimate of the transformation. The square of a large number is very large indeed, so that reducing a large distance somewhat can justify incurring small to medium error on many other pairs (Figure **??**). A simple procedure to manage this effect is to ignore corresponding pairs if the distance between them is too large. One estimates the transformation using only pairs where distances are small. If

points were omitted in one step of the iteration, they may return in another. This strategy can be helpful, but there is a danger that too many pairs are omitted and the iteration does not converge. Corresponding pairs with large distances between them are likely *outliers* – measurements or data that will not conform to a model, but can have significant impact on estimating the model. Well established procedures for handling outliers are easily adapted to registration problems.

15.3.1 IRLS: Weighting Down Outliers

Rather than just ignoring big distances, one might weight down correspondences that seem implausible. Doing so requires some way to estimate an appropriate set of weights. A large weight for errors at points that are "trustworthy" and a low weight for errors at "suspicious" points should result in a registration that is robust to outliers. We can obtain such weights using a *robust loss*, which will reduce the cost of large errors. This can be seen as modifying the probability model. Gaussian noise tends to produce few large values (which so have very large negative log-likelihood), and we want a model that has higher probability of large errors (equivalently, penalizes them less severely than a normal model would). Write θ for the parameters of the transformation, $\mathcal{T}_\theta$ for the transformation, and $r_i(\mathbf{x}_i, \mathbf{y}_{c(i)}, \theta)$ for the residual error of the model on the ith measurement and its corresponding reference point. For us, r_i will always be $l2norm\mathbf{x}_i - \mathcal{T}_\theta(\mathbf{y}_{c(i)})$. So rather than minimizing

$$\sum_i (r_i(\mathbf{x}_i, \mathbf{y}_{c(i)}, \theta))^2$$

as a function of θ, we will minimize an expression of the form

$$\sum_i \rho(r_i(\mathbf{x}_i, \mathbf{y}_{c(i)}, \theta); \sigma),$$

for some appropriately chosen function ρ. Clearly, our negative log-likelihood is one such estimator (use $\rho(u;\sigma) = u^2$). The trick is to make $\rho(u;\sigma)$ look like u^2 for smaller values of u, but ensure that it grows more slowly than u^2 for larger values of u.

The *Huber loss* uses

$$\rho(u;\sigma) = \begin{cases} \frac{u^2}{2} & |u| < \sigma \\ \sigma|u| - \frac{\sigma^2}{2} & \end{cases}$$

which is the same as u^2 for $-\sigma \leq u \leq \sigma$, switches to $|u|$ for larger (or smaller) σ, and has continuous derivative at the switch. The Huber loss is convex (meaning that there will be a unique minimum for our models) and differentiable, but is not smooth. The choice of the parameter σ (which is known as *scale*) has an effect on the estimate. You should interpret this parameter as the distance that a point can lie from the fitted function while still being seen as an *inlier* (anything that isn't even partially an outlier).

The *Pseudo Huber loss* uses

$$\rho(u;\sigma) = \sigma^2 \left(\sqrt{1 + \left(\frac{u}{\sigma}\right)} - 1 \right).$$

FIGURE 15.5:
TODO: Figure showing a bunch of robust loss functions **TODO:** Source, Credit, Permission

A little fiddling with Taylor series reveals this is approximately u^2 for $|u|/\sigma$ small, and linear for $|u|/\sigma$ big. This has the advantage of being differentiable.

The **** **TODO:** what is this loss called uses

$$\rho(u;\sigma) = \frac{\sigma^2 u^2}{u^2 + \sigma^2}$$

which is approximately u^2 for $|u|$ much smaller than σ, and close to σ^2 for $|u|$ much larger than σ.

Each of these losses increases monotonically in $|u|$ (the absolute value is important here!), so it is always better to reduce the residual. For the Huber loss and the Pseudo-Huber loss, the penalty grows with $|u|$, but grows more slowly with big $|u|$ than with small $|u|$. This implies that the underlying probability model will produce very large distances less often than large distances, but more often than a Gaussian model would. For the **** loss, the penalty eventually increases extremely slowly with increasing $|u|$, implying the underlying probability model is willing to produce arbitrarily large distances on occasion, and that the probability of large distances declines very slowly.

Our minimization criterion is

$$\begin{aligned} \nabla_\theta \left(\sum_i \rho(r(\mathbf{x}_i, \mathbf{y}_i, \theta); \sigma) \right) &= \sum_i \left[\frac{\partial \rho}{\partial u} \right] \nabla_\theta r(\mathbf{x}_i, \mathbf{y}_i, \theta) \\ &= 0. \end{aligned}$$

Here the derivative $\frac{\partial \rho}{\partial u}$ is evaluated at $r(\mathbf{x}_i, \mathbf{y}_i, \theta$, so it is a function of θ. Now notice that

$$\begin{aligned} \sum_i \left[\frac{\partial \rho}{\partial u} \right] \nabla_\theta r(\mathbf{x}_i, \mathbf{y}_i, \theta) &= \sum_i \left[\left(\frac{\frac{\partial \rho}{\partial u}}{r(\mathbf{x}_i, \mathbf{y}_i, \theta)} \right) \right] r(\mathbf{x}_i, \mathbf{y}_i, \theta) \nabla_\theta r(\mathbf{x}_i, \mathbf{y}_i, \theta) \\ &= \sum_i \left[\left(\frac{\frac{\partial \rho}{\partial u}}{r(\mathbf{x}_i, \mathbf{y}_i, \theta)} \right) \right] \nabla_\theta \left[r(\mathbf{x}_i, \mathbf{y}_i, \theta) \right]^2 \\ &= 0. \end{aligned}$$

Now $\left[r(\mathbf{x}_i, \mathbf{y}_i, \theta) \right]^2$ is the squared error. If we happened to know the true minimum $\hat{\theta}$ and wrote

$$w_i = \left(\frac{\frac{\partial \rho}{\partial u}}{r(\mathbf{x}_i, \mathbf{y}_i, \theta)} \right)$$

(evaluated at that minimum), then

$$\sum_i w_i \nabla_\theta \left[r(\mathbf{x}_i, \mathbf{y}_i, \theta) \right]^2 = 0$$

at $\theta = \hat{\theta}$. We do not know w_i, but if we did, we already have a recipe to solve this problem for a variety of different transformations (Sections 32.2, 32.2 and 32.2). A natural strategy to adopt is to start with some transformation estimate and unit weights, then repeat:

- **Estimate correspondences** using the estimated transformation. Because all the robust losses are monotonic in $|u|$, finding the closest reference point to each measurement will do.

- **Re-estimate weights** using the new correspondences and the transformation.

- **Re-estimate transformation** using the new correspondences and the new weights, and the closed form algorithms from Sections 32.2, 32.2 and 32.2.

This procedure is known as *iteratively reweighted least squares*

Procedure: 15.4 *Estimating a Transformation from Data with a Robust Loss: Initialization*

Given N known reference points $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,d})$ in affine coordinates and M measurements $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$, initialize by: **TODO:** What is best? likely translation from cogs, affine / euclidean from second moments, but how do you compute second moments robustly?

Procedure: 15.5 *Estimating a Transformation from Data with a Robust Loss: Iteration*

Start with N known reference points $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,d})$ in affine coordinates and M measurements $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,d})$, and a transformation estimate $\mathcal{T}_{\theta^{(1)}}$ with parameters θ^1. Form $\mathbf{m}_i^{(1)} = \mathcal{T}_{\theta^{(1)}}(\mathbf{y}_i)$, then iterate:

- for each $\mathbf{x}_i$, find $\mathbf{m}_{c(i)}^{(n+1)}$ that is closest;
- for each pair $(\mathbf{x}_i, \mathbf{m}_{c(i)}^{(n+1)})$, form $u_i = \| \mathbf{x}_i - \mathbf{m}_{c(i)}^{(n+1)} \|_2$ and

$$w_i = \left(\frac{\frac{\partial \rho}{\partial u}}{\mathbf{u}_i} \right);$$

- estimate $\mathcal{T}_{\theta^{(n+1)}}$ using the set of pairs $(\mathbf{x}_i, \mathbf{m}_{c(i)})$ and the weights w_i;
- form $\mathbf{m}_i = \mathcal{T}_{\theta^{(n+1)}}(\mathbf{m}_i)$.

Test for convergence by testing either that the correspondences did not change in a round, or by checking that $\mathcal{T}_{\theta^{(n+1)}}$ is close to the identity. The required transformation is $\mathcal{T}_{\theta^{(n+1)}} \circ \mathcal{T}_{\theta^{(n)}} \circ \ldots \circ \mathcal{T}_{\theta^{(1)}}$.