Robust Estimation: Inliers and outliers

One problem with least squares techniques is that they (implicitly) assume that the noise distribution is the same for all points. In many situations, though, the situation is more complicated, namely there are some points that obey the model (called *inliers*) and other points that do not (called *outliers*). In the line fitting problem, if we are penalizing all points by the squared distances to a candidate line, the outliers can have a huge penalty (sometimes called *loss*). This can drive the estimated solution away from the correct solution (that is, the correct solution *for the inliers*). The same problem comes up in vanishing point detection, when some edges "point to" a vanishing point, but others do not.

Let's now look at a few other methods that are more robust to outliers and that are popular in computer vision. We'll first look at the Hough transform which was invented in 1962. Then we'll look at a method called RANSAC (Random Sample Consensus) which was first published in 1981 by Fischler and Bolles.

To motivate the Hough transform, we consider again the line fitting problem. For each sample point (x_i, y_i) we are penalizing each line (θ, r) by adding a penalty that is the squared distance from the point to that line. There are other possibilities, though. We could penalize points by their squared distance from the line (or their absolute distance) *up to some distance*, say d_{max}. and then beyond this distance, have a constant error. There are many methods based on this idea and they fall under the name *robust statistics*.

One simple approach is to give zero error up to some margin distance τ and then constant error beyond that. We would then try to find the (θ, r) that minimizes this error. For example:

```
for each line (theta,r)   // need to choose a sampling (quantization)
   Count number of points (xi,yi) that fall outside the distance tau from line
Return the (theta,r) pair with the smallest count
```

or equivalently,

```
for each line (theta,r)   // need to choose a sampling (quantization)
    Count number of points (xi,yi) that fall inside the distance tau from line
Return the (theta,r) pair with the largest count
```

Hough transform

The Hough transform does essentially what the second algorithm above described, except that it swaps the order of the loops, namely the outer loop is over the positions (x_i, y_i) and the inner loop is over lines.

```
define a matrix H that counts the votes for a line defined by (r,theta), and
initialize this matrix to 0
for each (x_i,y_i){
   for each theta{  //  need to choose a sampling of thetas's
       r :=  round(x_i cos(theta) + y_i (sin theta))
       H(r,theta)++   // increment count
   }
}
return the (theta,r) value with the most counts (``votes'')
```


The *Hough Transform* refers to the transformation from a set of points (x_i, y_i) to a histogram of votes in the model parameter space, in this case, (r, θ).

[ASIDE: One minor technical point: for a given (x_i, y_i), the line (r, θ) is identical to the line $(-r, \theta + \pi)$. Thus the votes would always have a certain two-fold symmetry. One would choose the solution with $r > 0$.]

The advantage of Hough is that the outliers have very little effect on the estimate. As long as the outliers don't conspire to vote on some other particular model, the votes of the outliers will be spread out over the (r, θ) space. Of course, if there *are* two distinct lines present, then you will get two peaks and you would need to choose between them, or just take both and conclude there are two good models.

See lecture slides and Matlab code for examples of using the Hough transform to find lines that pass through a given set of points.

In the lecture, I also discussed the problem of estimating vanishing points using the Hough transform. This is a problem you will work on in Assignment 1. See the slides and lecture recording.

RANSAC (Random Sample Consensus)

Let's now consider second approach which is often used in computer vision when there are lots of outliers and when the number of parameters of the model is more than two. We will see models later in the course when the number of parameters is 7 or 8, and we'll want to estimate these parameters precisely. In that case, a Hough transform approach just doesn't work – the dimensionality of the model space is too high.

We start with the problem of fitting a line model to a set of points. We have seen that least squares tries to use all of the data to fit a model. This next approach (RANSAC) is the extreme opposite. It fits a model using the minimal number of points possible. For a line, the minimal number of points is 2.

RANSAC approach to find a line in a set of points

The RANSAC algorithm for line fitting is roughly as follows. Sample a large number of point pairs. For each point pair, fit a line model, namely find the unique line passing through the two points. Check all the other points and see how many of them lie near the line, where "near" means within some distance threshold τ. These points are called the *consensus set* for that line model. Repeat this some number of times. Then, choose the model that has the largest consensus set. Finally, use least squares (either with regression or with total least squares) to fit a model to this consensus set and terminate.

At first glance, this algorithm seems a bit crazy. You might think that, even if you happen to sample only inliers, each of the points has noise and so the line that passes through these points will be very sensitive to this noise. Surely (you think) it would be better to randomly choose 3 points (or more) since they would better fend off the noise and give a better fit.

The problem with this intuition is that the more points you sample to fit a line, the greater the likelihood that one of the sample points will be an outlier, and if one of your points is an outlier then your fit most likely will be terrible. That means that you would have to sample and fit more times in order to find triples (or quadruples) of points that are all inliers.

So here is the RANSAC algorithm for fitting a line: [**UPDATE (Sept. 24): The last line was modified slightly from what was originally posted.**]

```
Repeat many times {
   Randomly sample 2 data points and fit a line to them
   Examine the remaining points N-2 data points & count how many (C) of them are within
   a threshold distance (tau) from the line.  These points are the 'consensus set'.
   If C is sufficiently large (greater than some threshold), then    use the points
   in the consensus set to refit the line e.g. using least squares.   If the error
   to the fit is smaller than the best model so far, then this becomes the best line model.
}
```

Here is a more general description:

```
Repeat many times{
    Randomly sample n data points and fit a model where n is
    the minimum number of points needed for an exact model fit;
    Examine all remaining points (N-n) and count how many C are within some threshold
    distance tau  from the model. (This is the 'consensus set' for the model.)
    If C is sufficiently large (greater than some threshold), then    use the points
    in the consensus set to refit the model e.g. using least squares.   If the error
    to the fit is smaller than the best model so far, then this becomes the best model.
}
```

[UPDATE (Sept. 24): Note that you can iterate the following: find consensus set, fit model, find consensus set, fit model, etc. This iteration would potentially give you a larger consensus set and a better fit.]

RANSAC has a number of parameters that need to be chosen. For example, we need to decide when a point (or edge, in the case of the vanishing point problem) is close enough to a model that we consider it part of the consensus set. We also need to decide when to stop looking for a better model. These parameters or stopping criteria are related, e.g. by loosening the criterion for accepting a point into the consensus set, we will increase the number of points in the consensus set which could lead us to terminate the loop earlier, namely if we terminate when the consensus set becomes sufficiently large.

RANSAC and Probability

As mentioned earlier, one counterintuitive idea with RANSAC is that if we only choose the minimal number of samples to fit a model, then it seems unlikely that we will fit a good model. In particular, it seems unlikely that we will pick all inliers. So let's explore the probabilities a bit.

Let w be the probability that if that a randomly sampled point is an inlier, so $0 < w < 1$. Suppose we need n points to fit a model. In the case of a line $n = 2$. Drawing n points is called a *trial*. The probability of all n points in a trial being inliers is w^n, and the probability that we fail to find all inliers in a trial is $1 - w^n$, that is, this is the probability that at least one of the n points in a trial is an outlier.

If we run k trials, then the probability that we fail to find all inliers in *all* k trials is $(1 - w^n)^k$. So if you somehow have an idea of what w is and if you have some desired probability z of getting at least one good model, then you can estimate how many trials k you need by solving $z = 1-(1-w^n)^k$ for k.

