Using Newton’s method, we can find the position implied by these data:

\[
\begin{align*}
\gg \text{newtonMethod}('gpsfun', [0,0,0,0], \text{sats})
\text{ans: } 1.0e-014 * \\
&0.1508 \\
&-0.1529 \\
&-0.1593 \\
&0.1548
\end{align*}
\]

The positions are practically zero, which is the right answer.

Linearize the GPS equations around the solution point when the satellites are randomly placed in three-dimensional space, and find the condition number of the system. Repeat this when the satellites are placed randomly in a plane or along a line.

17.11 Project: Alignment of Images

A tourist stands on a hilltop, surveying the panorama around her. Turning clockwise around, she snaps a series of photos, capturing the horizon in a full circle. She makes sure that the left side of each photo will overlap the right edge of the previous one. Once at home, she will arrange the photos in sequence, using the overlap to align them.

But the developed photos bring a disappointment. No matter how the photographer shifts or rotates the photos, the overlapped areas do not align.

The problem is one of perspective. The adjacent photos were taken with the camera pointing in different directions. Each photo is a projection of a three-dimensional scene onto the two-dimensional firm; in such a projection, lines that are parallel in 3D become, in 2D, convergent to a point on the horizon. The projection distorts shapes and different projections distort shapes differently. Two differently distorted pictures cannot be aligned by shifts of position or rotation.

To accomplish the alignment, we need a more general method. Consider a pair of images. We call one the target and the other the source. Our goal is to pull pixels from the source and place them in the correct position in the target. Figure 17.19 shows a source and target image plotted with coordinate scales. The source coordinates are labeled \((x,y)\) and the target coordinates \((u,v)\).

We want to extend the target image to the left. For instance, we would like to fill in the pixel at position \((u = -20, v = 400)\). To find a color for this pixel, we use information in the source picture. We need to find where in the \((x,y)\) source coordinates the point \((u = -20, v = 400)\) is located. To do this, we find a pair of functions

\[
\begin{align*}
x &= f(u,v) \\
y &= g(u,v)
\end{align*}
\]
17.11 Project: Alignment of Images

Figure 17.19. Two photos from a panoramic sequence. Some easily identified corresponding pairs of points in the two images have been marked.

These two functions, once we find them, allow us to take any \((u,v)\) coordinate and find the corresponding \((x,y)\) coordinate.

Given the functions in Eq. (17.25), the image-merging algorithm will work like this:

0 Take as arguments a source image \(s\), a target image \(t\), and the coordinate transformations \(f(u,v)\) and \(g(u,v)\).
1 Pick a set of indices in \((u,v)\) coordinates that we wish to fill in. Call these \((u_{\text{new}}, v_{\text{new}})\).
2 Transform \((u_{\text{new}}, v_{\text{new}})\) into \((x,y)\) coordinates:

\[
\begin{align*}
x_{\text{new}} &= f(u_{\text{new}}, v_{\text{new}}) \\
y_{\text{new}} &= g(u_{\text{new}}, v_{\text{new}})
\end{align*}
\]

3 Read off the pixels \(s(x_{\text{new}}, y_{\text{new}})\) and fill in the target image \(t\) with these values. Since in general \(x_{\text{new}}\) and \(y_{\text{new}}\) will not be integers, interpolation may be necessary.

The question we face is how to find the functions \(f(u,v)\) and \(g(u,v)\). One way to do this is by fitting. In Figure 17.19, several pairs of corresponding points, one in each image, have been marked. We can use the coordinates of these points as data to fit the transformation functions. By fitting, of course, we mean adjusting parameters in a predefined form of function. What should that form be?

To mimic the manual alignment process, involving rotating and shifting the photo without changing its shape as in Figure 17.20, we can use the functional form

\[
\begin{align*}
x &= \cos(\theta)u - \sin(\theta)v + c \\
y &= \sin(\theta)u + \cos(\theta)v + d
\end{align*}
\]

(17.26)
Figure 17.20. The alignment of the panoramic photographs using pure rotation and translation of the left image, as would be done by hand. The rotation is by a small angle in the clockwise direction. In the region of overlap between the two photographs, the target photograph has been drawn to be slightly transparent, so that the discrepancy between the two photographs can be seen. For example, note that the car in the right part of the parking lot is located in different positions in the two photographs.

$\theta$, $c$, and $d$ are the parameters that are adjusted to accomplish the alignment.

The transformation functions in Eq. (17.26) depend linearly on the variables $u$ and $v$. This means that a straight line in $(u,v)$ coordinates will remain a straight line after transformation to $(x,y)$ coordinates, exactly what we would expect by a transformation that involves only rotation and shifting. The functions are, however, nonlinear in the parameter $\theta$.

A slightly more general system, allowing for magnifying or reducing the photo before rotating it, is linear both in the variables $u,v$ and in the parameters, which are $a$, $b$, $c$, and $d$.

$$
\begin{align*}
  x &= au - bv + c \\
  y &= bu + av + d
\end{align*}
$$

Equations (17.27) and (17.26) are “models” of the true transformation that would have mapped the real-world points into the target image. Like almost all models, they are certainly not true; nevertheless they may be useful for accomplishing the task at hand.

Whatever the true transformation is, we know that it can be approximated by a linear form, at least over small distances. The full linear form has six individual parameters $a$, $b$, $c$, $d$, $e$, and $f$.

$$
\begin{align*}
  x &= au + bv + c \\
  y &= du + ev + f
\end{align*}
$$
The full linear transformation, as well as Eqs. (17.27) and (17.26), which are special cases of linear transforms, has the nice property that straight lines stay straight under transformation. Thus, the horizon remains a straight line. However, under these transformations, parallel lines remain parallel, which isn’t consistent with the sorts of distortions introduced by perspective. A substantially better approximation—but still an approximation—is given by

\[
\begin{align*}
x &= a_1 u + b_1 v + c_1 uv + d_1 \\
y &= a_2 u + b_2 v + c_2 uv + d_2
\end{align*}
\]  

(17.29)

To fit this model, take the \( spts \) data, which we write as \( x_i, y_i \) for \( i = 1, \ldots, p \), where there are \( p \) data points. Similarly, the \( tpts \) data is \( u_i, v_i \) for \( i = 1, \ldots, p \). Since this is a model that is linear in the parameter, the least-squares fit can be found by solving an equation \( A \cdot x = b \). Let the unknown parameter column vector be the eight parameters \( x = (a_1, b_1, c_1, d_1, a_2, b_2, c_2, d_2) \). The column vector \( b \) is \( b = x_1, x_2, \ldots, x_p, y_1, y_2, \ldots, y_p \). The matrix \( A \) is

\[
A = \begin{pmatrix}
    u_1 & v_1 & u_1 v_1 & 1 & 0 & 0 & 0 & 0 \\
    u_2 & v_2 & u_2 v_2 & 1 & 0 & 0 & 0 & 0 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    u_p & v_p & u_p v_p & 1 & 0 & 0 & 0 & 0 \\
    0 & 0 & 0 & 0 & u_1 & v_1 & u_1 v_1 & 1 \\
    0 & 0 & 0 & 0 & u_2 & v_2 & u_2 v_2 & 1 \\
    \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
    0 & 0 & 0 & 0 & u_1 & v_p & u_p v_p & 1
\end{pmatrix}
\]  

(17.30)

The solution \( x \) can be found in the familiar way. Similar systems can be set up for the other linear models.4

Figure 17.21 shows the alignment using the nonlinear transformation of Eq. (17.29) as fitted to the points marked in Figure 17.19. The nonlinear transformation has curved the edges of the source photograph. The agreement between the source and target photographs is very good for areas of the image near the \( spts \) and \( tpts \) data and much less good far away. In particular, note how the far-left side of the image appears to be distorted compared to the original source image.

All of the models, Eqs. (17.27), (17.28), or (17.29), can be fitted by solving sets of linear equations. To illustrate, we fit Eq. (17.29). The data are the pairs of corresponding points \( (u_i, v_i) \) and \( (x_i, y_i) \) shown in Figure 17.19.

In this project, you are to build a system for aligning photographs using the four methods described previously. To start, you will need the two photographs to be aligned. (The photographs used in the preceding examples are \( \text{left.png} \) and \( \text{right.png} \).)

---

4Since the matrix \( A \) consists of two copies of a \( 4 \times 4 \) matrix, it’s possible to find \( a_1, b_1, c_1, d_1 \) independently of \( a_2, b_2, c_2, d_2 \) by solving two \( 4 \times 4 \) systems. But this isn’t true for the system in Eq. (17.27).
Figure 17.21. Alignment of the two images using the nonlinear model of Eq. (17.29). Compared with the rotation/shift model shown in Figure 17.20, the nonlinear model shows much better alignment within the region of overlap, but there is a distortion at the extreme lower-left region of the image.

The program identPts will display the source and target images so that corresponding points can be identified. You can use this program to create two sets of points, one for the target and one for the source image, as indicated in Figure 17.19. The syntax is

\[
\gg [\text{spts, tpts}] = \text{identPts(source, target)};
\]

Exercise 17.7:
Write a program fitLinearModel to find the parameters of Eq. (17.28) from the data in spts and tpts.

Exercise 17.8:
Write a program doLinearInterp that takes several arguments:

\begin{itemize}
  \item \text{unew} and \text{vnew} Lists of new points in the target image coordinate system that you want to find the pixel values of.
  \item \text{s} The source image.
  \item \text{params} The parameters of the transformation.
\end{itemize}

The program should transform \text{unew} and \text{vnew} to the \((x,y)\) coordinate system, find the pixel values at those coordinates using interpolation, and return the set \text{unew} and \text{vnew} along with the pixel values \text{svals}.