Map Stitching Basic Method

Diganta Bhattacharya

February 26, 2022

Contents

1 Making a vector map stitching satellite screenshots

Consider a region small enough so that earth's curvature may be ignored, yet large enough so that Google map cannot fit it in a single screen at high resolution. We cover the region with a number of overlapping screenshots all of the same resolution. All the screenshots are numbered and are in the shape of rectangles. The red dots are known locations that you can identify at the current resolution of the screenshot. For each screenshot find the approx pixel coordinate of the location (e.g., by clicking on the centre of each red disk, and reading the mouse coordinates). For example, see the picture below.

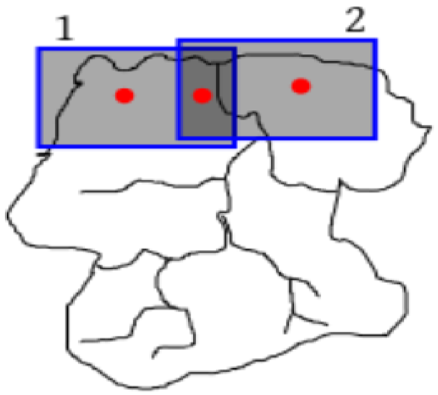

Figure 1: Example screenshots

We measure (r_{1j}, c_{1j}) from screenshot 1.

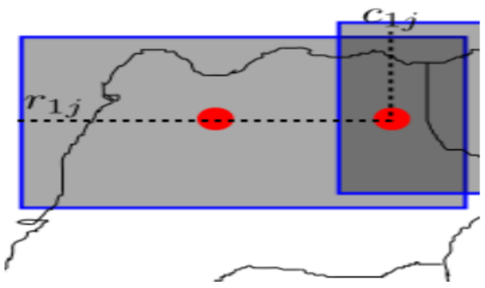

Figure 2: First screenshot

Here r stands for "row", and c for "column". The first subscript is the screenshot number, the second is the location number (we assume that the central red dot in the j-th in the list of the known locations). Similarly, from screenshot

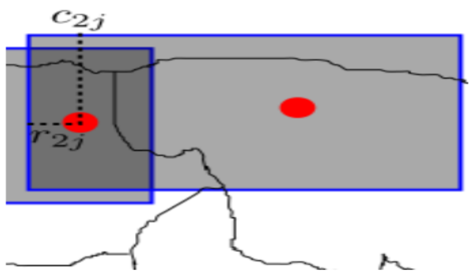

Figure 3: Caption

2 we measure (r_{2j}, c_{2j}). Notice that we are using the same j, as it is the same location. Notice that we are using the same j, as it is the same location.

1.1 Aim

Our aim is to write a program in R which does the following-

- Collects the data points from the user for each screenshot.
- States if there appears to be a mistake in data input for the locations.
- Gives a plot describing the positions of all the marked locations relative to each other in case it is possible to decide that from the given information.
- States that it is not possible to give the plot from the given data otherwise.

1.2 Form of the data collected

Thus our data set consists of a subset of (r_{ij}, c_{ij})'s for $i = 1, \ldots$,number of screenshots, and $j = 1, \ldots$,number of locations which is the coordinate of the j-th location in the i-th screenshot in pixels. Of course, not all (i, j) pairs occur in the data, since the j-th location may not show up in the i-th screenshot.

2 Taking input from user

To take input from the user, we used the *locator*() function available in R. This function allows the user to click on a uploaded image or plot a desired number of times and then returns the value of the x and y coordinates of the clicked on points with respect to the origin of the plot or the image. We uploaded the screenshots one by one in the R code and asked the user to mark the location points from the sreenshots while naming each of them as distinct integers from 1 to number of locations. It is necessary that the same location is given the same name in every screenshot.

3 Model of the problem

There are two parts of the problem for which we will consider two separate models. the first part is the problem of bringing all the screenshots to the same scale used for distances and the second part is joining the screenshots in the same scale.

3.1 Scaling the screenshots

As it is sufficient to just describe the relative positions in the final plot, we will scale all the screenshots to any fixed global scale. Let there be N screenshots and P locations in total. Thus, there are $\binom{p}{2}$ distances between the locations in the global map. Let the true lengths of these $\binom{p}{2}$ distances be $\alpha_1, \alpha_2, \ldots, \alpha_L$, where $L = \binom{p}{2}$. Let the l-th lines length be calculated as D_{il} in the i-th screenshot. Of corse all D_{il}s are not available. then,

$$D_{il} \approx \beta_i \cdot \alpha_l \tag{1}$$

The approximation being due to possible error in marking by the user. But as above is not a linear model, for our convenience we take logarithm on both sides to get-

$$log\ D_{il} \approx log\ \beta_i + log\ \alpha_l \tag{2}$$

In above, we assume the errors to be independently identically and normally distributed with mean zero to get

$$log\ D_{il} = log\ \beta_i + log\ \alpha_l + \epsilon_{il} \tag{3}$$

We assume for our model that

$$\vec{\epsilon} \sim N(0, \sigma^2) \tag{4}$$

for some unknown positive σ^2. This completes the task of presenting a model for the scaling parameters for each screenshot.

3.2 Stitching the scaled screenshots

For this part of our model we will assume that all the screenshots have already been brought to the same scale. We will now try to position all the screenshots with respect to some fixed global coordinates. Let the true coordinates of the j-th location be (x_j, y_j), $j = 1, \ldots, P$. Let the i-th screenshot's origin , O_i be placed at global coordinates (u_i, v_i), for $i = 1, \ldots, N$. Then,

$$r_{ij} \approx x_j - u_i \tag{5}$$

$$c_{ij} \approx y_j - v_i \tag{6}$$

where the approximation is due to error made by the user while marking the locations. So, the above can be made into a simple linear model my taking

$$r_{ij} = x_j - u_i + \epsilon_{ij} \tag{7}$$

and

$$c_{ij} = y_j - v_i + \epsilon'_{ij} \tag{8}$$

We assume errors along the X-axis and the Y-axis to be independent. Thus we have two separate linear models with

$$\vec{\epsilon} \sim N(0, \sigma'^2) \tag{9}$$

and

$$\vec{\epsilon'} \sim N(0, \sigma''^2) \tag{10}$$

This completes our final linear models.

4 Solving the linear models

All that is left to complete our program is to find a method to solve our linear models described in the previous section. For this purpose we use the $lm()$ function available in R which gives us details about the fit of a given linear model and data.

- In case the data is sufficient, we will have the least square estimates for each of the parameters.
- From the first linear model, we receive the least square estimates of logarithms of the scaling parameters which we use to scale the data in a fixed global scale.
- The scaling parameter of the first screenshot is always dropped as this is taken as the global scale.
- In case the least square estimate for any of the other screenshots are not available, the program states that it is not possible to give any result.
- In case we were successful in scaling the screenshots, we fit the next two linear models described for stitching the map.
- If the estimates of global coordinates are available for each screenshot, we plot them to get the final result.
- If the estimate of global coordinate of any location is not available from the $lm()$ function, the program states that it is not possible to give any result.
- If the fitted and the true value of any global coordinate differ largely, the program states that there was a possible error in entering the data.

5 The final program

Following the above method, the final code is given as below-

```r
merge<-function()
    {
     library("OpenImageR")
     P= readline(prompt="Enter the total numer of points in the map.")
     P=as.integer(P)
     S = readline(prompt="Enter the total number of screenshots.")
     S = as.integer(S)
     data=NULL
     for(i in 1:S)
        {
         img=readline("Enter the name of the screenshot.")
         img=as.raster(readImage(img))
         plot(img)
         k=readline(prompt="Enter the number of points to be marked.")
         k=as.integer(k)
         for(j in 1:k)
            {
             print("Mark the next point.")
             coord=locator(1)
             name=readline(prompt= "Name this point.")
             name=as.integer(name)
             data=rbind(data,c(coord$x,coord$y,name,i))
            }
        }
     class(data)="numeric"
     dist=NULL
     for(i in 1:S)
        {
         loc=which(data[,4]==i)
         for(j in loc)
            {
             for(k in loc)
                {
                 if(data[j,3]<data[k,3])
                    {
                     temp=((data[j,1]-data[k,1])^2+(data[j,2]-data[k,2])^2)^(1/2)
                     temp=log(temp)
                     name=paste(data[j,3],data[k,3],sep=",")
                     dist=rbind(dist,c(temp,name,i))
                    }
                }
            }
        }
     a=dist[,1]
     class(a)="numeric"
     b=factor(dist[,2])
     c=factor(dist[,3])
     resize=lm(a~b+c-1)
     name=paste("c",1:S,sep="")
     scale=coef(resize)[name]
     if(length(which(is.na(scale)==TRUE))>1)
        {
         print("The scales cannot be determined for certain
             pictures. Please try again.")
        }
     scale=exp(scale)
#########################

###########################
     n=nrow(data)
     for(i in 1:n)
        {
         if(data[,4]!=1)
            {
             pic=data[i,4]
             data[i,1]=data[i,1]/scale[pic]
            }
        }
     points=factor(data[,3])
     screenshots=factor(data[,4])
     fit.x=lm(data[,1]~points+screenshots-1)
     name=paste("points",1:P,sep="")
     name2=paste("screenshots",1:P,sep="")
     xcoord=coef(fit.x)[name]
     fit.y=lm(data[,2]~points+screenshots-1)
     ycoord=coef(fit.y)[name]
     plot(xcoord,ycoord)
     t=levels(points)
     text(xcoord,ycoord-5,t)
     print("The max error in both coordinates are--")
     print(c(max(abs(fit.x$residuals)),max(abs(fit.x$residuals)
         )))
     result=cbind(xcoord,ycoord)
     return(as.matrix(result))
    }
```

6 Outcome of the program

We used screenshots of the following picture to test our program. The final location points were plotted as below- So our program works sufficiently.

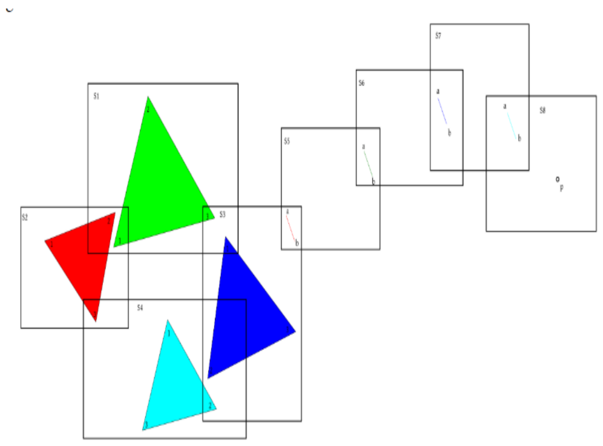

Figure 4: Test picture

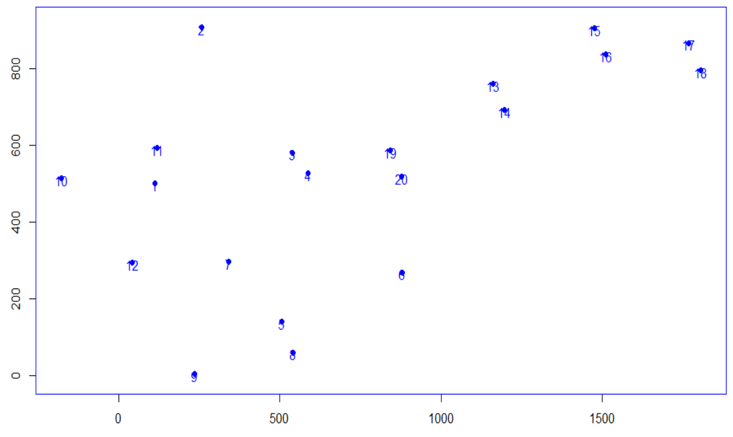

Figure 5: Outcome picture