Optical Flow Analysis using R

This tutorial demonstrates how to track movement in gridded regions over successive frames using the R package imagefx. Applications for this type of analysis may include tracking granular flow experiments, clouds in satellite images, traffic patterns, or general movement in video frames.
The tutorial is broken into the following sections:

1. Read in image data
2. Grid Image
3. Track Motion in Grids
4. Plot Grid Motion

We assume the user has R installed on their computer and has a basic familiarity with R syntax, base functions, and coding in general. To install the `imagefx` package from the R console use `install.packages('imagefx')` and then load with:

```
library(imagefx)
```

Read in image data

Data used in this tutorial can be found on github. Navigate to the Optical Flow repository and then:

1. Click *Clone or download*
2. Click *Download Zip*
3. Use your computers software to extract the repository contents in a location you can easily navigate to
4. Open up RSTUDIO or an R terminal and navigate to the directory you specified above.

For this tutorial I saved the optical_flow-Master repository on my local machine but the reader should modify any code to reflect the location where they unpacked the zip file or to the location of the data of interest. All the data we analyze are jpegs and thus require the 'jpeg' package, which can be installed using

```
install.packages('jpeg')
```

Now we can use the *imagefx* package and *jpeg* packages to read in some example image data.

```
library(imagefx)
library(jpeg)

## identify where the image files are located
##img.dir='~/r_vignettes/optical_flow/optical_flow-master/example_datasets/fluidized_flow/'
##img.dir='~/r_vignettes/optical_flow/optical_flow-master/example_datasets/granular_flow/'
img.dir='~/r_vignettes/optical_flow/optical_flow-master/example_datasets/traffic_flow/'

## list all the files within this directory
img.files <- list.files(img.dir)

## choose the first file in the image files
cur.file = img.files[1]

## read in the JPEG image
cur.img <- readJPEG(paste(img.dir,cur.file,sep=""))
```

Grid the Image

Optical flow algorithms generally break an image into gridded regions whose motions will be tracked between successive frames.

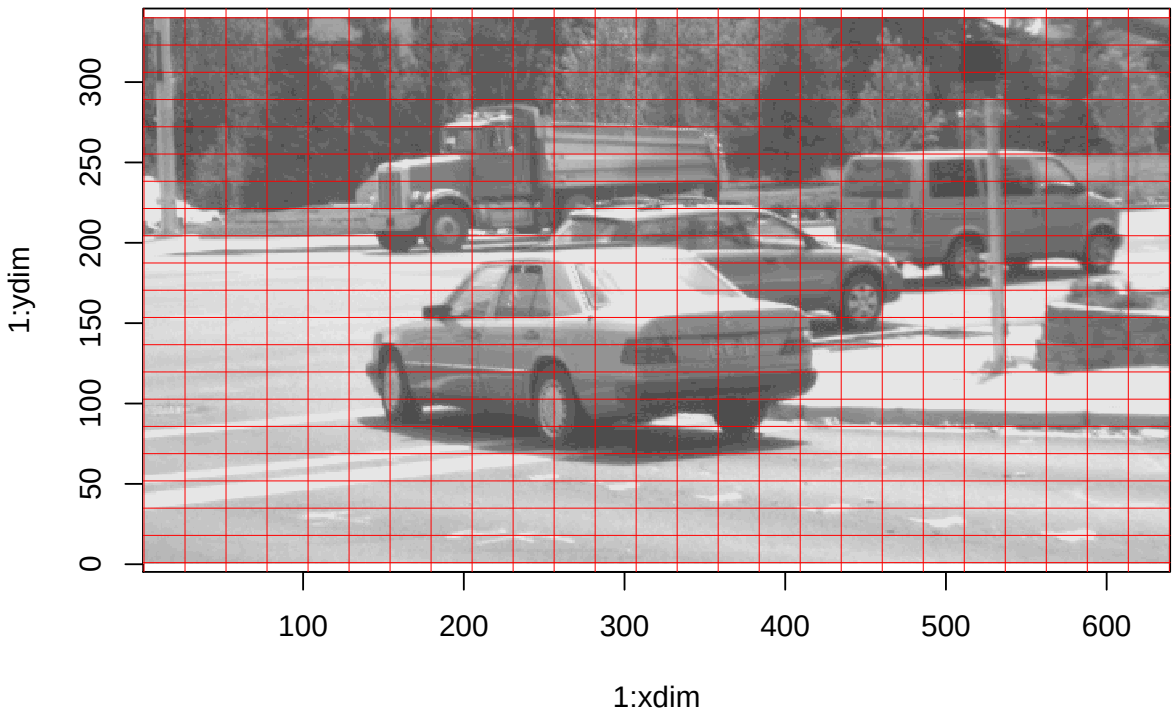
```
## at this point is important to rotate the image 90 degrees clockwise
## this is necessary bc R reads images with the top left corner as the reference frame.
## in other words, rotate the image so the origin is located in the bottom right
img.rot <- t(apply(cur.img,2,rev))

## define the domain of this image
xdim <- nrow(img.rot)
ydim <- ncol(img.rot)

## choose the number of grids in the x and y direction
num.xgrid = 25
num.ygrid = 20

## find the grid xs and ys
grid.xs <- seq(1,xdim,length.out=num.xgrid+1)
grid.ys <- seq(1,ydim,length.out=num.ygrid+1)

## plot the grid locations
image(1:xdim,1:ydim,img.rot,col=gray.colors(20),asp=1)
abline(v=grid.xs,col='red',lwd=0.5)
abline(h=grid.ys,col='red',lwd=0.5)
```



Track Motion in Grids

Now read in successive images, grid each image, and track the relative movement between frames. This is done with two nested loops that loop over each gridded region in the x and y direction. The relative movement between gridded regions in successive frames is accomplished by either `xocrr3d` or `pcorr3d`, which perform cross correlations and phase correlations, respectively. The choice in the algorithm will depend on the dataset used but, in general, phase correlations work better with high frequency images.

```
## initilize a list to hold all the shifts in each grid over every image sequence pair
shift.list <- list()

## loop over all the images
ii=1
while(ii<length(img.files)) {

    ## Read in the current 2 images
    img1.org <- readJPEG(paste(img.dir,img.files[ii],sep=''))
    img2.org <- readJPEG(paste(img.dir,img.files[ii+1],sep=''))

    ## rotate them 90 degrees clockwise
    img1 <- t(apply(img1.org,2,rev))
    img2 <- t(apply(img2.org,2,rev))

    ## what are the image dimensions
    xdim <- nrow(img1)
    ydim <- ncol(img1)

    ### discritize the grid.
    num.xgrid = 25
    num.ygrid = 20

    ## find the grid xs and ys
    grid.xs <- seq(1,xdim,length.out=num.xgrid+1)
    grid.ys <- seq(1,ydim,length.out=num.ygrid+1)

    ## create a matrix holding the x and y shifts and the correlation value for reach grid cell
    ## columns are x grid location, y grid location, x shift, y shift, correlation value
    shift.mat = matrix(NA,nrow=num.xgrid*num.ygrid,ncol=5)

    ## loop over grid xs and ys to find the movement between frames in each grid cell
    k=1
    i=1
    while(i<=num.ygrid) {

        j=1
        while(j<=num.xgrid) {

            ## define the x and y indices for the current gridded region
            x.inds <- grid.xs[j]:grid.xs[j+1]
            y.inds <- grid.ys[i]:grid.ys[i+1]

            ## find the cur grid for each image
            cur.grid1 <- img1[x.inds,y.inds] - mean(img1[x.inds,y.inds])
            cur.grid2 <- img2[x.inds,y.inds] - mean(img2[x.inds,y.inds])
```

```
            ## sometimes it is helpful to window in on the gridded regions
            wind.gaus = build.gaus(nrow(cur.grid1),ncol(cur.grid1),sig.x=10)
            cur.grid1 = cur.grid1*wind.gaus
            cur.grid2 = cur.grid2*wind.gaus

            ## track the movement between grids using the CROSS correlation function
            cur.corr <- xcorr3d(cur.grid1,cur.grid2)

            ## OR track the movement using the PHASE correlation function
            ##cur.corr <- pcorr3d(cur.grid1,cur.grid2)

            ## save the current current correlation components to the shift matrix
            shift.mat[k,] <- c(mean(x.inds),mean(y.inds),cur.corr$max.shifts,cur.corr$max.corr)

            k=k+1
            j=j+1
        }

        i=i+1
    }

    ## save the shift matrix in the shift list
    shift.list[[ii]] <- shift.mat

    ii=ii+1
}
```

Plot Grid Motion

The object `shift.list` holds all the shift vectors (x,y) for every grid location and for every image sequence pair. With this list you can visualize the 'flow' of your data by plotting the vectors as arrows. This can be done for every frame or over the entire sequence by taking the average, which we do below:

```
## convert each matix in the shift list into a data frame
shift.list <- lapply(shift.list,as.data.frame)

## create a matrix of all the shift values in each list element
all.shift.xs <- matrix(unlist(lapply(shift.list, "[", 3)),nrow=num.xgrid*num.ygrid)
all.shift.ys <- matrix(unlist(lapply(shift.list, "[", 4)),nrow=num.xgrid*num.ygrid)

## take the row averages for all the shifts in the x and y direction
avg.shift.xs <- rowMeans(all.shift.xs)
avg.shift.ys <- rowMeans(all.shift.ys)

## take the x and y grid locations from an example shift element
avg.xs = shift.list[[1]][,1]
avg.ys = shift.list[[1]][,2]

## find the shifts that are greater than some minimum tolerance
## this is to avoid warnings when plotting arrows later on
pos.shift.inds <- which(abs(avg.shift.xs) > 0.1 | abs(avg.shift.ys) > 0.1)
```

```
## limit the average shift xs, ys, and x and y locations to the positive indices
pos.shift.xs <- avg.shift.xs[pos.shift.inds]
pos.shift.ys <- avg.shift.ys[pos.shift.inds]
pos.xs <- avg.xs[pos.shift.inds]
pos.ys <- avg.ys[pos.shift.inds]

## what are the useres margins
mar.org=par()$mar

## set the margins for this particular plot
par(mar=c(0,0,0,0))

## plot an example image
image(1:xdim,1:ydim,img1,col=gray.colors(20),useRaster=TRUE,asp=1,axes=FALSE,xlab='',ylab='')

## return the margins to the users original values
par(mar=mar.org)

## define how to scale the arrows (for easier visualization)
arrow.scale = 5

## define the arrow locations
x0=pos.xs
y0=pos.ys
x1=x0+(pos.shift.xs * arrow.scale)
y1=y0+(pos.shift.ys * arrow.scale)

## plot the motion indicating the average shift between frames
arrows(x0=x0,y0=y0,x1=x1,y1=y1,length=0.05,lwd=2,angle=25,col='gray30')
arrows(x0=x0,y0=y0,x1=x1,y1=y1,length=0.05,lwd=1,angle=25,col='red')
```