Madagascar tutorial

Maurice the Aye-Aye

ABSTRACT

In this tutorial, you will go through different steps required for writing a research paper with reproducible examples. In particular, you will

1. identify a research problem,
2. suggest a solution,
3. test your solution using a synthetic example,
4. apply your solution to field data,
5. write a report about your work.

PREREQUISITES

Completing this tutorial requires

- MADAGASCAR software environment available from `http://www.ahay.org`
- LaTeX environment with `SEGTeX` available from `http://www.ahay.org/wiki/SEGTeX`

To do the assignment on your personal computer, you need to install the required environments. An Internet connection is required for access to the data repository.

The tutorial itself is available from the MADAGASCAR repository by running

```
svn co https://rsf.svn.sourceforge.net/svnroot/rsf/trunk/book/rsf/school2009
```

INTRODUCTION

In this tutorial, you will be asked to run commands from the Unix shell (identified by `bash$`) and to edit files in a text editor. Different editors are available in a typical Unix environment (`vi`, `emacs`, `nedit`, etc.)

Your first assignment:

1. Open a Unix shell.
2. Change directory to the tutorial directory

   ```
   bash$ cd $RSFSRC/book/rsf/school2009
   ```

3. Open the `paper.tex` file in your favorite editor, for example by running

   ```
   bash$ nedit paper.tex &
   ```

4. Look at the first line in the file and change the author name from Maurice the Aye-Aye to your name (first things first).

PROBLEM

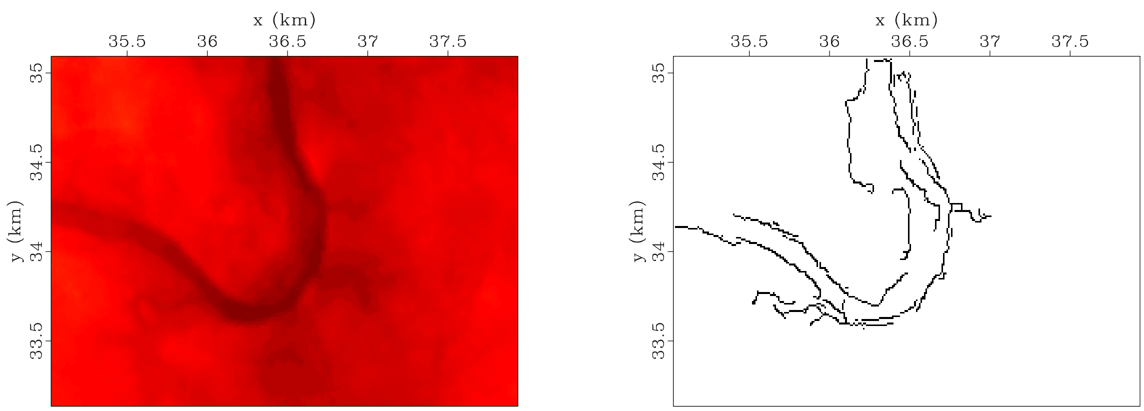


Figure 1: Depth slice from 3-D seismic (left) and output of edge detection (right).

The left plot in Figure 1 shows a depth slice from a 3-D seismic volume[1]. You notice a channel structure and decide to extract it using and edge detection algorithm from the image processing literature (Canny, 1986). In a nutshell, Canny's edge detector picks areas of high gradient that seem to be aligned along an edge. The extracted edges are shown in the right plot of Figure 1. The initial result is not too clear, because it is affected by random fluctuations in seismic amplitudes. The goal of your research project is to achieve a better result in automatic channel extraction.

1. Change directory to the project directory

   ```
   bash$ cd channel
   ```

2. Run

[1]Courtesy of Matt Hall (ConocoPhillips Canada Ltd.)

```
bash$ scons horizon.view
```

in the Unix shell. A number of commands will appear in the shell followed by Figure 1 appearing on your screen.

3. To understand the commands, examine the script that generated them by opening the `SConstruct` file in a text editor. Notice that, instead of Shell commands, the script contains rules.

   - The first rule, `Fetch`, allows the script to download the input data file `horizon.asc` from the data server.
   - Other rules have the form `Flow(target,source,command)` for generating data files or `Plot` and `Result` for generating picture files.
   - `Fetch`, `Flow`, `Plot`, and `Result` are defined in MADAGASCAR's `rsf.proj` package, which extends the functionality of SCons (Fomel and Hennenfent, 2007).

4. To better understand how rules translate into commands, run

   ```
   bash$ scons -c horizon.rsf
   ```

   The `-c` flag tells scons to remove the `horizon.rsf` file and all its dependencies.

5. Next, run

   ```
   bash$ scons -n horizon.rsf
   ```

   The `-n` flag tells scons not to run the command but simply to display it on the screen. Identify the lines in the `SConstruct` file that generate the output you see on the screen.

6. Run

   ```
   bash$ scons horizon.rsf
   ```

   Examine the file `horizon.rsf` both by opening it in a text editor and by running

   ```
   bash$ sfin horizon.rsf
   ```

   How many different MADAGASCAR modules were used to create this file? What are the file dimensions? Where is the actual data stored?

7. Run

   ```
   bash$ scons smoothed.rsf
   ```

Notice that the `horizon.rsf` file is not being rebuilt.

8. What does the `sfsmooth` module do? Find it out by running

   ```
   bash$ sfsmooth
   ```

   without arguments. Has `sfsmooth` been used in any other MADAGASCAR examples?

9. What other MADAGASCAR modules perform smoothing? To find out, run

   ```
   bash$ sfdoc -k smooth
   ```

10. Notice that Figure 1 does not make a very good use of the color scale. To improve the scale, find the mean value of the data by running

    ```
    bash$ sfattr < horizon.rsf
    ```

    and insert it as a new value for the `bias=` parameter in the `SConstruct` file. Does smoothing by `sfsmooth` change the mean value?

11. Save the `SConstruct` file and run

    ```
    bash$ scons view
    ```

    to view improved images. Notice that `horizon.rsf` and `smoothed.rsf` files are not being rebuilt. SCons is smart enough to know that only the part affected by your changes needs to be updated.

As shown in Figure 2, smoothing removes random amplitude fluctuations but at the same broadens the channel and thus makes the channel edge detection unreliable. In the next part of this tutorial, you will try to find a better solution by examining a simple one-dimensional synthetic example.

```
from rsf.proj import *

# Download data
Fetch('horizon.asc','hall')

# Convert format
Flow('horizon','horizon.asc',
     '''
     echo in=$SOURCE data_format=ascii_float n1=3 n2=57036 |
     dd form=native | window n1=1 f1=-1 |
     put
     n1=196 o1=33.139 d1=0.01 label1=y unit1=km
```

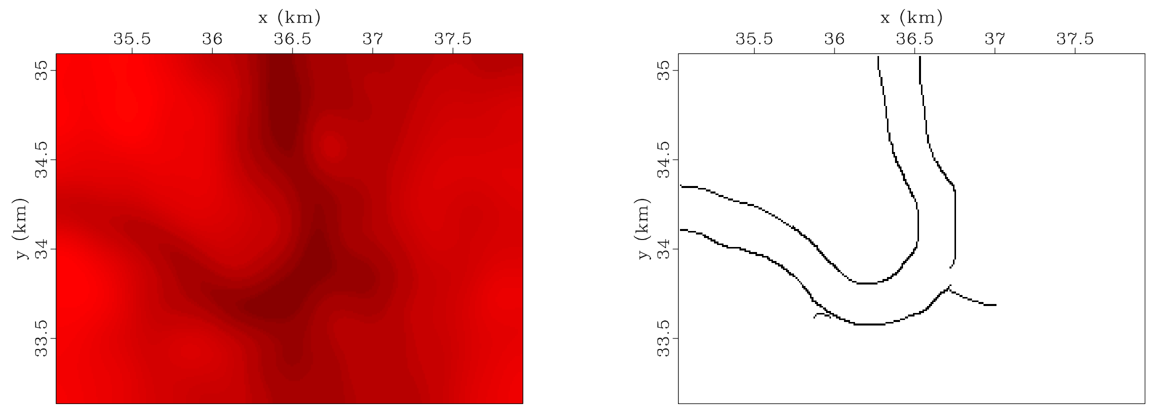


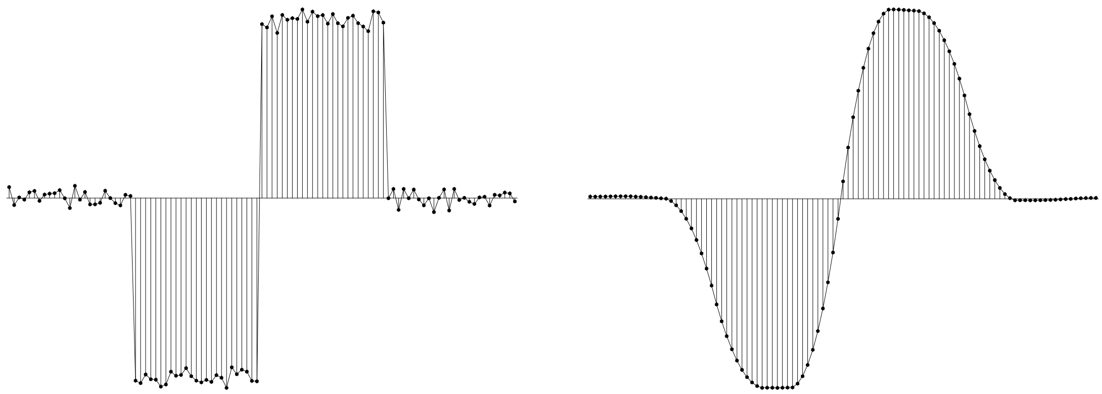
Figure 2: Depth slice from Figure 1 after smoothing (left) and output of edge detection (right).

```
        n2=291 o2=35.031 d2=0.01 label2=x unit2=km
        ''')

# Triangle smoothing
Flow('smoothed','horizon','smooth rect1=20 rect2=20')

# Display results
for horizon in ('horizon','smoothed'):
    # --- CHANGE BELOW ---
    Plot(horizon,'grey color=j bias=0 yreverse=n wanttitle=n')
    edge = 'edge-'+horizon
    Flow(edge,horizon,'canny max=98 | dd type=float')
    Plot(edge,'grey allpos=y yreverse=n wanttitle=n')
    Result(horizon,[horizon,edge],'SideBySideIso')

End()
```

1-D SYNTHETIC

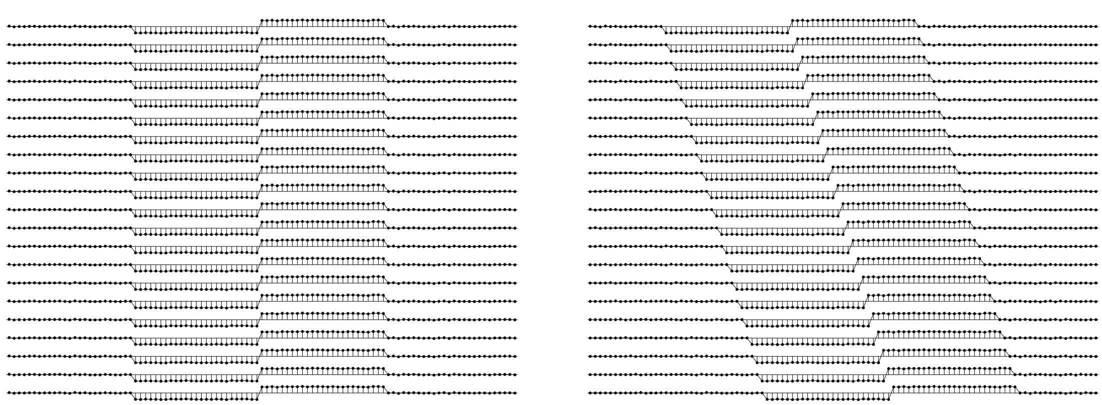
To better understand the effect of smoothing, you decide to create a one-dimensional synthetic example shown in Figure 3(a). The synthetic contains both sharp edges and random noise. The output of conventional triangle smoothing is shown in Figure 3(b). We see an effect similar to the one in the real data example: random noise gets removed by smoothing at the expense of blurring the edges. Can you do better?

To better understand what is happening in the process of smoothing, let us convert 1-D signal into a 2-D signal by first replicating the trace several times and then shifting the replicated traces with respect to the original trace (Figure 4). This creates a 2-



(a) (b)

Figure 3: (a) 1-D synthetic to test edge-preserving smoothing. (b) Output of conventional triangle smoothing.



(a) (b)

Figure 4: (a) Input synthetic trace duplicated multiple times. (b) Duplicated traces shifted so that each data sample gets surrounded by its neighbors. The original trace is in the middle.

D dataset, where each sample on the original trace is surrounded by samples from neighboring traces.

Every local filtering operation can be understood as stacking traces from Figure 4(b) multiplied by weights that correspond to the filter coefficients.

1. Change directory to the project directory

   ```
   bash$ cd ../local
   ```

2. Verify the claim above by running

   ```
   bash$ scons smooth.view smooth2.view
   ```

   Open the `SConstruct` file in a text editor to verify that the first image is computed by `sfsmooth` and the second image is computed by applying triangle weights and stacking. To compare the two images by flipping between them, run

   ```
   bash$ sfpen Fig/smooth.vpl Fig/smooth2.vpl
   ```

3. Edit `SConstruct` to change the weight from triangle

   $$W_T(x) = 1 - \frac{|x|}{x_0} \tag{1}$$

   to Gaussian

   $$W_G(x) = \exp\left(-\alpha\,\frac{|x|^2}{x_0^2}\right) \tag{2}$$

   Repeat the previous computation. Does the result change? What is a good value for α?

4. Thinking about this problem, you invent an idea[2]. Why not apply non-linear filter weights that would discriminate between points not only based on their distance from the center point but also on the difference in function values between the points. That is, instead of filtering by

   $$g(x) = \int f(y)\,W(x-y)\,dy\ , \tag{3}$$

   where $f(x)$ is input, $g(y)$ is output, and $W(x)$ is a linear weight, you decide to filter by

   $$g(x) = \int f(y)\,\hat{W}\left(x-y, f(x)-f(y)\right)\,dy\ , \tag{4}$$

   where and $\hat{W}(x,z)$ is a non-linear weight. Compare the two weights by running

[2] Actually, you reinvent the idea of *bilateral* or *non-local* filters (Tomasi and Manduchi, 1998; Gilboa and Osher, 2008).

```
bash$ scons triangle.view similarity.view
```

The results should look similar to Figure 5.

5. The final output is Figure 6. By examining `SConstruct`, find how to reproduce this figure.

6. **EXTRA CREDIT** If you are familiar with programming in C, add 1-D non-local filtering as a new MADAGASCAR module `sfnonloc`. Ask the instructor for further instructions.

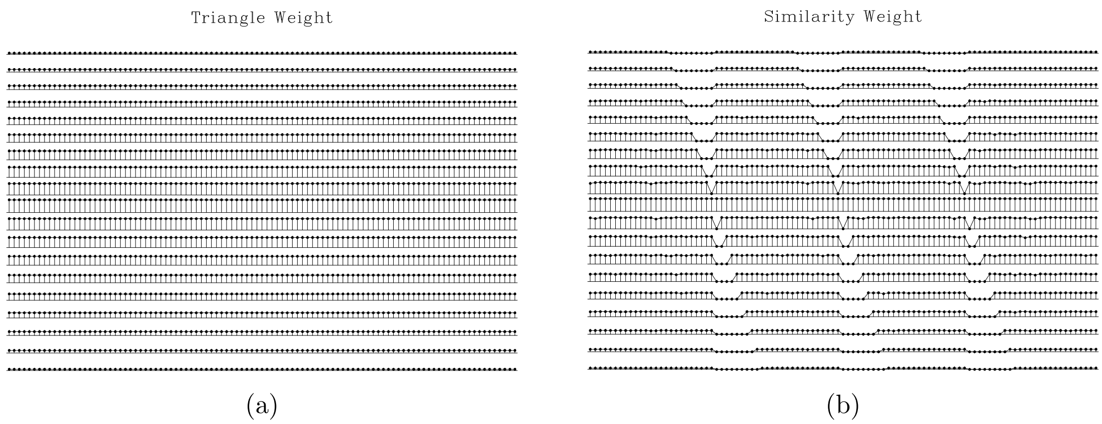


Figure 5: (a) Linear and stationary triangle weights. (b) Non-linear and non-stationary weights reflecting both distance between data points and similarity in data values.

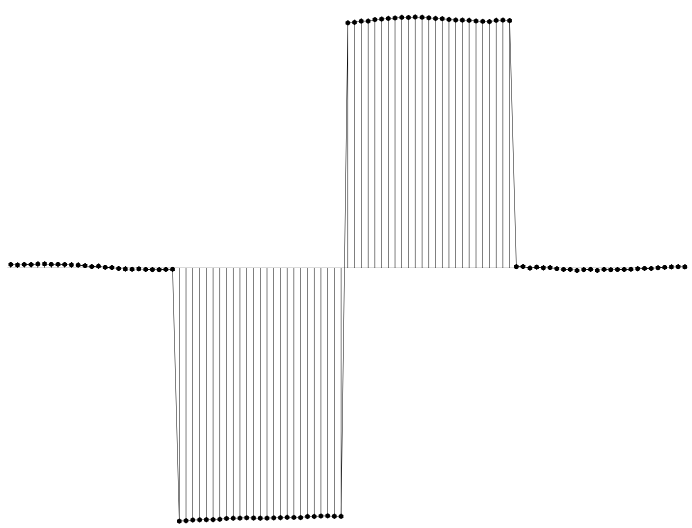


Figure 6: Output of non-local smoothing

Figure 6 shows that non-linear filtering can eliminate random noise while preserving the edges. The problem is solved! Now let us apply the result to our original problem.

```
/* Non-local smoothing. */
#include <rsf.h>

int main (int argc, char *argv[])
{
    int n1,n2, i1,i2, is, ns;
    float *trace, *trace2, ax, ay, t;
    sf_file inp, out;

    /* initialize */
    sf_init(argc,argv);

    /* set input and output files */
    inp = sf_input("in");
    out = sf_output("out");

    /* get input dimensions */
    if (!sf_histint(inp,"n1",&n1))
        sf_error("No n1= in input");
    n2 = sf_leftsize(inp,1);

    /* get command-line parameters */
    if (!sf_getint("ns",&ns)) sf_error("Need ns=");
    /* spray radius */

    if (!sf_getfloat("ax",&ax)) sf_error("Need ax=");
    /* exponential weight for the coordinate distance */

    trace = sf_floatalloc(n1);
    trace2 = sf_floatalloc(n1);

    /* loop over traces */
    for (i2=0; i2 < n2; i2++) {
        /* read input */
        sf_floatread(trace,n1,inp);

        /* loop over samples */
        for (i1=0; i1 < n1; i1++) {
            t = 0.;

            /* accumulate shifts */
            for (is=-ns; is <= ns; is++) {
                if (i1+is >= 0 && i1+is < n1) {

```

```
                        /* !!!MODIFY THE NEXT LINE!!! */
                        t += trace[i1+is]*expf(-ax*is*is);
                    }
                }

                trace2[i1] = t;
            }

            /* write output */
            sf_floatwrite(trace2,n1,out);
        }

        /* clean up */
        sf_fileclose(inp);
        exit (0);
}
```

SOLUTION

1. Change directory to the project directory

   ```
   bash$ cd ../channel2
   ```

2. By now, you should know what to do next.

3. Two-dimensional shifts generate a four-dimensional volume. Verify it by running

   ```
   bash$ scons local.rsf
   ```

   and

   ```
   bash$ sfin local.rsf
   ```

   View a movie of different shifts by running

   ```
   bash$ scons local.vpl
   ```

4. Modify the filter weights by editing `SConstruct` in a text editor. Observe your final result by running

   ```
   bash$ scons smoothed2.view
   ```

5. The file `norm.rsf` contains the non-linear weights stacked over different shifts. Add a `Result` statement to `SConstruct` that would display the contents of `norm.rsf` in a figure. Do you notice anything interesting?

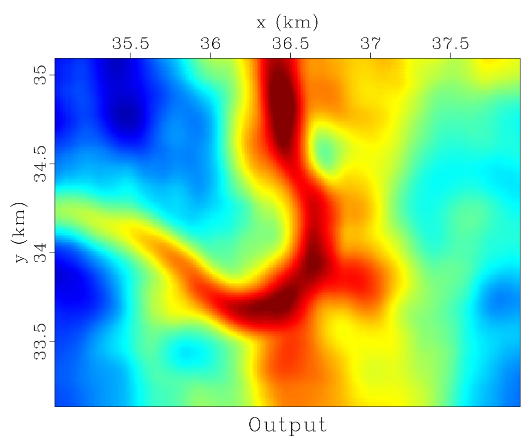
6. Apply the Canny edge detection to your final result and display it in a figure.

7. **EXTRA CREDIT** Change directory to `../mona` and apply your method to the image of Mona Lisa. Can you extract her smile?

```
from rsf.proj import *

# Download data
Fetch('horizon.asc','hall')

# Convert format
Flow('horizon2','horizon.asc',
     '''
     echo in=$SOURCE data_format=ascii_float n1=3 n2=57036 |
     dd form=native | window n1=1 f1=-1 |
     add add=-65 | put
     n1=196 o1=33.139 d1=0.01 label1=y unit1=km
     n2=291 o2=35.031 d2=0.01 label2=x unit2=km
     ''',stdin=0)
Result('horizon2','grey  yreverse=n color=j title=Input')

# Spray
Flow('spray','horizon2',
     '''
     spray axis=3 n=21 o=-0.1 d=0.01 |
     spray axis=4 n=21 o=-0.1 d=0.01
     ''')

# Shift
Flow('shift1','spray','window n1=1 | math output=x2')
Flow('shift2','spray','window n2=1 | math output=x3')

Flow('local','spray shift1 shift2',
     '''
     datstretch datum=${SOURCES[1]} | transp |
     datstretch datum=${SOURCES[2]} | transp
     ''')
Plot('local','window j3=4 j4=4 | grey color=j',view=1)

# --- CHANGE BELOW ---
# try "exp(-0.1*(input-loc)^2-200*(x3^2+x4^2))"
```

```
Flow('simil','spray local',
     '''
     math loc=${SOURCES[1]} output=1
     ''')

Flow('norm','simil',
     'stack axis=4 | stack axis=3')

Flow('smoothed2','local simil norm',
     '''
     add mode=p ${SOURCES[1]} |
     stack axis=4 | stack axis=3 |
     add mode=d ${SOURCES[2]}
     ''')
Result('smoothed2','grey yreverse=n color=j title=Output')

End()
```

Figure 7: Your final result.




```
from rsf.proj import *

# Download data
Fetch('mona.img','imgs')

# Convert to standard format
Flow('mona','mona.img',
     '''
     echo n1=512 n2=513 in=$SOURCE data_format=native_uchar |
     dd type=float
     ''',stdin=0)

Result('mona',
       '''
```

Figure 8: Can you apply your algorithm to Mona Lisa?

```
        grey transp=n allpos=y title="Mona Lisa"
        color=b screenratio=1 wantaxis=n
        ''')

End()
```

WRITING A REPORT

1. Change directory to the parent directory

   ```
   bash$ cd ..
   ```

   This should be the directory that contains `paper.tex`.

2. Run

   ```
   bash$ sftour scons lock
   ```

   The `sftour` command visits all subdirectories and runs `scons lock`, which copies result files to a different location so that they do not get modified until further notice.

3. You can also run

   ```
   bash$ sftour scons -c
   ```

   to clean intermediate results.

4. Edit the file `paper.tex` to include your additional results. If you have not used LaTeX before, no worries. It is a descriptive language. Study the file, and it should become evident by example how to include figures.

5. Run

   ```
   bash$ scons paper.pdf
   ```

   and open `paper.pdf` with a PDF viewing program such as **Acrobat Reader**.

6. Want to submit your paper to *Geophysics*? Edit `SConstruct` in the paper directory to add `option=manuscript` to the `End` statement. Then run

   ```
   bash$ scons paper.pdf
   ```

   again and display the result.

7. If you have LaTeX2HTML installed, you can also generate an HTML version of your paper by running

   ```
   bash$ scons html
   ```

   and opening `paper_html/index.html` in a web browser.

REFERENCES

Canny, J., 1986, A computational approach to edge detection: IEEE Trans. Pattern Analysis and Machine Intelligence, **8**, 679–714.

Fomel, S., and G. Hennenfent, 2007, Reproducible computational experiments using SCons: 32nd International Conference on Acoustics, Speech, and Signal Processing (ICASSP), IEEE, 1257–1260.

Gilboa, G., and S. Osher, 2008, Nonlocal operators with applications to image processing: Multiscale Model & Simulation, **7**, 1005–1028.

Tomasi, C., and R. Manduchi, 1998, Bilateral filtering for gray and color images: Proceedings of IEEE International Conference on Computer Vision, IEEE, 836–846.