International Journal of Wavelets, Multiresolution and Information Processing

BeamLab and Reproducible Research

David L. Donoho

Stanford University, Department of Statistics
Stanford, CA 94305, United States of America
donoho@stat.stanford.edu

Xiaoming Huo

Georgia Institute of Technology, School of ISyE
Atlanta, GA 30332, United States of America
xiaoming@isye.gatech.edu

Received (4 April 2004)
Revised (10 May 2004)
Communicated by (xxxxxxxxxx)

In the first 'Wavelets and Statistics' conference proceedings [1], our group published 'Wavelab and Reproducible Research', in which we advocated using the internet for publication of software and data so that research results could be duplicated by others. Much has happened in the last decade that bears on the notion of reproducibility, and we will review our experience. We will also describe a new software package BeamLab containing routines for *multiscale geometric analysis*, and describe some of its capabilities.

BeamLab makes available, in one package, all the code to reproduce all the figures in our recently published articles on beamlets, curvelets and ridgelets. The interested reader can inspect the source code to see what algorithms were used, and how parameters were set to produce the figures, and will then be able to modify the source codes to produce variations of our results.

Some new examples of numerical studies based on BeamLab are provided here.

Keywords: WaveLab; Beamlab; Reproducible research.

AMS Subject Classification: 68N99, 68U99, 62-07.

1. Introduction

In the bad old days before ubiquitous computation and communication, which can be called the *ancien regime*, research in the mathematical sciences was a highly intellectual endeavor, in which publication required careful discussion and proof. This was true not only in, say, applied mathematics, statistics and computer science, but also, to our knowledge, in geophysics, astronomy, chemistry and even social sciences. Towards the end of that era, papers of a new stamp began to appear, so-called 'computational science' papers, in which procedures were motivated sketchily or heuristically at best, and according to the principle that the 'proof

of the pudding is in the eating', justification was purely empirical. Software was written, computational experiments were conducted, and the results were reported, often with some figures or tables. To those who were schooled under the *ancien regime*, the new 'computational science' seemed pretty shaky, thin on analysis and easily dismissed. A typical attitude that one author (DLD) remembers hearing in the late 1970's (from a practical seismologist) was

'they only ever publish the one example where their idea works'.

The implication being that by the new rules in this emerging world, *any idea*, no matter how silly, if one took the pains to analyze it, *could be published*, since you could always find (or make up) *one example* where it *did something* suitable for pictures.

By the early 1990's it was clear that this emerging model would sweep field after field, to the point that the situation started to seem dangerous and/or scandalous. The basic problem about computational science in the twilight of the *ancien regime* was the lack of *reproducibility*. Someone could sketchily describe an ambitious computation that took many months to program and then display a *purported* computational result. There was literally no chance of exposing fundamental errors in such work, and such errors could creep in anywhere, as algorithmic problems, software implementation mistakes, data management problems, shading the truth (e.g., cherry-picking of conflicting results to overstate success), even outright fraud. Thus using the term 'science' in computational science seemed a dangerous illusion.

Around 1990, Dr. Jon Claerbout, an exploration geophysicist at Stanford, proposed a paradigm of *reproducible computationally-driven research* [15]. He envisioned a situation where every scientific article was published in a hypertext format, accompanied by the complete software environment that conducted the experiments, and by the command scripts that generated all the computational results. This would allow both reproduction of the results, and, in some cases, modification of the computational procedures or of the original data in order to see how the computational methods would work on other examples.

Inspired by Claerbout's vision, DLD began to work within the paradigm of reproducible research, which led to the article 'Wavelab and Reproducible Research' [6].

In the current paper we would like to briefly review the original ideas of reproducibility, the arguments in favor of reproducibility, and some counter-arguments we have learned over the ensuing decade. We will discuss the outlook for the future, and then describe a new package of multiscale tools, BeamLab, implemented under this paradigm.

The rest of this article is organized as follows. In Section 2, we review our thoughts and experience with reproducible research. In Section 3, we describe demonstrations in BeamLab. Section 4 describes the structure of BeamLab. In Section 5, we illustrate how BeamLab can be utilized as a research tool to solve new problems. Some concluding marks are made in Section 6.

2. Reproducible Research

2.1. *The Vision*

In an ideal situation, every scientific paper with computational results would be published in hypertext format, linking to code-bodies, command scripts, data viewers, and so on, to support the analysis in the paper. It should be possible to rerun the computations, and even to do different computations with the same code, for example with other data, or with different parameters.

This idea cannot always be followed, for a variety of reasons. These can include proprietary restrictions on the data and on the algorithms, reliance on commercially licensed software, or reliance on special-purpose computers or specific capabilities of a particular operating system. But we think all readers will realize that the model can *often* be followed. For example, by using Adobe Acrobat [16], one can create `PDF` (portable documentary format) files which contain hyper-links to URL's (universal resource locators) across the web and which might lead to all sorts of side effects, including downloading and installation of software packages and data on the reader's computer. Modern quantitative programming environments, commercial systems like MATLAB [17], S-PLUS [18], and IDL [19], and freeware systems, such as R [20] and OCTAVE [21], provide a convenient setting for computational experiments, which operate the same way on different types of computers. JAVA virtual machine [23] is another option.

There is in principle no technological excuse for not using this approach as the default mode of publication in computational science. It demands, however, a substantial change in habits of work. In order to produce work that can be reproduced, it is essential to start out from the beginning of the project knowing how to do this, and to follow a plan. It simply will not do to try to clean things up after the works. In fact, most people who develop code without special precautions simply do not know what the code is doing after they put it aside even briefly.

The details of work habits that are involved depend greatly on the way in which reproducibility is to be achieved (e.g. with JAVA, or MATLAB, etc.). Whatever they are, they must support two basic ideas:

- Documenting and formatting code so that some months later, its intentions and restrictions are clear to the authors and others. This can be addressed by segmenting code into general-purpose tools and special-purpose scripts, and following a documentation standard for each tool and script, explaining inputs, outputs, and assumptions.
- Managing code development so that code obeys certain rules: e.g., so that it can run on many different machines or so that it can be used for more than one purpose. This can be addressed by relying on a high-level environment such as JAVA or MATLAB and enforcing standard practices in the naming of files, documentation, and directory structures.

An idea of the discipline involved can be gleaned by examining the structure of

WaveLab and BeamLab.

2.2. *Why we should do it*

Some colorful figures might actually *insist* on reproducible research:

- *The Curmudgeon Scientist.* This one is aghast at the widespread practice of developing data analysis and data processing methodology based on the standard that 'it seems to run well'. Such a scientist is not convinced unless he/she can tear apart the tools and verify their correct construction. By publishing tools, others allow the curmudgeon to indulge his craving for transparency.
- *The Curmudgeon Advisor.* This one is annoyed at graduate students who seem so disorganized that he/she comes to doubt whether *anything* good can be expected of them. By demanding that students make all their work reproducible tools, the curmudgeon indulges his/her craving for transparency of the students work.
- *The Curmudgeon Congressperson.* This is the political figure who wants to know what all that tax money is going for, and wondering why it can not be made available for the use of all taxpayers. An aide tells him/her that the NSF (United States, National Science Foundation) actually *requires* NSF-funded software and data to be publicly available, but not many people obey this requirement. The congress-person asks: can't we somehow *force* people to obey this rule?
- *The Flamboyant Revolutionary.* This is the young turk impatient with talks where big shots talk about tools and methods that are not available to young scientists to actually deploy. He or she thinks: what is the point of talking about something if no-one can use it *now*.

Admittedly, these are all people saying *other* people should follow these rules.

At a less colorful level, there are good reasons why the people who actually *do* the research should *want* to support reproducible research. This is crucial because it is those people who have to adopt and impose a disciplined way of working. Our list of reasons includes:

- *Scientific Quality.* Results are simply better when you know other people are going to be looking over your shoulder.
- *Scientific Productivity.* Once you force yourself to work in a way that others can re-use your ideas, then you can re-use them as well. You will find you are able to build incrementally on your own work and go farther and deeper in each direction than if you had not gone to this effort.
- *Diffusion of Ideas.* If you want people to use your ideas, make it as easy as clicking on a button to trigger a download/installation/execution of your software. If you want your method to be more widely applied than a

competitor, hope that he does not publish his methods in an easy-to-access way, and just publish your tools. The world will flock to your methods.
- *Graduate Education.* It is very valuable for graduate students to know that their software will be seen by others in principle, that there is a discipline to create for publication, and to be shown how to do all this.
- *Mentoring.* Students get to mentor the next generation by creating tools which future students can learn from and emulate.
- *Laboratory Stability.* As individuals leave a laboratory, if they have followed reproducibility, their tools stay behind (and come along as well), allowing continuity in their previous lab and in their new institution.
- *Recruiting.* New recruits into a field can be obtained from among people who download and use computational tools they learned about from articles in that field.
- *Collective Debugging.* You will have better tools if people all around the world are using and commenting on your published results.

We found all these arguments compelling for us in the early 1990's. We find them even more compelling today, but we will point out below that not everyone is persuaded.

2.3. *Signs of the Times*

When reproducible research first caught on, the vision seemed far reaching and challenging to many. In the last decade several related trends have emerged which are so well known that they can be considered part of today's global culture. Given these trends, advocacy of reproducible research today would seem unsurprising, particularly to young scientists. The main trends include:

- *Self Publication.* People everywhere in all walks of life now have websites and publish `PDF` files.
- *Open Source Software.* Collective efforts have produced the Linux operating system which runs on tens of millions of computers worldwide. Under this system, the operating system is free and freely distributable, in contrast to proprietary systems like Microsoft Windows.
- *Mobile Code.* It is possible to write code that runs on many kinds of computers, from supercomputers to mobile phones, using systems like JAVA; in science, systems like MATLAB and IDL are near-universal.
- *Client/Server Software.* It is possible to set up servers in any university or research lab, offering specific computational services or database services to users all around the world.

In today's world even traditionally closed proprietary products like operating systems and newspapers, or computer time, are being offered free over the internet. It seems anachronic for a researcher to 'hoard' the details of a computational experiment.

2.4. *Arguments Against Reproducibility*

Despite all this, there are good arguments against reproducibility. By 'good' we mean, not that we approve or would be persuaded, but that we know people who would approve or be persuaded. These are the following:

- *It is More Work.* Because one has to follow a publication and programming discipline, it is more work to do this, certainly initially.
- *It is More Trouble.* Because one is exposing the details of one's work, someone else might find a mistake and "cause trouble".
- *I'm Giving Up Future Projects.* Because one is exposing the details of one's work, others might poach and use our work to do projects we were planning to do.
- *Science Advances through Proprietary Interests.* This, the most sophisticated argument, was explained to one author (DLD) by a crystallographer. The idea here is that each scientific laboratory tries to develop its own 'secret sauce', a collection of procedures and methods. This is a kind of valuable property which leads to specialization, trade, and socialization. The proprietary interest leads to an exchange economy. "I know how to do X and you know how to do Y, if you collaborate with me I will contribute X and you will contribute Y." This organizes a society of scientists into sharing individuals. Ultimately through exchange, much more gets done than if individuals thought they could get whatever they need by downloading information from the internet and mastering all of it themselves. It also helps form young scientists by providing a social mentoring context to acquisition of specialized knowledge. By joining a laboratory, young scientists get exposed to proprietary methods, which endows them with valuable intellectual capital on which they can build a future career.

In practice, of course, the argument of *It is More Work* probably outweighs all others.

2.5. *The Proof of the Pudding...*

Of course, each researcher decides for him/herself whether he/she will attempt reproducibility. DLD was mostly convinced by ideological reasons, but fully understands why others would not. He believes however, that the decision to pursue reproducibility with WaveLab was fully rewarded. Here are two reasons:

- *Influence.* There was a time when the WaveLab toolbox was the only freely available software for wavelet analysis. In practice, this meant that tens of thousands of people downloaded and used this software in many ways. As a result, it amounted to substantial 'publicity' and 'impact'. In particular, many commercial software wavelet packages clearly relied on WaveLab itself, as the starting point for designing wavelet packages and

for documenting such packages. The examples and datasets in the WaveLab toolbox were also used in numerous textbooks and monographs.

- *Imitation.* Dozens of papers were written over the last ten years following the algorithm of modifying the wavelet denoising methods in papers [7,8], and modifying the software in WaveLab to illustrate the new methods and compare with older ones. The availability of WaveLab made it extremely easy to write papers of this kind. One only had to slightly modify the existing scripts in WaveLab to invoke the new method, and, with very little effort, the new method could be tested quantitatively against the performance of previous methods.

The upshot of all this was that the papers which WaveLab reproduces became highly cited. Some of the papers had well over 200 citations and were in the top few cited mathematical sciences papers in the year they appeared. David L. Donoho and Iain M. Johnstone at one point were named among the top three cited mathematical scientists in the 1990's (source: ISI, makers of Science Citation Index). DLD also mentions that after he was inducted to the National Academy of Sciences, several senior members approached him to say that they enjoyed using the WaveLab software. Presumably the visibility of WaveLab favorably influenced the voting, although doubtless there were many other factors at work.

2.6. *Your Choice*

As a reader, you obviously have to weigh many factors in deciding whether you will work to enable reproducibility. This paper illustrates one approach as it has been applied in a new field — Multiscale Geometric Analysis. The reasons have primarily been to encourage research assistants and postdocs to learn how to create reliable scientific tools, but also we hoped to encourage the spread of new methods and to recruit new scientists into this area. In reviewing the above pro's and con's you may have completely different factors at work in making your decision.

3. Demos in BeamLab

BeamLab makes available in one package the tools to reproduce figures that have been used in recent publications by authors and their collaborators. The references are given below. The demos in BeamLab serve as a starting point of browsing.

3.1. *Overview*

BeamLab is a library of Matlab routines for beamlet, curvelet and ridgelet analysis. The library is the basis for multiscale analysis research pioneered by the authors, and can be used to reproduce the figures in their published articles, and to recompute those figures with variations in the parameters.

The library is available free of charge over the Internet by WWW (world wide web) access. It is downloadable at

`http://www-stat.stanford.edu/~beamlab.`

The package can be considered an extension of WaveLab [4,5,6]. In fact, the structure of BeamLab is similar to WaveLab and some functions actually depend on scripts from WaveLab. We believe that by studying these scripts, one can quickly learn the practical aspects of various multiscale analysis systems. Some examples will be provided later.

There are other resources for obtaining information about BeamLab. There is a "*BeamLab Architecture*" guide, which gives details about how BeamLab is constructed and maintained. We also provide updated electronic versions of papers by the authors [3,10,11,12,13,14]. See URLs (universal resource locators) in the following description.

The body of software is under continuing development by a team of researchers. The main aim is research — to develop specific tools for specific goals in multiscale geometric analysis. We conduct our research, planning to implement our tools for publication in BeamLab. The discipline that this software entails makes our research of a higher quality than otherwise possible.

3.2. *Demos*

BeamLab gives readers the possibility of reproduce results in our papers. Here is an up-to-date listing of demos in version 0.200, and the articles to which they correspond:

- BMIADemo - demo for paper "Beamlets and Multiscale Image Analysis" [10].
- BL3DDemo - demo for paper "3D Beamlets" [13].
- DCRTDemo - demo for paper "Digital Curvelet Transform: Strategy, Implementation and Experiments" [14].
- DRTDemo - demo for paper "Digital Ridgelet Transform" [11].
- FSSDemo - demo for paper "Fast Slant Stack" [3].
- RPDemo - demo for paper "Ridgelet Packets" [12].

After downloading and installing BeamLab, following the instructions at the website, you can browse around to see what files BeamLab contains. These demos will tell readers what BeamLab can do.

For readers who want to locate the files that are used for demos, the subdirectory `BeamLab/Papers` contains several subdirectories; each one of these contains scripts that were used to produce figures in published articles.

Each subdirectory contains a "demo" file (e.g. `BMIADemo.m` in directory `BMIA`, `BL3DDemo.m` in directory `BL3D`, and so on). This file generates figures for a corresponding article.

When you invoke that file in Matlab by typing its name (without the `.m` extension) in the command line, a menu bar will appear on the screen. By mouse-clicking

the menu button `Run All Fig`, you will see, in sequence, each figure in the corresponding article. As each figure appears in MATLAB's figure window, the command window will contain narrative explaining what you see in the figure window.

3.3. *Snapshots*

We strongly recommend obtaining the articles corresponding to these demos, reading through them and inspecting the figures in hard-copy form, and then asking yourself a question: what would happen if I changed the parameters in this algorithm. Later on, you may delve into the scripts, creating slightly modified ones that recreate the figures with modified parameters. Most of these papers are available at

`http://www-stat.stanford.edu/~donoho/reports.html`.

Some of these experiments may take a long time (hours or even days). For those case, instructions are given on how to obtain a pre-computed results.

In this section, we demonstrate what can be done by calling existing demos in BEAMLAB. In Section 5, we will provide new numerical examples, which have not been included in BEAMLAB, however can be done by calling BEAMLAB functions.

3.3.1. 2-*D Beamlets*

Beamlets (at 2-D) are introduced by authors [9,10] as a tool for multiscale analysis of linear and curvilinear features. In paper [10], several interesting applications are illustrated by figures. As an example, Figure 11 in paper [10] shows how beamlet can be utilized to compute something called β-thickness, which then can be used to describe the smoothness of a geometric object that is merely made by curves. BEAMLAB offers the possibility to reproduce this figure. More specifically, if you run 'BMIADemo' from the MATLAB command line, the window (Figure 1 (a)) will appear.

We explain with a little bit more details on how our Demo window works. At the beginning, the figure pane is empty. By clicking the numbers in the Figure pane, one can reproduce the corresponding figures. For example, if '11' is selected from the right bar, the plot in Figure 1 (a) will be generated, reproducing Figure 11 from paper [10]. (The slopes of the curves in the right column are associated with the roughness of the curves in the left column.) Users can examine the source code by clicking the button `See Code`. They will find that MATLAB actually calls for the function **BMIAfig11.m** to create the figure.

Note that the computing time associated with some of these figures is intolerably long (several hours on a 1GHz PC), so the default option of these demos is to prompt the user to download the pre-computed results. For example, the results for paper [10] are available as **.mat** files at

`http://www.isye.gatech.edu/~beamlab/data/`.

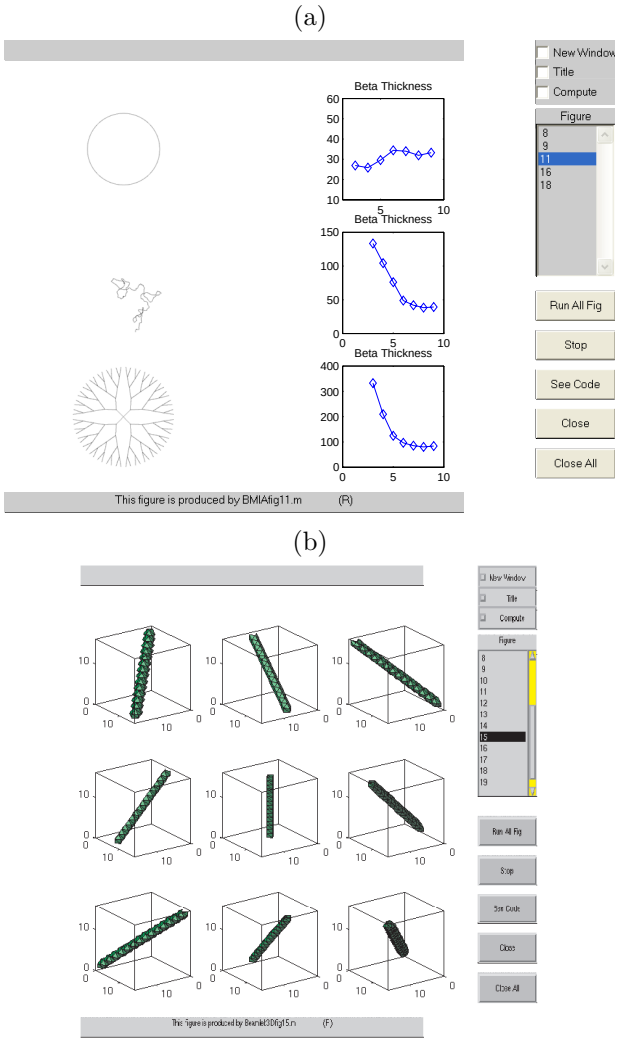

Fig. 1. (a) A demo window for Beamlets and its application of measuring roughness for curves. (b) Some examples of 3-D beamlets.

If the user wants to create the figure from scratch, the `Compute` box should be checked prior to the selection of the figure number.

A closely related new example is reported in Section 5.1.

3.3.2. 3-*D Beamlets*

If one types 'BL3DDemo' in MATLAB, a window similar to Figure 1 (a) will appear. Supposing the Figure 15 in the paper [13] is of interest, one can redo this figure by clicking the number `15` in the `Figure` window. After a few seconds, Figure 1 (b)

will appear.

The figure displays some 3D beamlets. If we want to see how they are generated, we may simply click the button `See Code`. A new window will pop up displaying the body of the function `Beamlet3Dfig15.m`. This is the script that is used to generate the above figure. After examining the figures of interest, users can close the demo by clicking the button `Close All`.

3.3.3. *Digital Curvelet Transform*

We have included a preliminary version of Digital Curvelets written by David L. Donoho in 1998. This version of Curvelets was described on a technical report [14]. The code is a little outdated, since it is based on an old ridgelets code, but it still illustrates the point that curvelets can be discretized and codified, preserving most of its nice theoretical qualities. The curvelets demo is still under construction, but a selection has been included in the current release. Typing 'DCTdemo' in the command prompt and selecting `4` or `5` it will show a description of the Analysis and Synthesis of the figure BigMac, as shown in the two panels of Figure 2.

3.3.4. *Digital Ridgelet Transform*

For someone interested in studying the digital ridgelet transform, the demo 'DRT-Demo' will be of interest. Orthonormal ridgelets can be shown by selecting the number `8` in the right pane of the Demo window. The user can choose to load the data to reproduce the figure or to compute the transform by checking the box `Compute`. Several wait bars will appear while computing each of the panels of the figure. Figure 3 shows some digital orthogonal ridgelets.

Some of the figures in this demo are computationally intensive, so we recommend to load the data while exploring the figures for the first time.

3.3.5. *Fast Slant Stack*

Figure 4 illustrates the idea of *shearing* that is reported in the paper [3]. It can be evoked by typing `FSSDemo` in `Matlab`.

Several workouts have also been devised for the illustration of the Slant Stack construction, and may be run by typing `guiparFSSFig1` to `guiparFSSFig4` in the command prompt of MATLAB.

3.3.6. *Ridgelet Packets*

In paper [12], Ridgelet Packets are introduced. A key idea is to implement a repartition in the Fourier domain. Figure 5 (a) provides a graphical illustration. This is part of `RPDemo`.

The Figure 5 (b) shows some results by running the algorithm that is reported in article [12], with a fixed type of entropy, taking several values. A workout was written

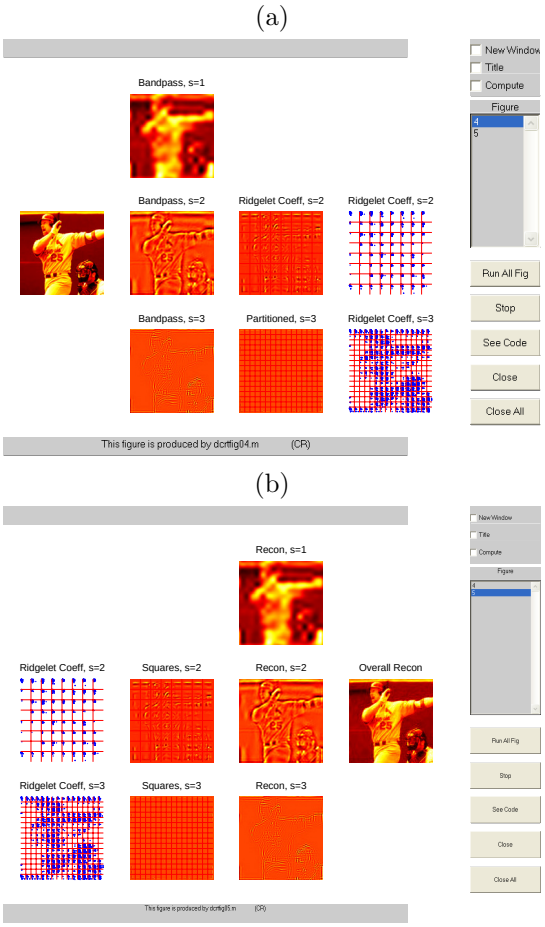

Fig. 2. An example of digital curvelet transform of the Bigmac image. (a) Curvelet Analysis of Bigmac. (b) Curvelet Synthesis of Bigmac.

to allow the user to explore all the parameters of the Ridgelet Packets algorithm provided in `BeamLab`, which can be called by typing `BestPartitionDemo` in the command line. The Figure 6 shows a snapshot of the Graphical User Interface (GUI) from this demo.

4. Structure of BeamLab

The BeamLab library contains `.m` files (Matlab code), `.mex` files (compiled dynamically loadable code), datasets, documentation, scripts and workouts (both also `.m` files) for reproducing the figures in articles by the authors.

The whole library consists of over 500 files and 50 subdirectories. It requires more than 16 MB on disk once it is downloaded, decompressed and installed and

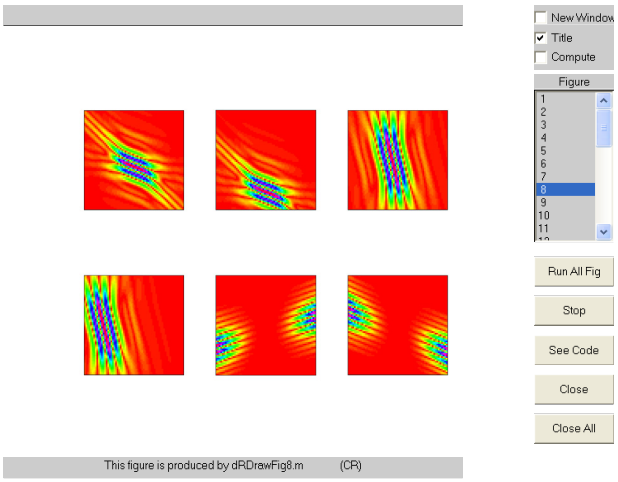

Fig. 3. Several ortho-ridgelets.

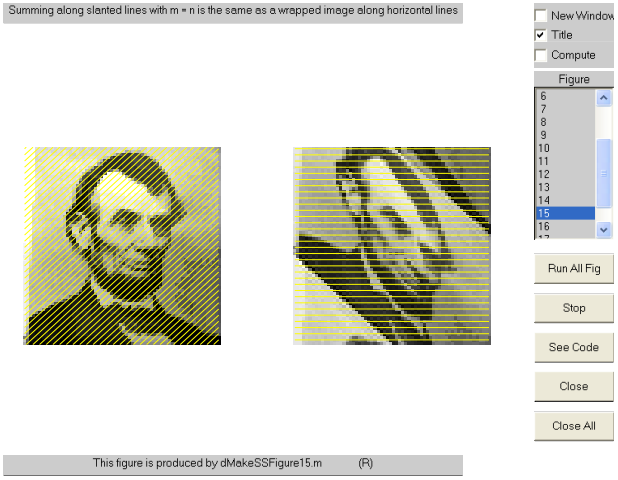

Fig. 4. An illustration of shearing from the paper about Fast Slant Stack.

takes possibly up to 180 MB disk space for output data. We recommend running the system on a machine with a minimum of 64 MB RAM.

If you just snoop around in the BeamLab file structure, you will notice many directories and a great range of different information about the system itself and what it can do. We list here some basic facts.

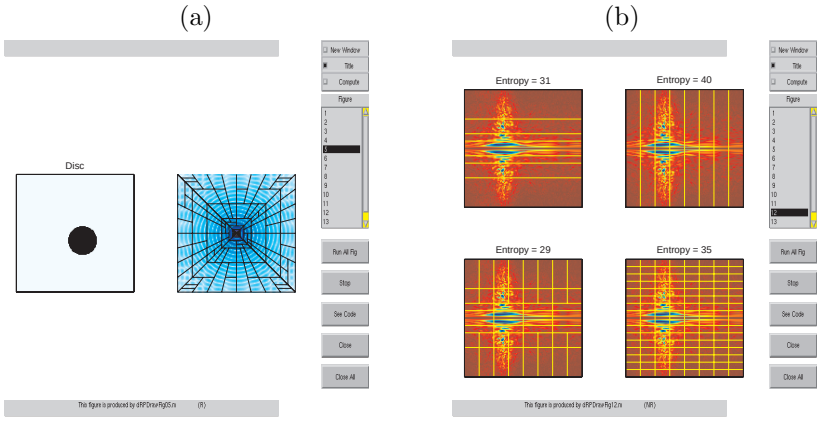

Fig. 5. (a) A new way to partition in the Fourier domain. A figure that is used in the paper on Ridgelet Packets. (b) Results of running an algorithm of Ridgelet Packets with different algorithmic parameters.

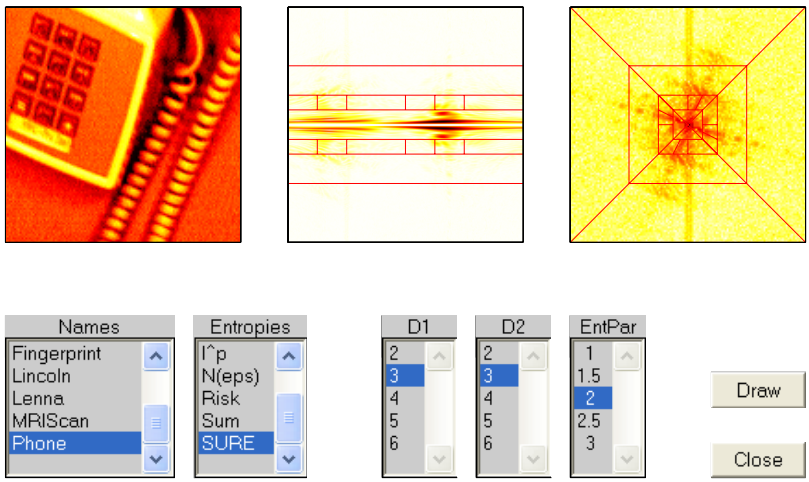

Fig. 6. Results of running BestPartitionDemo on Phone.

4.1. *Contents Files*

Each directory has a `Contents.m` file, which explains the contents and purpose of that directory. The directory `SlantStack` contains the central wavelet transform tools; its `Contents.m` file looks as follows:

```
% SlantStack: Contents v.200 -- Digital Radon Transform
%
% The routines in this directory perform digital radon transforms
```

```
% and associated functions (e.g. ridgelet transforms). Some
% of these tools and examples require Wavelab .80X to run properly.
%
%
%          Fast Radon Transforms
%
% FastSlantStack       -  Digital Radon Transform
% Inv_FastSlantStack   -  Inverse Radon Transform
% Adj_FastSlantStack   -  Adjoint Radon Transform
%
%           Slow Radon Transform
%
% SlowSlantStack       -  Digital Radon Transform
% Inv_SlowSlantStack   -  Inverse Radon Transform
% Adj_SlowSlantStack   -  Adjoint Radon Transform
%
%           Pseudopolar FFT
%
% PseudopolarFFT       -  Fast, Pseudopolar FFT
% Inv_PseudopolarFFT   -  Inverse Pseudopolar FFT
% Adj_PseudopolarFFT   -  Adjoint Pseudopolar FFT
%
%           Support for Pseudopolar FFT (PPFT)
%
% PtP                   - Gram operator of PPFFT
% Inv_PtP               - Inverse Gram Operator
% PrecondPPFT           - Preconditioner for PPFT
%
%           Ridgelet Transform
%
% FastRidgeletTransform       -  Digital Ridgelet Transform
% Inv_FastRidgeletTransform   -  Inverse Ridgelet Transform
% Adj_FastRidgeletTransform   -  Adjoint Ridgelet Transform
% FastOrthoRidgeTransform     -  Ortho Ridgelet Transform
% Inv_FastOrthoRidgeTrans     -  Inverse Fast Ortho Ridgelet
% Transform
%
%           Utilities Supporting all This
%
% FractionalFFT_mid0        - centered Fractional FFT
% fft_mid0                  - centered fft
% ifft_mid0                 - centered ifft
% fftshift1d                - fftshift on columns
```

```
% Shift1dSignal             - interpolate 1d signal
% UFWT_CDJV                 - unpreconditioned boundary wavelet
% analysis
% UIWT_CDJV                 - unpreconditioned boundary wavelet
% synthesis
%
```

4.2. *Help for Functions*

Each function in BeamLab has help documentation. For example, `FastSlantStack` is a basic transform routine in BeamLab. If you are in Matlab and type `help FastSlantStack`, Matlab will type out the following documentation:

```
FastSlantStack: Radon Transform for Discrete Data
   Usage:
      S = FastSlantStack(X)
   Inputs:
     X      n*n matrix (x,y)
   Outputs:
     S      2n*2n matrix (t,theta)
   Description:
     Sums array along lines.
```

4.3. *Source Browsing*

Most of the algorithms in BeamLab are available for inspection — including some of those that are actually implemented by fast complied C code as `.mex` files. For example, if you are in Matlab and type `type FastSlantStack` you get the following documentation:

```
function S = FastSlantStack(X)
% FastSlantStack: Radon Transform for Discrete Data
%  Usage:
%     S = FastSlantStack(X)
%  Inputs:
%    X      n*n matrix (x,y)
%  Outputs:
%    S      2n*2n matrix (t,theta)
%  Description:
%    Sums array along lines.
%
    Pre = 0; % No preconditioning
    P = PseudopolarFFT(X,Pre).';
```

```
    P = fftshift1d(P);
    S = fftshift1d(ifft(P));
%
%   Copyright (c) 2000. Amir Averbuch and David Donoho
%
%
% Part of BeamLab Version: 200
% Built:Thursday,14-June-2002 00:00:00
% This is Copyrighted Material
% For Copying permissions see COPYING.m
% Comments? e-mail beamlab@stat.stanford.edu
%
```

Notice that the source contains information about the author and date of compilation, as well as copyright of the routine. Also, the help information is built in as the first thing following the function header. Notice also that the Fast Slant Stack routine depends on other routines, such as `PseudopolarFFT` and `fftshift1d`, which are also part of BeamLab and can also be inspected at source level.

5. Adapting BeamLab to New Uses

Although the BeamLab package is designed for reproducing existing published results, it may also be adapted to new applications. In this section, we give three examples.

5.1. *Beamlet Edge Detection*

The readers of our paper [10] see that Figure 9 in that paper illustrates how to use beamlets as a multiscale tool to analyze the linear and curvilinear components in a pen-and-ink sketch called 'Picasso'. Suppose the reader is intrigued by this example, but has in mind a totally different area of application. Rather than applying the technique to a scanned image of fine art (as in Picasso), she/he would like to apply to noisy low-contrast imagery. The idea is that traditional edge detectors work at pixel scale and are very vulnerable to heavy noise. Following such edge detectors by beamlet detectors will create the possibility of noise rejection by averaging edge detector output along lines.

It is a relatively simple matter to adapt the existing BeamLab code to perform such analysis. First step: find the relevant code; second step: adapt the code to new purposes. So, the reader looks in **Papers/BMIA**, and locates the script **BMIAfig09.m** which produced the original figure of interest; the script calls **fig09work.m**. The Matlab command `which fig09work` locates the relevant file, which can be inspected and analyzed.

The reader clones the relevant code: giving it new names, and then edits the clones, inserting some preprocessing steps designed to test the reader's idea. The

results of the first task produced by the reader are shown in Figure 7 of this paper.

Figure 7(a) shows a noisy version of HalfDome, with a relatively sharp edge against a noisy background. A conventional edge filter is applied in Panel 7(b), and a huge number of false edges is detected. Next, the beamlet transform of the edge detector results is taken. In figures 7(c)-(e), the 100 most significant beamlet coefficients are shown at each beamlet scale, by depicting the beamlets themselves. Clearly, the feature of interest is present in all these scales.

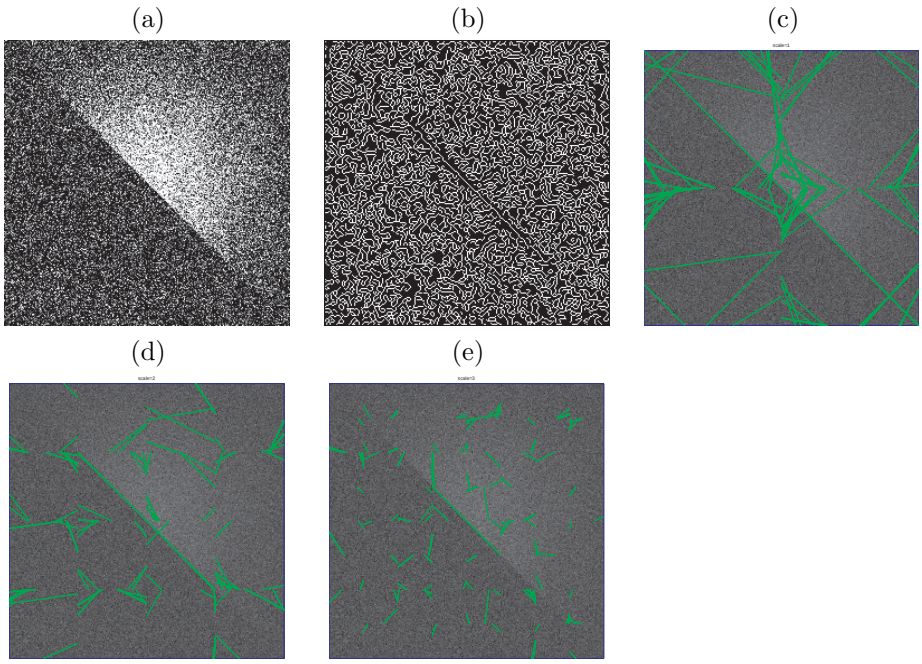

Fig. 7. Example of beamlet-enhanced edge detection. (a) A noisy image with an embedded edge; (b) the result of the canny edge filter; (c)-(e) the locations of significant beamlets at different scales: scales are $1, 2$, and 3 respectively.

5.1.1. *Edge Detection with a Controlled 'False Edge Rate'*

With a better thresholding rule, it might be possible to do even better in isolating the desired signal along the true edge from unwanted false edges.

The reader of "Wavelets in Statistics", the proceedings of the first wavelets and statistics meeting [1], would be aware that False Discovery Rate Thresholding (FDR Thresholding) is a potentially attractive tool for the analysis of noisy multiscale transform coefficients. In particular, FDR provides an objective way to decide on a threshold which will keep false discoveries under control. In the present setting, that would imply the control of the "false edge rate."

The reader can easily adapt the experimental outputs in Figure 7 to produce a beamlet edge detector with False Edge Rate control. More details will appear in future publications of author's.

5.2. *Wedgelet-Based Estimation*

In article [10], the authors briefly described the use of wedgelets in image de-noising. The purpose there is an underlying image U with added noise N, so $I = U + N$ is observed. Supposing the noise N is white Gaussian noise, the articles [9,10] proposed the use of wedgelet-based reconstruction solving

$$\min_R \|I - R\|_2^2 + \lambda \#(R),$$

where R is a wedgelet representation based on Beamlet-Decorated Recursive Dyadic Partitioning, $\#(R)$ is the complexity of the partitition, and $\lambda = \sigma^2 \log(n)$ is used for $n \times n$ images based on some asymptotic analysis. This idea was tried out for some simple blobs in Figure 18 in article [10].

The reader of article [10] might wonder how the procedure performs at various signal-to-noise ratios. It is easy to conduct such an experiment with simple modifications of the code to figure 18 of article [10]. The files to use as a starting point can be found starting with the browser for the article [10]: **BMIABrowser.m**. Looking at files in the same subdirectory, the implementation for figure 18 of article [10] is found to be in **BMIAfig18.m**, which calls **fig18_run.m** and **fig18_dr.m**. Cloning these files and modifying them slightly gives the experimental results shown in Figure 8 of this paper. One runs the wedgelet reconstruction algorithm at 12 noise levels, from very slight $\sigma = 1/10$ to very severe $\sigma = 2.3$.

Many interesting variations can be tried at this point, including variations on the $\lambda = \sigma^2 \log(n)$ threshold rule. Perhaps with a high threshold the clutter in the reconstructions (m) can be cleaned up. The reader of this paper (which is reproducible!) can conduct such experiments using BeamLab (which contains the code for Figure 8 of this paper!).

5.3. *Performance of Blob Detection*

At the 2003 Wavelets and Statistics conference, one of us (DLD) spoke about our joint paper [2] for detection of geometric objects in heavy noise. The theory developed there was asymptotic in nature, and described a threshold at which blobs are just barely detectable. However, asymptotic results are not by themselves convincing. It would be interesting to know if there are computationally-feasible methods for finite-sample situations.

In the article [10], the authors proposed a multiscale blob detector base on beamlets, wedgelets, and linear programming. The method developed there can be used to locate/detect blobs. It might be of interest to run a small simulation study, illustrating the detectability threshold for this method. Such a study can be based on

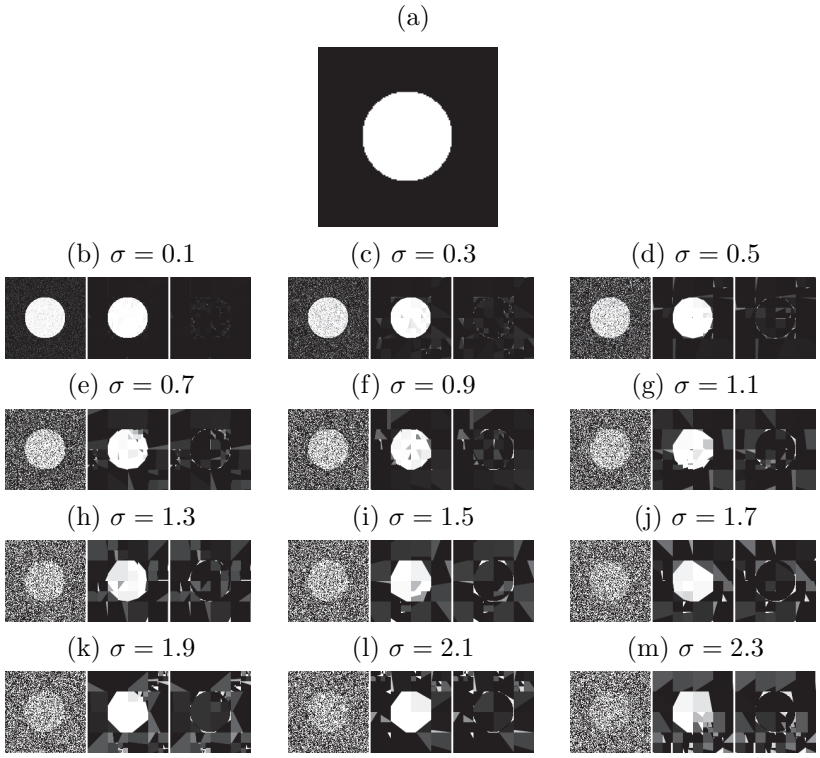

Fig. 8. Performance of Wedgelet Approximation. (a) The underlying noiseless object is a disk; (b)-(m) results of wedgelet approximation for a range of noise levels, σ. In each panel, from left to right, the subpanels show the noisy image, the wedgelet approximation, and the difference between the wedgelet approximation and the noiseless object.

the code underlying Figure 26 in article [10]. This code is located in the BeamLab library inside the directory

Beamlets/Beamlets2D/GraphicAlgorithms/MCTTRC/.

The central engine of the blob finder is linear programming, for which many implementations exist. Our approach has three different implementations, driven by **MCTTRC_pdsco.m**, **MCTTRC_pdsco_rz.m**, and **MCTTRC_LP.m**, respectively. The first two depend on a commercial software package PDSCO [22] produced by Michael Saunder of Stanford University. It is not distributed with BeamLab. The third one uses the LP Solver in Matlab's optimization toolbox, which is also commercialized and not distributed with BeamLab.

Once the user accesses a linear programming solver, it is not hard to run experiments with the blob finder. Figure 9 in this paper gives the result of an experiment evaluating the blob finder's performance at noise levels ranging from moderate to extremely heavy. With $I = U + \sigma N$ as before, U the indicator of a disk and N

a standard Gaussian white noise. Figure 9 shows the results of experiments at σ ranging from 1/2 to 16.

Very strikingly, the blob finder works well even at noise levels so heavy that the eye is unable to verify the presence of anything besides noise.

With more studies, it would be possible to compare the finite-sample detection threshold (in this case somewhere between 8 and 16) against the asymptotic formulas in paper [2].

Such studies would be easy to conduct, because the code for figure 9 of this paper is included in BeamLab, and is easy to modify.

5.3.1. *Role of convexity in MCTTRC?*

Our blob detection empirically favors convex solutions. How does the shape of the underlying object affect the solution of MCTTRC? Note that in the previous example, the object of interest — a simple disk — is convex. We can easily experiment with non-convex objects. Figure 10 presents results for noise levels $\sigma = 1/2, 1, 2, 4,$ and 8, with a nonconvex underlying object. We can summarize our observations as follows.

(1) The results of MCTTRC do favor the convex objects, as the solutions in Figure 10 show.
(2) At $\sigma = 2$, the results of MCTTRC are still reasonably close to the underlying object, while in the noisy image, the embedded object is very hard to observe.
(3) When $\sigma = 4$, the MCTTRC starts to fail.

6. Conclusion

BeamLab is a Matlab toolbox which allows users to reproduce computational results from papers by the authors and their collaborators. With adequate knowledge of Matlab, users can rerun most of the experiments with variations of algorithmic parameters. BeamLab provides a graphical user interface, which gives users an easy way to locate related functions and scripts. For researchers who want to study recent developments in *Multiscale Geometric Analysis*, BeamLab will be valuable.

Acknowledgment

This work was partially supported by NSF DMS 0072661 and 0140698 (FRG, Stanford), and DMS 0140587 (Georgia Tech.), by DARPA, and other sponsors.

Contributors to this software include Amir Averbuch, Tel Aviv University; Emmanuel Candès, California Institute of Technology; Raphy Coifman, Yale University; Michael Saunders, Stanford University; and Jean-Luc Starck, Centre dÉtudes Nucléaires de Saclay (CENS). The first author also thanks the following institutions for their hospitality during the pursuit of this work: Sackler Institute, Tel Aviv University; Mathematics and Computer Science Department, Tel Aviv University; and Mortimer and Raymond Sackler Institute of Advanced Studies at Tel

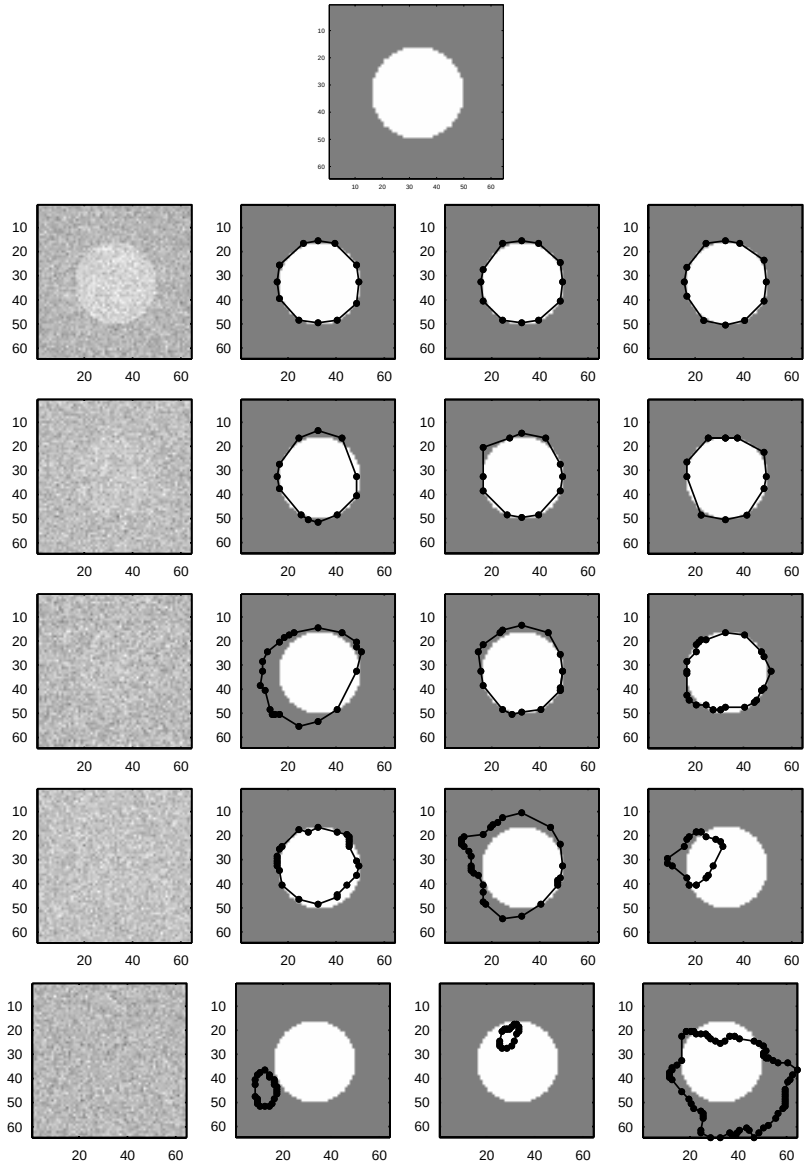

Fig. 9. Performance of MCTTRC blob finder in heavy noise. The top image shows the underlying object. Other rows show results with noisy data $\sigma = 1/2, 2, 4, 8, 16$. Panels show, from left to right, noisy data, and three reconstructed blobs. Three realizations were generated per noise level.

Aviv University. The authors would like to thank our students and postdocs for their contribution in writing, testing, and evaluating the software: Georgina Flesia, Michael Elad, Sou Cheng Choi, Danzhu Shi, and Arne Stoschek.

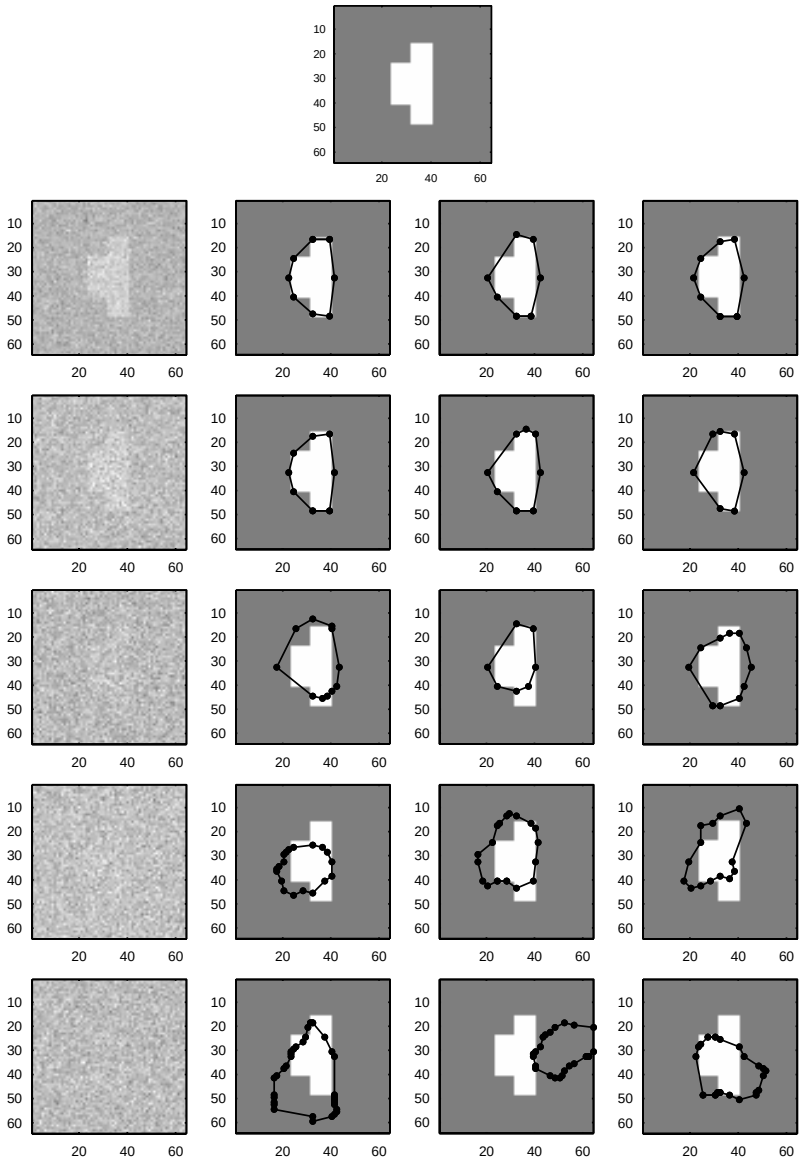

Fig. 10. Performance with non-convex blobs. The top image shows the underlying object. Other rows have a similar meaning to Figure 9. $\sigma = 1/2, 1, 2, 4$, and 8.

References

1. A. Antoniadis and G. Oppenheim (Eds.), *Wavelets and Statistics,* (New York: Springer-Verlag, 1995).
2. E. Arias, D. L. Donoho and X. Huo, *Near-optimal detection of geometric objects by fast multiscale methods*, Submitted manuscript, (2003).

http://www-stat.stanford.edu/∼donoho/Reports/2003/MultiScaleDetect.pdf

3. A. Averbuch, R. Coifman, D. L. Donoho, M. Israeli and J. Walden, *Fast slant stack: a notion of Radon transform for data on a Cartesian grid which is rapidly computable, algebraically exact, geometrically faithful, and invertible*, Technical Report, Department of Statistics, Stanford University, Stanford, CA 94305, May (2001).
4. J. B. Buckheit and D. L. Donoho, *WaveLab architecture*, (1995). http://www-stat.stanford.edu/∼wavelab/WaveLabArch.ps.
5. J. B. Buckheit, S. Chen, D. L. Donoho, I. M. Johnstone, and J.D. Scargle, *WaveLab reference manual*, (1995). http://www-stat.stanford.edu/∼wavelab/WaveLabRef.ps.
6. J. B. Buckheit, and D. L. Donoho, *WaveLab and reproducible research*, In A. Antoniadis and G. Oppenheim (Eds.), *Wavelets and Statistics*:55-81. (New York: Springer-Verlag, 1995).
7. D. L. Donoho and I. M. Johnstone, *Adapting to unknown smoothness via wavelet shrinkage*, *J. Amer. Statist. Assoc.* **90**(432) (1995): 1200–1224.
8. D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard, *Wavelet shrinkage: asymptopia?, J. Roy. Statist. Soc.* **Ser. B 57** (2) (1995) 301–369.
9. D. L. Donoho, *Wedgelets: nearly minimax estimation of edges, Annals of Statistics,* **27** (1999): 859–897.
10. D. L. Donoho and X. Huo, *Beamlets and multiscale image processing,* In *Multiscale and Multiresolution Methods*. Eds T.J. Barth, T. Chan, and R. Haimes, Springer Lecture Notes in Computational Science and Engineering, **20** (2001) 149–196.
11. D. L. Donoho and A. Flesia, *Digital ridgelet transform based on true ridge functions,* in *Beyond Wavelets*, G. V. Welland (Ed.), 1-30. (Academic Press, 2003).
12. A. Flesia, H. Hel-Or, A. Averbuch, E. Candès, R. Coifman, D. L. Donoho, *Digital implementation of ridgelet packets*, in *Beyond Wavelets*, G. V. Welland (Ed.), 31–50 (Academic Press, 2003).
13. D. L. Donoho and O. Levi, *Fast X-Ray and beamlet transforms for three-dimensional data*, in *Modern Signal Processing*, D. Rockmore and D. Healy, (Eds.), *Mathematical Sciences Research Institute Publications*. (Cambridge University Press, 2003).
14. D. L. Donoho and M. R. Duncan, *Digital curvelet transform: strategy, implementation, experiments*, in *Wavelet Applications* **VII,** H. Szu, M. Vetterli, W. Campbell, J. Buss (Eds.), Proc. SPIE, **4056**, (2000) 12-30.
15. J. Claerbout, *Hypertext documents about reproducible research* (1994). http://sep.stanford.edu.
16. Adobe. http://www.adobe.com/.
17. MATLAB web site. http://www.mathworks.com/.
18. S-PLUS web site. http://www.insightful.com/products/splus/.
19. IDL web site. http://www.rsinc.com/idl/.
20. R software web site. http://www.r-project.org/.
21. OCTAVE web site. http://www.octave.org/.
22. PDSCO web site. http://www.stanford.edu/group/SOL/software/pdsco.html.
23. JAVA Technology. http://java.sun.com/.