Using MATLAB to Generate HTML

R. S. SCHESTOWITZ*
IMAGING SCIENCE AND BIOMEDICAL ENGINEERING
UNIVERSITY OF MANCHESTER
UNITED KINGDOM

*roy.schestowitz@isbe.man.ac.uk

About this Document

I decided to share my experience outputting my MATLAB experiments onto public HTML files. Autonomous experimentation becomes very trivial in this way. The great benefits can possibly be understood by viewing the example on the next few slides.

Disclaimer

This document was written in haste. Please report typos and mistakes when found.

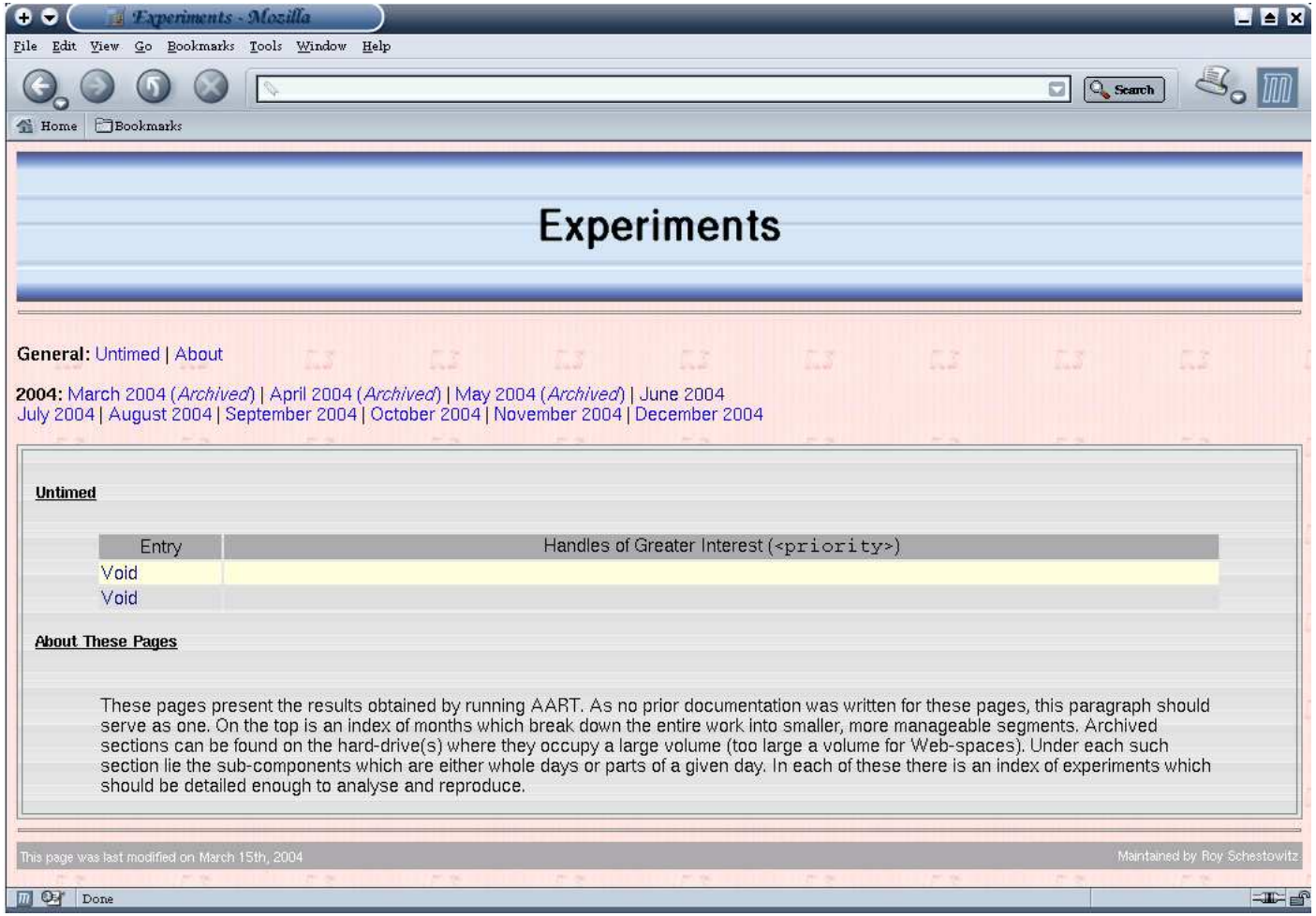
Motivation

- Many experiments performed day by day
- Input variables change
- Output data disappears
- Manual work needed for useful documentation
- Experiment reconstruction is *hard*

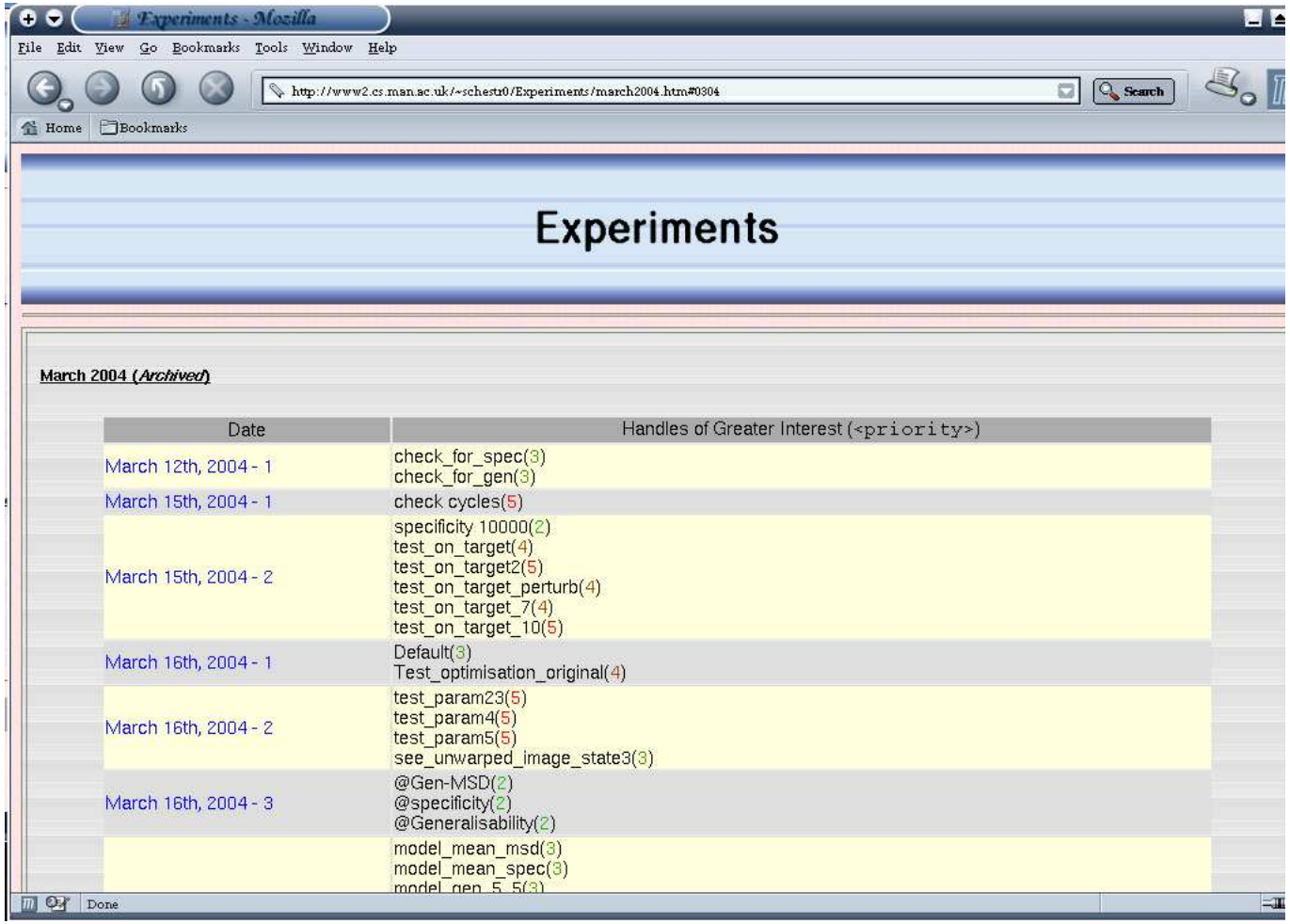
The Solution

1. Save all input values automatically
2. Save all figures automatically
3. Save all videos automatically
4. Record data for context, e.g. date, version, description
5. Collate (1)-(4) in a coherent interlinked manner

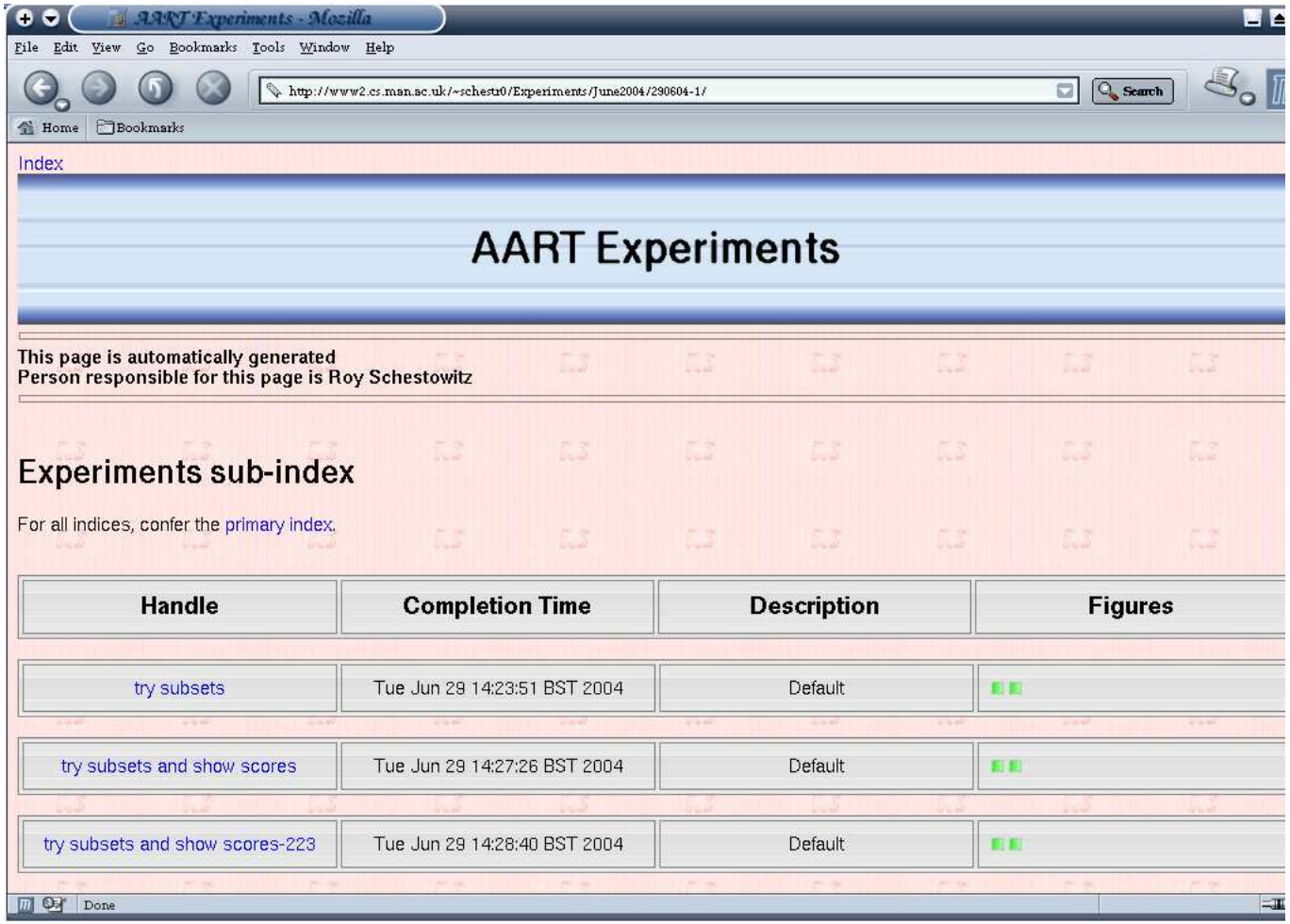
Example - Top Level



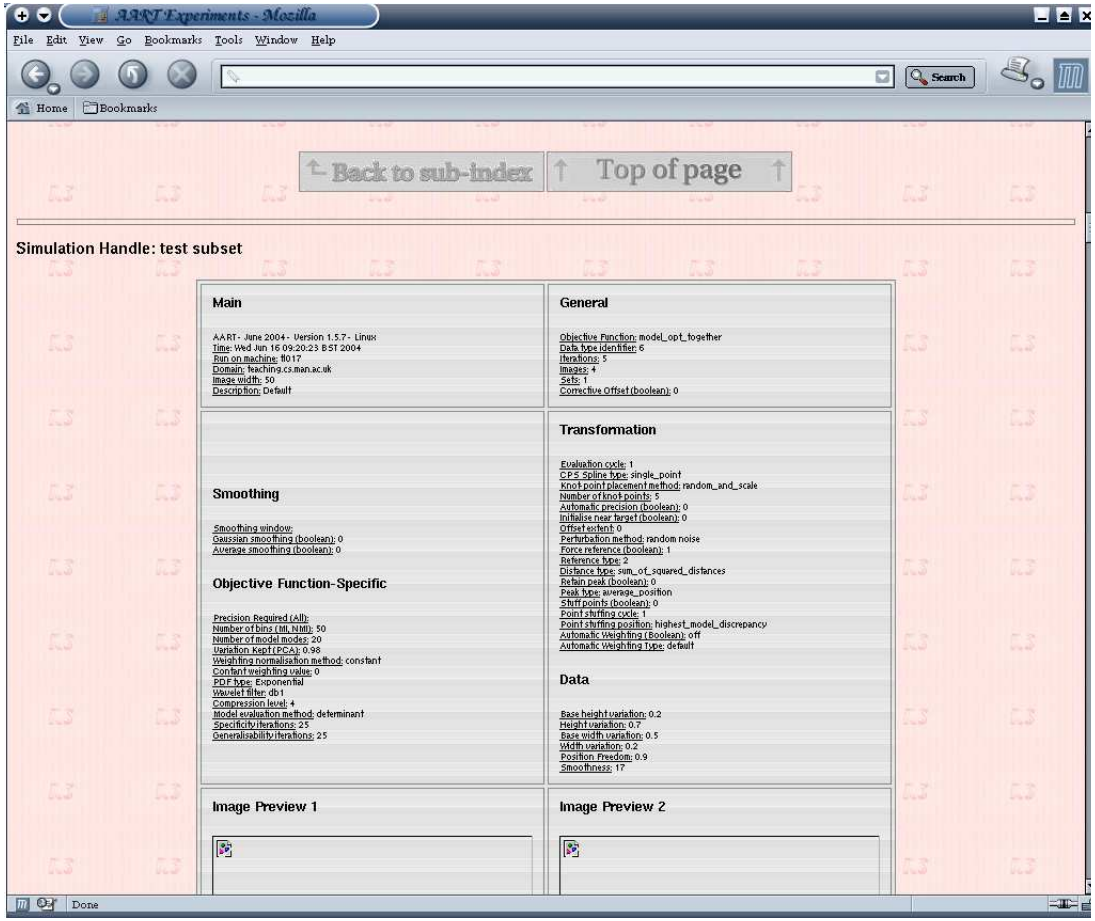
Example - Month Level



Example - Day Level



Example - Lower Level



Initial Implementation Steps

1. Recording Inputs

2. Recording Output (Images/Videos)

3. Linking

Step 1: Recording Inputs

A good starting point would be to print out all outputs to the user, e.g.:

```
disp('|== Inputs ================================');
disp(['| Input String: ', some_string)]);
disp(['| Input Boolean: ', num2str(some_boolean)]);
disp(['| Input Number: ', num2str(some_value)]);
disp('|=========================================');
```

Step 1: Recording Inputs Ctd.

We now wish to do the same, but record the text in a file so that it remains accessible afterwards.

```
fid = fopen('log.htm','a');
fprintf(fid, ['<H2>Inputs</H2>']);
fprintf(fid, ['\n<BR><U>Input String:</U> ', ...
  some_string]);
fprintf(fid, ['\n<BR><U>Input Boolean:</U> ', ...
  num2str(some_boolean)]);
fprintf(fid, ['\n<BR><U>Input Number:</U> ', ...
  num2str(some_value)]);
fclose(fid);
```

Step 2: Recording Outputs

What is important is to keep track of filenames and be systematic about it. We begin by setting a counter.

```
figure_handle_number = 1;
   % A number to be incremented every time a figure is
   % saved. It manages the number of figures that
   % are generated.
```

Step 2: Recording Outputs Ctd.

Whenever a figure is generated, it needs to be saved as follows:

```
figure(current_figure_handle);
plot(data);
saveas(current_figure_handle, ...
[[current_experiment_handle],'-', ...
num2str(figure_handle_number)], 'jpg');
figure_handle_number = figure_handle_number + 1;
```

Step 2: Recording Outputs Ctd.

Q: What about videos?

They can be saved as AVI files in a uniform way and later be linked to.

Q: What about data files?

Just save them and remember the filenames.

```
save([[current_experiment_handle],['.mat']] ...
  , 'data1','data2');
```

Step 3: Linking

Possibly the most important and difficult steps. The current directory now has a collection of files: MATLAB data files, JPEG files, HTML files and AVI files. We need to group them together in a meaningful way.

How can they be grouped? Use HTML. What follows is some sample code.

Low Level linkage - Images

```
fprintf(fid, '\n<TABLE><TR><TD><H2>Image Preview 1...
  </H2><BR><A HREF="');
fprintf(fid, [[handle], '-1.jpg"><IMG HEIGHT=200 ...
  WIDTH=300 BORDER=0 SRC="']);
fprintf(fid, [[handle], '-1.jpg"> ...
  </A></TR></TABLE>']);
```

Low Level linkage - Videos

```
fprintf(fid, '\n<TABLE><TR><TD><A HREF="']);
fprintf(fid, [[handle],'-1.avi">Video #1 ...
  </A></TD></TR></TABLE>']);
```

Low Level linkage - Data and Statistics

```
fprintf(fid, '\n<TABLE><TR><TD><A HREF="');
fprintf(fid, [[handle],'.mat">Input Data ...
</A></TD><TD><A HREF="']);
fprintf(fid, [[handle],'.htm">Statistics ...
</A></TD></TR>']);
fprintf(fid, '\n</TABLE>');
```

Page Headers

Since the file is opened in append mode ('a'), it will record all data sequentially and create a large file without a title. Use the following to add a header only at the start.

```
if (strcmp(which('log.htm'), '')),
  add_headers = 1;
else
  add_headers = 0;
end
fid = fopen('log.htm','a');
if (add_headers == 1),
  add_html_headers(fid);
end
```

Header Example

```
fprintf(fid, ['<HTML><HEAD><TITLE> ...
  My Experiments</TITLE>\n']);
fprintf(fid, ['<link rel="stylesheet" ...
  href="exp.css" type="text/css">\n']);
fprintf(fid, ['</head>\n']);
fprintf(fid, ['<BODY ...
  background="../../bg.gif">\n']);
fprintf(fid, ['<A NAME="top" ...
  HREF="index.htm">Index</A>\n']);
fprintf(fid, ['<H1>My Experiments ...
  </H1>\n']);
fprintf(fid, ['<HR SIZE=5>\n']);
```

A Word on Indexing

At some stage it would be sensible to create an index file dynamically. This way, all experiments are put in a central page which allows a broader scope.

```
fid = fopen('index.htm','a');
fprintf(fid, ['\n<P><CENTER> ...
  <TABLE WIDTH=100%% BORDER=BOX CELLPADDING=10 ...
  CELLSPACING=3 BACKGROUND="bg.gif"> ...
  <TD ALIGN=CENTER WIDTH=25%%><A ...
  HREF="log.htm#', [handle], '">', ...
  [handle], '</A></TD><TD ALIGN=CENTER ...
  WIDTH=25%%>']);
```

```
fprintf(fid, ['\n', [description], ...
  '</TD><TD 25%%> ']);
fclose(fid);
```

Final Word

- Start with simple text file outputs.
- Build up towards HTML formatted files.
- Link to external outputs, e.g. pictures.
- Create index to all indices to progressively build a useful hierarchy.