DIPOLE LATTICE

ATUL SINGH ARORA

Upscaling a nano-structure

Dr. Ravi Mehrotra
National Physical Laboratory, New Delhi

May-July, 2013

Atul Singh Arora: *Dipole Lattice,* Upscaling a nano-structure,

Every honest researcher I know admits he's just a professional amateur. He's doing whatever he's doing for the first time. That makes him an amateur. He has sense enough to know that he's going to have a lot of trouble, so that makes him a professional.

— Charles F. Kettering (1876-1958) (Holder of 186 patents)

ACKNOWLEDGEMENTS

I thank Dr. Ravi Mehrotra for, well conceiving the experiment and guiding me through the process of its realization.

CONTENTS

1 This section can be skipped, without any loss of continuity.
2 This section onwards, its assumed that you have 'temperature' programmed and the dipoles setup ready

LIST OF FIGURES

Part I

CHAPTERS

1 PROLOGUE

Atoms and molecules are far too small to be observable as individual entities, with our eyes alone. Scientists have come a long way at understanding *their* world. It has been attempted to recreate a specific micro-structure, at a scale where we can directly observe it.

The configuration we've studied here, is that of a Magnetic Dipole Lattice, viz. Magnetic Dipoles that can only rotate about their axis, placed on a grid. Their physics by itself is rather interesting and can be simulated to observe the dynamics. The experiment is expected to show the same dynamics, that of the microscopic world, only directly observable.

1.1 PRIOR ART

TODO: Complete this part after understanding the physics and simulations on the system.

1.2 EXPERIMENTAL SETUP

The upscale version consists of Physical Magnetic Dipoles, that rest on near zero friction spots on a grid. A camera sits on top, with all the dipoles in its field of view. The Lattice Analyser takes the input from the camera and simulates the given temperature through a hardware unit and the coils attached to each dipole. It is that simple.

For implementation details, you may read the following sections.

1.2.1 *The Dipole*

According to the current design (as of May 19, 2013), the Magnetic Dipole is built off of two small cylindrical rare earth magnets, attached to a needle, with their flat face's surface normal perpendicular to the axis of the needle. The needle rests in an assembly with a glass slide at the bottom. This keeps it upright and nearly free of friction. Each dipole further has a circular disc on top, with its centre passing through that of the dipole. The disc has a pattern printed, designed to find its angular position using a camera. Further, the dipole assembly also has two coils along an axis perpendicular to the needle.

1.2.2 *Lattice Analyser*

This is the application that

1. records the dynamics of the system
2. calculates the required field strength of each electromagnet

using a webcam and computer vision techniques. The results of the latter part depend on the temperature that is to be simulated; temperature is not maintained by providing heat, but instead by providing a certain distribution of speeds to the dipoles.

1.2.3 *Temperature*

This is the hardware unit, (will be built around an ATmega 16) that provides the coils with the current as calculated by the Lattice Analyser (using a USB interface).

2 WATCH IT GROW

2.1 SENTIMENTAL INTRODUCTION[1]

Science often seems like a blackbox that relates observables. Even more often, it is rather convenient to lose touch of observables altogether, and wander in the blackbox. Performing an experiment, gets one closer to nature, to the roots of the subject.

2.2 THE JOURNEY

2.2.1 *Look it has begun*

This experiment wasn't started from scratch. My guide, Dr. Ravi Mehrotra, had already worked with a team and created the Dipoles as described earlier. The team had also worked on the image detection algorithms, but their work wasn't usable.

There were three tasks at hand, of which one had been significantly simplified by the prior work.

1. The Dipole

   This had one apparent problem; the dipoles had to be made virtually frictionless (which is not to say they had excessive friction, infact they would oscillate atleast about 8 times before stopping aligned with earth's magnetic field)

2. The Image Analysis

   This part I had to start from the beginning with two basic objectives, as stated earlier; measuring the angle of the dipoles and evaluating the current to be pumped based on the temperature selected.
   What was known soon, was that C++ will be used for programming and linux would be the operating system, to facilitate USB interface with the AVR (next step)

3. The Current Control Hardware

   This is simply for providing a current pulse proportional to the intensity calculated by the lattice analyser. Some schematics for this were available, but were found to be inaccurate and incomplete.

1 this section can be skipped, without any loss of continuity.

2.2.2 *Time Line*

Section A.1 is the event log, which has the progress as and when it was made.

2.2.3 *Construction of the Dipole*

To remove the friction, there were various ideas, including use of a super conductor. However, eventually three methods were considered and experimentally tested.

1. Ferro-Fluid:

   As it turns out, there are substances that have a ferro magnetic properties but in the liquid form. Consequently, a strong enough magnet would glide if coated with this substance.

   Experimentally, it was found that the friction was higher than the 'needle on glass' setup.

2. Magnetic Levitation:

   A magnet can easily suspend another magnet, granted it doesn't flip. This idea was used and a magnetic cylinder was placed coaxial to the needle, using a cylindrical eraser and glue. Beneath the glass slide, an identical magnet was placed with the face that repels upwards.

   Experimentally, again it was found that the motion was more damped than the 'needle on glass' setup. The reason for this case was obvious after a little analysis and closer observation. The dipole would align to the field of the magnet, viz. the magnetic field was interfering with the dipole.

3. Air Levitation:

   To test this, the very first requirement was a source of stream of air. For this, we started small. We arranged for a small USB fan from a colleague. The next task was to channel the flow of air. This was accomplished by attaching the front part of a Pepsi Bottle such that the larger diameter was closer to the fan and the mouth of the bottle had the chord stuck to it (could still be moved if required), as diagrammatically given in Figure 1. The final setup has been given in Figure 2.

   This failed miserably for the air pressure would fall the moment the assembly covered the fan. Introducing slits to allow air to flow in created no appreciable difference. This idea had to be dropped in favour of the vacuum cleaner setup as shown in Figure 3

   The vacuum cleaner was used as a blower and connected to a box using a pipe. On one of the faces of the box, four holes

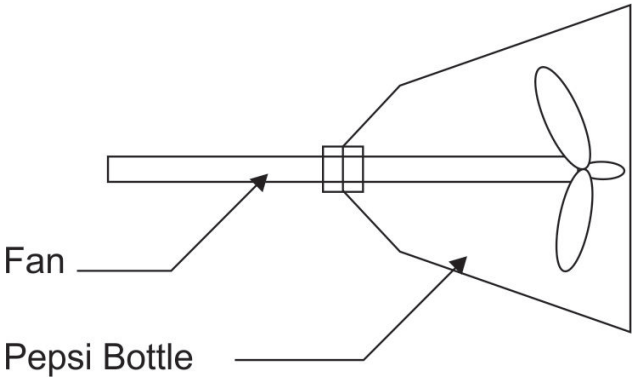

Figure 1: Fan Setup

Figure 2: Final Fan Setup

were drilled (which were later enlarged). These were covered to increase the pressure when required. On the open hole, one dipole was placed (which had to be re built with a minor modification, refer to Figure 4) with a disc at the bottom. This is when an apparently bizarre observation was made. At a given pressure, it was found that the dipole could remain suspended in air beyond a certain height (and obviously there was an upper bound for the same). However, for heights lower than that, the dipole would fall. The explanation which seemed to resolve this was that the air could spread while rising sufficiently only beyond that height to apply pressure at a large enough surface area, thus create enough force. Albeit the experiment was not performed under precisely the same conditions, when it was repeated with a larger disc, the same problem was encountered, suggesting that there may be more to the explanation that deduced so far. Other geometries at the base (other than the disc) were also tried, such as a cone and a thermecol sphere, nei-

Figure 3: Vacuum Cleaner Setup

ther of which worked at low pressures which were enough to suspend the disc based setups. Further, at high pressures, disturbances in the form of torque in the dipole's axis of rotation begin to appear, which are fatal for the experiment.

Other problems with air-suspension included damping in the vertical direction. Once the dipole reaches the vertical equilibrium point, it overshoots just as an oscillator. Since the friction at the plates holding the needle vertical is negligible, the oscillations don't get damped quick enough. This was experimentally observed also. The energy ratio between the rotational part and the translation part, in the earth's field, was calculated and found to be approximately one for the given setup.

The most stable we could achieve with this setup was using a cylindrical projection from which the high pressure air escapes and a disc of comparable size attached to the dipole. This did get suspended satisfactorily, unlike the other methods where the suspension could not be maintained at the desired height, however the needle became rather wobbly and unstable resulting in increased friction.

Online research indicated that air-bearings do function well enough and so do the boards of air hockey. These will be explored in the coming weeks to improve upon the methods to achieve the desired results.

THE DIPOLES

Air hockey table method, DIY air hockey was a better method of searching, got cues
aluminium disk's introduction (For air suspension), after finalizing one, and testing it, we found comparable resistance with the inverted needle top on glass slide as opposed to when air

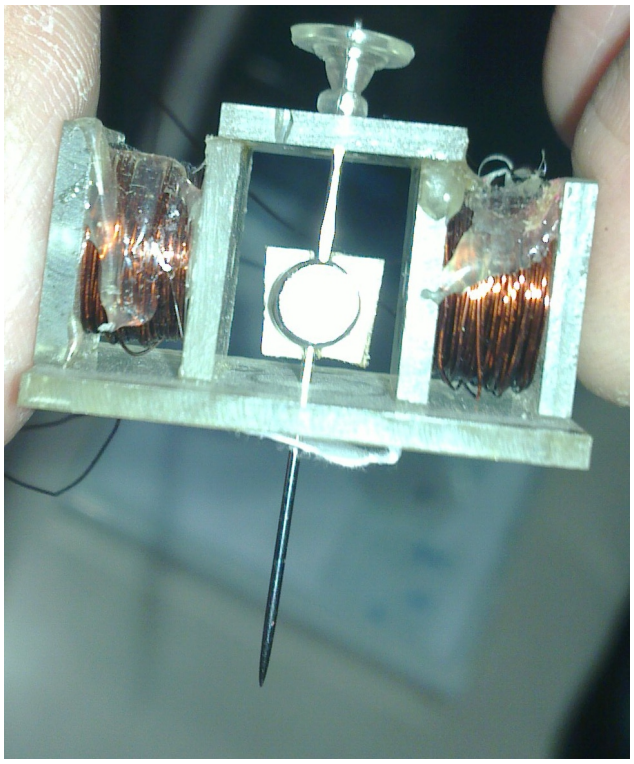

Figure 4: The Modified Dipole

suspended. Hypothesised that the main source of friction was infact the guiding edges, which provide the opposing torque.
an interesting setup was devised, with only the needle edges held using a glass slide and magnets. resist friction etc.
froze the design with aluminium disks at the bottom and made four good dipoles for testing in a 2x2 lattice

2.2.4 *Construction of the Lattice Analyser*

2.2.4.1 *Proof of Concept: Recognizing Patterns*

The lattice analyser has come a long way. Image detection trials were initiated with Figure 5.

The idea was that once the ellipses have been detected, and they are different in colour, one can evaluate from their centroids, the position and the angle of the dipole. It must be stated that earlier it was attempted to use the greyscale image as was provided. However soon the shadow interference led to using coloured patterns instead. These patterns were not printed but displayed on a screen and the camera aimed appropriately.

So first, the algorithm for detection of relevant part of the image had to be frozen. There were two candidates for this

1. Hough Transform Method

Figure 5: Sample Image

Either one could use the already available in OpenCV, line detection or circle detection, both would've required changing the pattern on the dipole

Or one could use an ellipse modification for the same, which would require programming the algorithm.

2. Contour Detection and Ellipse Fitting

   This method detects contours in a given image, and the OpenCV example also shows ellipse fitting for the same. This seemed promising too, but it seemed more expensive (computationally) than looking for predetermined shapes.

This work had been done within the first few days.

Next, a colour filter was to setup to improve the accuracy. When the algorithms were implemented, it was found that the Hough Transform method often misses detection of circles, refer to Figure 6 (this is ofcourse after attaching a video stream instead of images to the code) as compared to contour detection Figure 7.

After the detection, according the plan, two colours were to be used for the ellipses. However, running the hough transform twice would've dropped the detection speed to half, which wasn't worth it. It was then decided that the shapes should be made different instead of relying on two colours for the same information. After looking at various combinations, Figure 8 was finalized, with an ellipse at the centre, and a circle along the minor axis for breaking the symmetry. This method did infact work as shown in Figure 9.

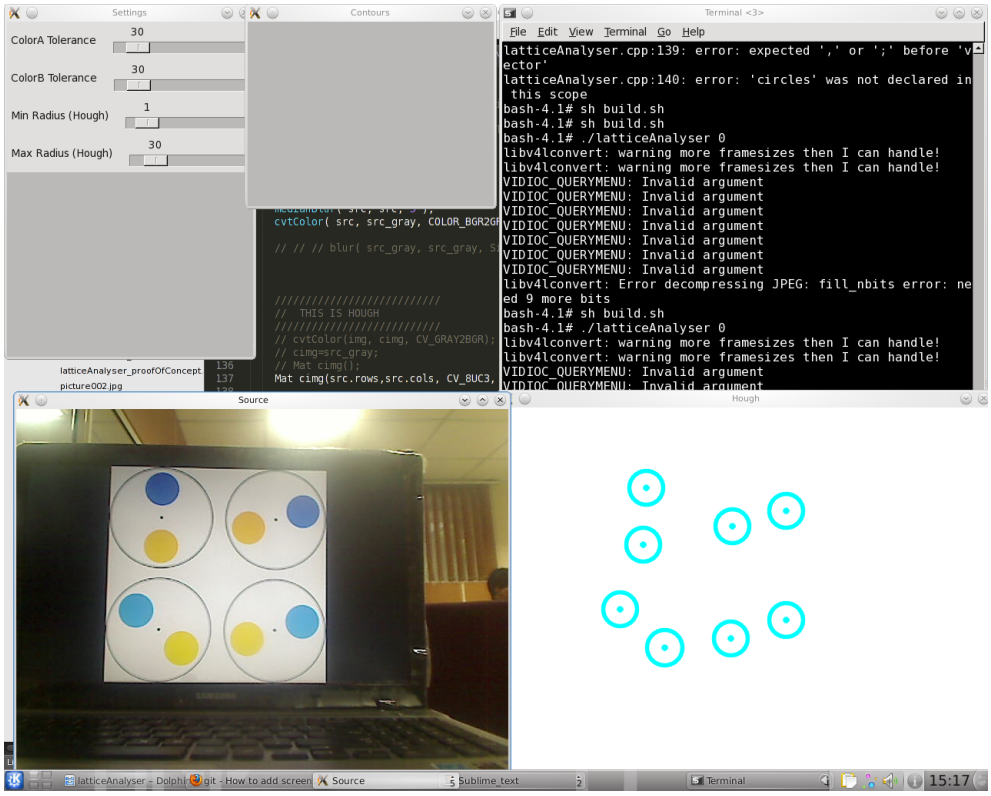

Figure 6: Hough Transform

2.2.4.2 *Aftermath and Recognition Tests*

The next challenge was to realize that a dipole detection can be missed and therefore mess up the counting, if that is the only way of uniquely identifying them. Unique identification is obviously required, as the external hardware must fire the coils of the right dipole. Thus a reference frame was used to uniquely identify the dipoles initially. This is expected to happen when they are stationary to get a good reading. In each frame, whenever a dipole is detected, it is associated with the dipole in the reference frame, by matching its location. If a dipole is not detected in a given frame, the software knows it was unable to record it and doesn't mess up neither the numbering nor the observations.

After implementation of the last part, an animation sequence was created in Power Point, with the dipoles rotating with a constant speed and the camera was aimed at the screen. A still from the same is given in Figure 11. Figure 10, shows the angular position versus time plot, for the first dipole and yes, it is linear, just as expected. Standard deviation tests are still to be done.

Further tests, relating to it's timing were done and it was found that the processing itself was taking longer than the time taken by

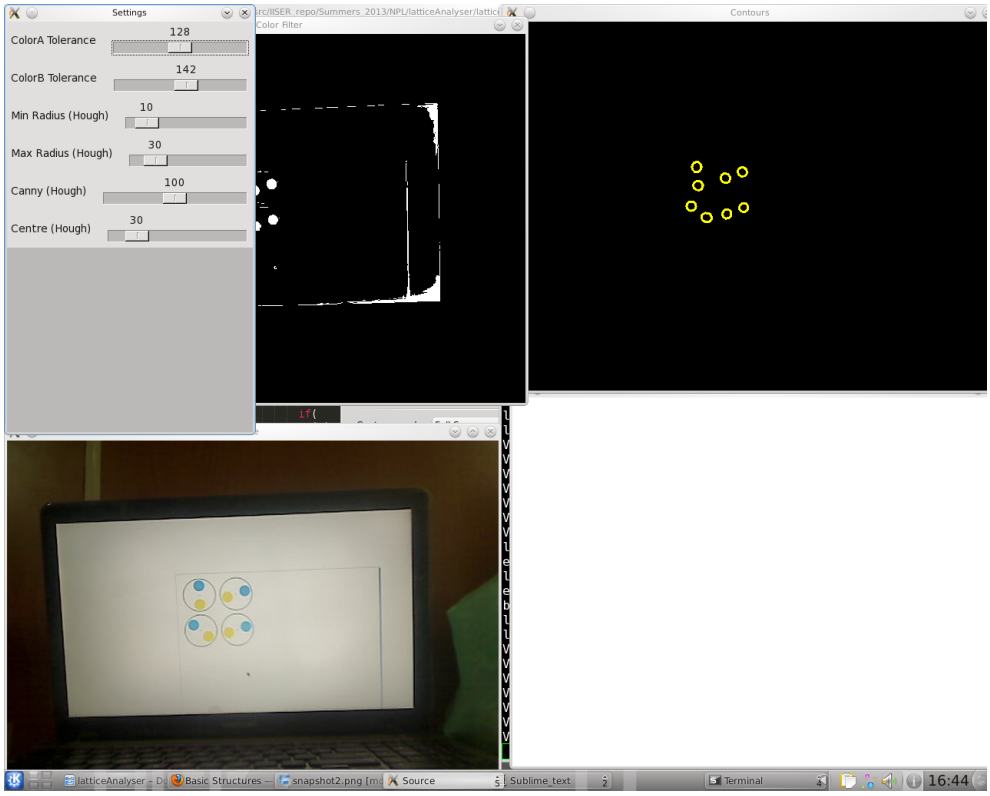

Figure 7: Contour Detection

one frame in a 30 FPS video stream. This could be improved with the following

1. Intel Performance Primitives (IPP)

   These are libraries that provide optimized algorithms for performing some of the basic tasks in OpenCV to speed up the overall computation

2. Multi Threading

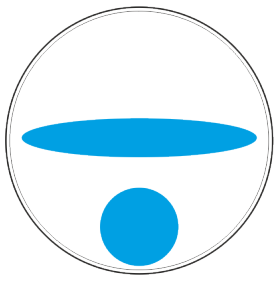

Figure 8: Final Pattern

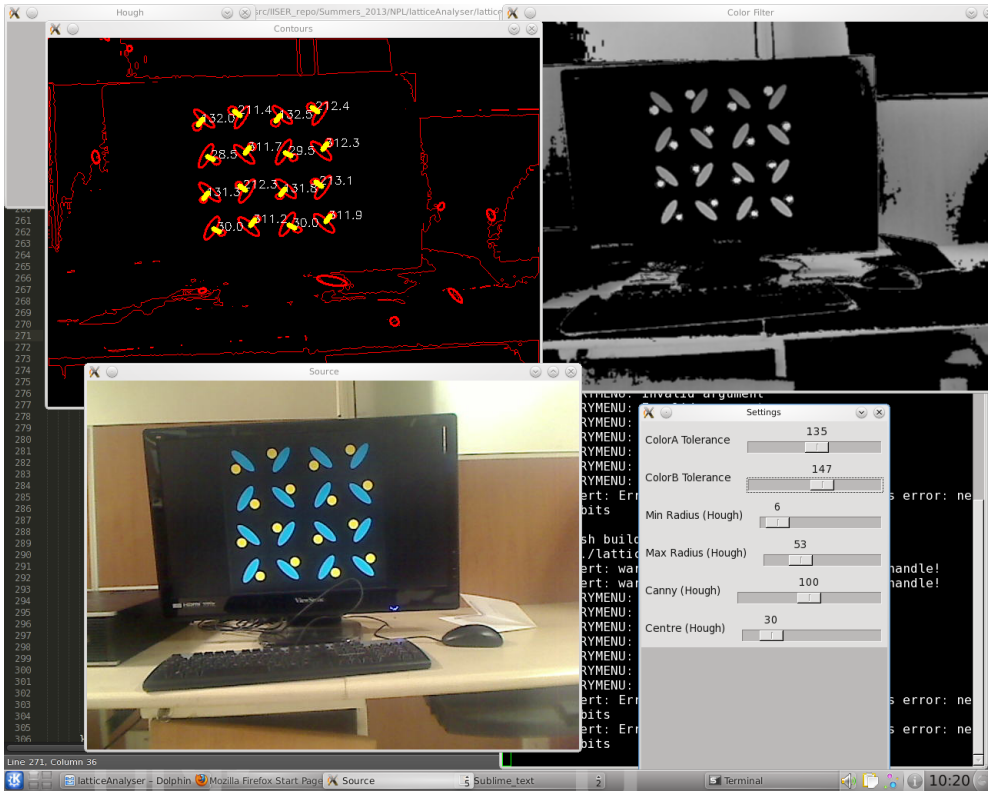

Figure 9: Multi Shape, Single Colour

Rendering the frames in a different thread could perhaps increase the render speed or atleast ensure it doesn't affect the rate of processing of the image

3. Camera Initial Delay

   Despite a 30 FPS smooth video, there seemed to be an initial delay which persisted in the video preview of the camera. This can be corrected for by polling the camera quickly instead of processing the image each time.

4. OpenTBB

   To enable multi-threading within OpenCV to speed up the algorithms

5. Hardware

   The results obtained earlier were for a Pentium 4 HT system. A faster multi-core processor could produce potentially, better benchmarks.

Painstakingly, IPP was downloaded, built and integrated with OpenCV and re-built. There were various issues, starting from an incompatible version of IPP, to faulty documentation in OpenCV. The results did

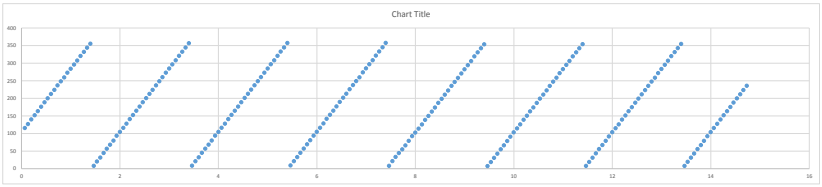

Figure 10: First Observation

not improve however despite this. Perhaps the algorithms used do not rely on the methods optimized by IPP

Multi-threading was implemented using C++ 11. Various synchronization methods were looked up, including mutexes, uniqe locks; eventually atoms were used and tested with Visual Studio Express 2012 on windows (implicitly implying the application was first made to run on windows), as the linux machine had an older kernel and upgrading GCC wasn't recommended.

The camera's initial delay is caused by the initial stream. To rectify this, OpenCV has two methods, one for grabbing a frame, and the other for decoding it. If in a separate loop, the camera is polled regularly for frames, the delay is minimized. The frame can be decoded as soon as processing of the previous frame is over. This, as is suggestive, also requires multi-threading

OpenTBB is a library that is required to enable multi-threading support in OpenCV. This too had to be fiddled with for a while, before being built successfully.

The hardware was changed to an i7 machine which has more then enough cores and computation power.

These modifications were all combined and the code compiled using GCC (except certain parts of multi-threading, due to a compiler issue) and it was found to be able to process in about 25 milliseconds for an 8×8 matrix of dipoles. For a 16×16 matrix, the resolution of the camera (Logitech Pro 9000, 640×480 at 30 FPS) was found to be

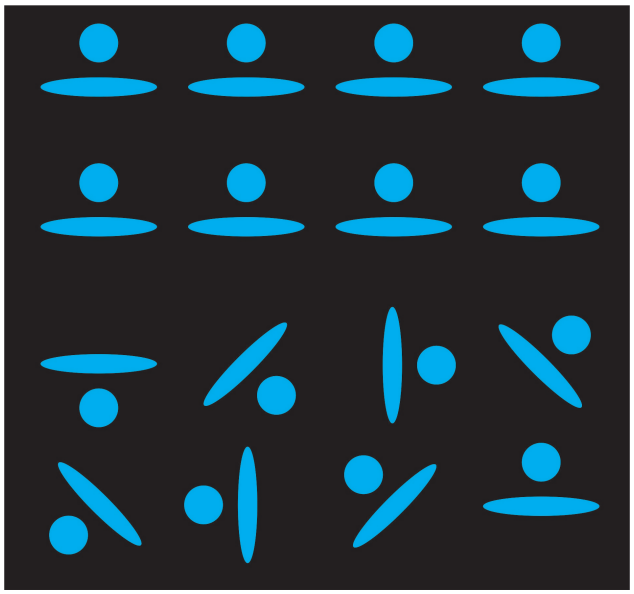

Figure 11: Final Test Pattern

insufficient. Higher resolutions in the same camera did not have a 30 FPS temporal resolution. The alternative seems to be a camera for the Rasberry Pi which can be used over ethernet.

Further, the CLI of the program was modified to add support for inputting various commands, which would be required for testing the hardware (which is scheduled to be built this week).

2.2.4.3 *'temperature' compatible*

And surprise surprise, the hardware interface was in fact done in the following week. Dr. Ravi Mehrotra had already created a simplified version of the Ob-Dev firmware for USB interface of an Atmega with the PC which was used. More on that is there in the temperature section (which follows). The Lattice Analyser had to pump in energy to the system by providing angular velocity to the system of dipoles repetitively at suitable time intervals, viz. when the temperature of the system drops. The temperature is calculated in realtime using the RMS angular velocity of each dipole in the system. It was immediately realized after a closer look that since the time window between two frames is roughly 30 ms, it is not possible to energize all the dipoles with velocities picked from a given distribution of temperature. Further, since the motion of the entire system is coupled, it suffices to energize *some* dipoles sufficiently to maintain the temperature.

To achieve this the following were done (or algorithms for achieving them written) [2]

2 For completeness sake, it must be stated that some modifications had to be made before the colour filtration became functional

1. The tools for compiling and programming the AVR installed
2. Bootloader programmed into the micro-controller along with the sample program given by Dr. Ravi Mehrotra
3. Communication with the PC was tested using the corresponding C example program for the PC
4. USB libraries were incorporated into the latticeAnalyser and linked (had to compile C objects with C++ objects)
5. A simple IO protocol between the Lattice Analyser and the microcontroller was written and tested.
6. The coil of a dipole was tested with a set of 4 A4 batteries and some basic current calculations were done
7. The hardware was setup to include a small current limiting resistor (about 120 ohms) and the coil powered using the Lattice Analyser
8. (was not required) Worked on sachetIO; a library for breaking down a large chunk of data into sachets intended to be transmitted and combined after being received.
9. (was not required) Looked up methods to amplify the current, since the micro-controller's current wasn't found to be strong enough to align the needle of the dipole to it.
10. Discussion about how to implement the energy pumping (which is how the previous two steps were found to be pointless)
11. A 2×2 lattice of dipoles was setup.
12. The angular velocities of each dipole calculated
13. With each frame, the RMS angular velocity calculated
14. An algorithm for position and velocity based energy pumping written
15. Interface added to test the same by supporting inversion of force direction (so that it results in halting) and an option to make blind (viz. disable the algorithm and instead send pulses periodically in time)

The final result was that energy could be pumped into one dipole satisfactorily. The next step was to extend the algorithm to an arbitrarily sized matrix. Certain steps such as velocity calculations when dipole detection fails (at high speeds) has arbitrary inaccurate behaviour in the said version. Algorithms for finding the axis of the coil in the dipoles also needs to be finalized, for there are more than one ways

of doing this, viz. powering the coils and reading the angular position of the dipole, finding the axis of the lattice by noting the Cartesian positions of the dipole, deduce it from the rest (initial) angular position of the dipoles which should be known given the lattice size, etc.

2.2.4.4 *Adding Realtime Graphing Capability*

Before working further on the aforesaid steps, realtime graphing was implemented. This was thought to be required for *seeing* what was happening, specially the angle detection and when it goes wrong. This is being emphasized because the angle detection would often result in incorrect angle calculation which was a serious issue.

After searching the internet for plotting methods, plplot was finalized as the graphing library to be used with C++. This was downloaded, built and installed. I wouldn't go into the implementation details, but would remark that it takes a little time to get everything up and running. The library supports various kinds of graphs. The two types of that were used for the project are

1. Realtime 3D plot
2. Strip chart

Further, the library provides multiple rendering options, of which for linux, a driver for the x11 graphics system was used for testing. In a single window all three graphs were created;

1. 3D: Angular Position vs Dipole ID vs Time
2. 3D: Angular Velocity vs Dipole ID vs Time
3. Stripchart: Average Square Angular Velocity (over dipoles) vs Time

Of course, these graphs weren't all added at once; Figure 12, Figure 13 and Figure 14 tell the story. For the stripchart it must be remarked that it auto scales and shifts as the data flows in. This proved to be more useful than expected for debugging the mod problem (helped visually see the co-relation between other parameters and occurrence of the error). Further it was also useful in identifying periodic fluctuations (with a somewhat constant time period) in the third graph.

This will probably be the final user interface improvement for the Lattice Analyser.

2.2.4.5 *The Modding Problem*

Consider an ellipse. To describe all its orientations uniquely, given it's position, we need a convention for defining the angle and the angle should span pi radians. Only π radians are required as opposed to 2π

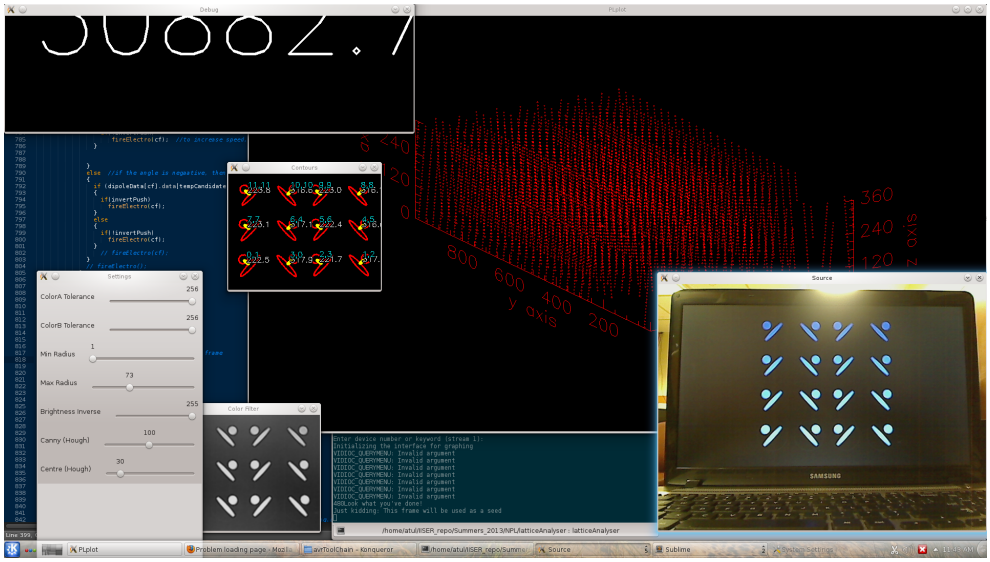

Figure 12: With the first graph

simply because of the symmetry of the ellipse. TODO: Put an image of an ellipse with the angle. We therefore need more information than the angle of the ellipse, to determine the angle of the dipole. The design of the pattern takes care of it by providing a circle to break the symmetry. Simplistically, the problem is now solved, for the circle's position can easily be used to decide when to add π to the ellipse angle and you have the dipole's angle. TODO: Add an image showing the ellipse with a circle and the dipole angle changed accordingly.

Upon implementing this simple logic as an algorithm and testing, it was found that it runs into troubles very quickly and periodically. To understand, first assume that the ellipse reads zero degrees when horizontal and thus, the circle, if it is above the

atan2 approximation for correcting the angle problem, atan2 assisted angle approximation, modular arithmetic based approach for correcting the dipole angle detection (figured this from a simpler problem of firing the elctromagnet)
mod problem for deciding when to fire the electro magnet, used modular arithmetic
RGB based colour detection to HSV based colour filtering, to improve high speed performance | various problems such as conversion of rgb to hsv and so on..
misc: cmake was fiddled around with, auto numbering of lattice points, writing all the data into a file, gstreamer plugins for reading x264
graph for kinetic energy added, there was a periodic shoot up, caused by modding problem
testing source of systematic error, eliminating a cause, camera defect | calibration. list the two kinds of defects

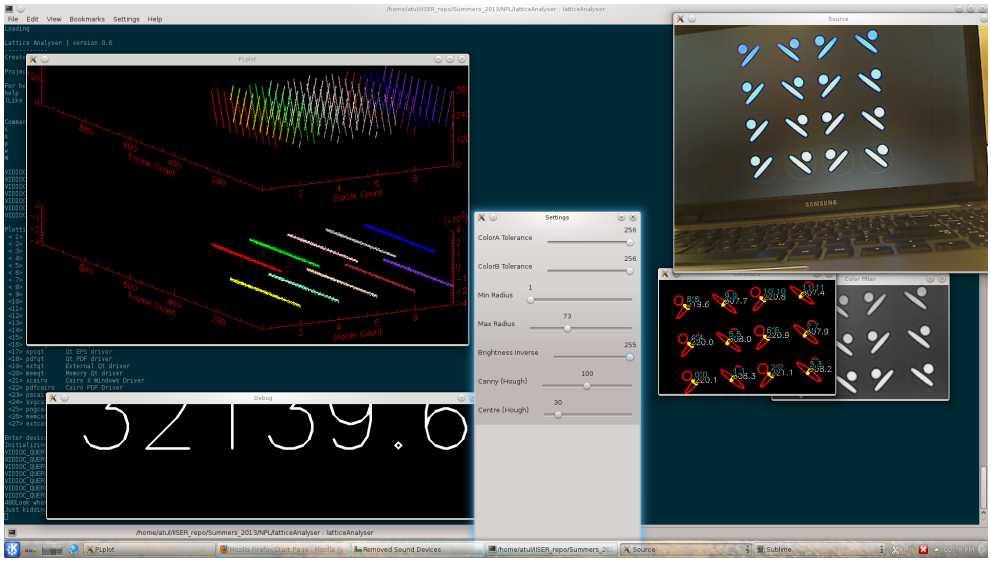

Figure 13: With the second graph

PHYSICS: error analysis: used a lot of GNUplot, to test the kind of deviations for a given dipole, in the angle determination when static, when moving. Then for a given point of time, the deviation in the angles of various dipoles. damping could be tested with realtime grpahs

The source code of the same is given in Section B.1, which has been made available online.

2.2.5 *temperature; the rise*

'temperature' was built around the Atmega8 processor to start with. It's task was to energize the coils of the dipoles, when instructed by the Lattice Analyser. It's source code was written in C, compiled with the free AVR-GCC compiler using some other sister tools of it. Atmega's interface to the computer was built from Dr. Ravi Mehrotra's library, which was built to simplify the existing ob-dev's virtual USB libraries. The coils are powered directly using the chip for the 2×2 setup. For larger matrices, a multiplexer will be required and built suitably.

The current that was roughly enough to align the dipole to the coil was (after experiments with a 6 Volt battery and resistance of coil measurement, viz. 7 ohms) found to be about 1 ampere. The microcontroller can sync at most 20 mA of current at a given pin. Further for the USB interface, the system was set to run at 3.3 V (this can be pushed up to 5V also, but that would've required redesigning the circuit) and the current limiting resistor was correspondingly chosen to be about 150 ohms. This was put in series with the coil and tested. It was found that the dipole wouldn't, for even a second long pulse,

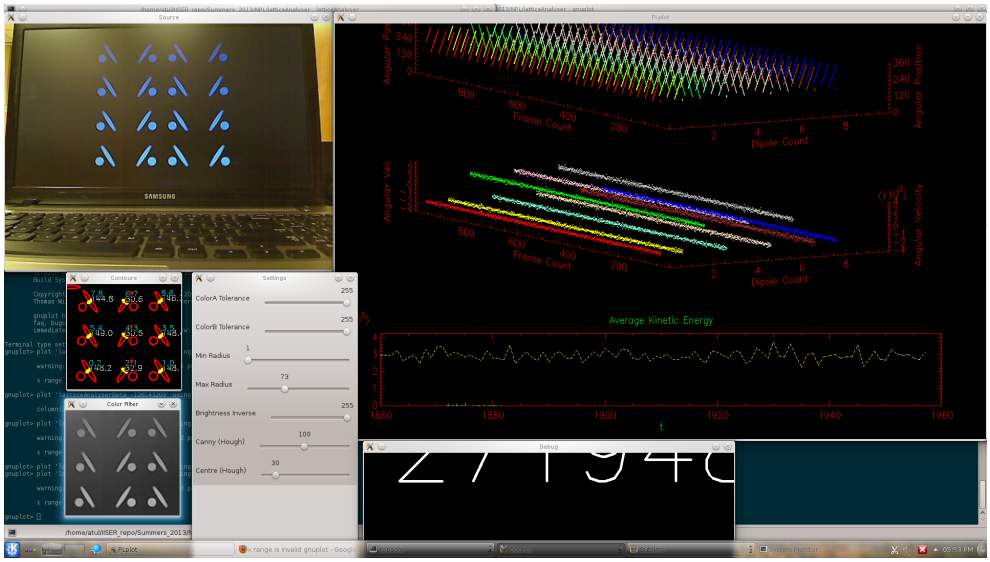

Figure 14: All the three graphs

align itself to the coil, starting from some initial angle. However, when further experiments were done for pumping in energy using the algorithms developed for the Lattice Analyser, this push was found to be enough, even with a few milliseconds long pulse. This version of temperature has been given in Figure 15

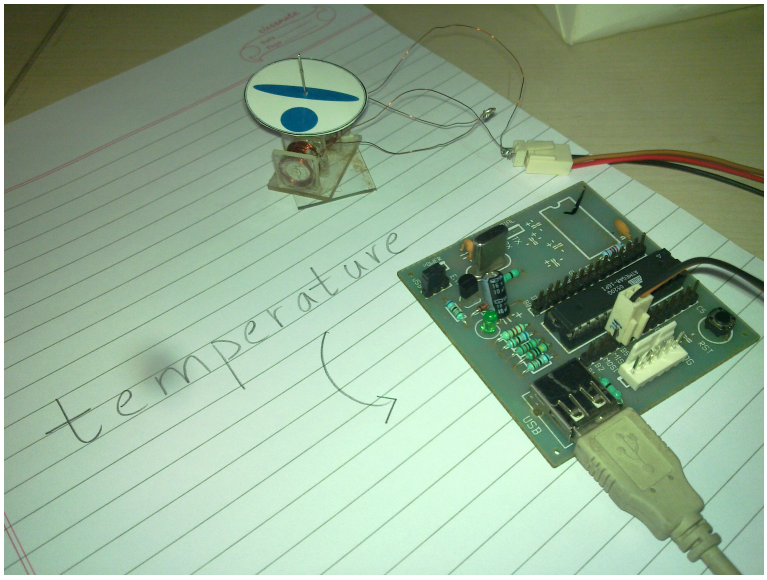

Figure 15: First Version of Temperature

The main file for the temperature has been listed in Section B.2.

3

HOW TO INSTALL AND USE

3.1 WHAT WILL BE COVERED

Here I'll describe how to install the latticeAnalyser and temperature and run the setup.

3.2 WHAT YOU'LL NEED

Lets itemize straight away:

1. A computer running Linux (tested on slackware and ubuntu)
2. A compatible webacm
3. An AVR programmer (one time programming)
4. Tools to fabricate a PCB to design
5. At least 4 dipoles

3.3 LATTICE ANALYSER

3.3.1 *Prerequisites*

You'll first off, need to install a bunch of software packages on your system. Here's a minimalistic list.

1. GCC (preferably a version that supports C++ 11)
2. cmake
3. make
4. git (recommended)
5. OpenCV
6. plplot
7. libusb

Let's start with slackware; most of the build utilities are already available, viz. you wouldn't need to care too much about installing GCC cmake and make. I encourage you to install git. Just download the tar.gz and then

```
cd /path/whereDownloaded
tar -xvf gitPackageName.tar.gz
cd gitPackageName
./configure #(maybe required)
make
make install
```

Now for OpenCV, do the following on a terminal

```
cd /path/where/you/have/space
git clone https://github.com/Itseez/opencv.git
cd opencv
mkdir release
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/
    local ..
cd release
make
make install
```

This should work. If you get errors, you may lookup the internet or drop me an email. If you wish, you can even install it with IPP and TBB (they help in boosting performace)
Next is plplot. You simply need to download it and follow the instructions given in the README after extracting with tar -xvf (oh the xvf is extract, verbose (tells you what its doing) and file (you specify it)). This too uses cmake.
The usb library I don't remember installing so it is I think already available in slackware. Else you may download it.

Now in Ubunutu, you just need to get 'synaptic' from the Software Centre and then download the 'dev' versions of all the software packages listed above. Simple.

3.3.2 *Compiling*

Now do the following in a terminal

```
cd /path/where/you/want/to/put/latticeAnalyser
git clone https://github.com/toAtulArora/dipoles.git
cd dipoles/latticeAnalyser
make clean
make
```

And you're good to go.

3.4 TEMPERATURE

3.4.1 *Prerequisites*

You'll need a programmer for the AVR and I am assuming you have some experience using it. Also I'm assuming its compatible with avr-

dude, at least for this set of instructions. So here's the list of packages you'll need.

1. bin-utils
2. gcc-core
3. avr-libc
4. avr-dude
5. bootloadHID

For slackware, you'll need to work a little. In ubuntu, you can install all (except the last) using synaptic. To manually install, download all these from the net into one folder say avr-stuff and fire up a terminal. Run the following.

```
cd /where/you/downloaded/avr-stuff
#extract all files at once
cat *.tar.gz | tar -xvf - -i
cat *.tar.bz2 | tar -xzvf - -i
# the i is for ignoring end of file

#bin-utils
#---------
cd bin-utils
#or equivalent, whatever your extracted folder reads. do an
#ls
#for listing the files in a folder
mkdir avr-binutils; cd &_
#make a folder for bin-utils and cd to it (in a single command)
../binutils-versionnumber/configure --target=avr --prefix=/opt/
    avr
make
make install
export PATH=$PATH:/opt/avr/bin
#if you haven't already, for checking do
#echo $PATH

#To do this automatically everytime, do the following
#in /etc/profile, add a line
#PATH=$PATH:/opt/avr/bin
#can use the following command
#vi PATH
#then i, and then use the arrow keys to go to the bottom, when
    done adding, press escape, and type :wq and press enter for
    saving (writing to file) and quiting


#avr-gcc
#-------
#again cd into avr-gcc first then
mkdir avr-gcc; cd &_
```

```
../gcc-versionnumber/configure --target=avr --prefix=/opt/avr --
    enable-languages=c --disable-nls --disable-libssp
make
make install

#avr-libc
#--------
#cd avr-libc-version
./configure --build=‘./config.guess‘ --host=avr --prefix=/opt/avr
#note the quotes are not the usual ones; on the us layout, they
    are on the ones on the left of 1
```

For avr-dude you can download and follow the instructions. Nothing complicated there. For bootloadHID, you may have to build the PC part. Once done, do the following

```
cd /to/where/you/built/it
ln ./bootloadHID /usr/local/bin/bootloadHID
#create a link so you can run it as bootloadHID in the terminal
chmod 777 /usr/local/bin/bootloadHID
#give it permissions to work
```

Brilliant. Now you're very close. Just burn the firmware given in bootloadHID (assuming you've built 'temperature' in accordance with the schematic given earlier) using your favourite programmer.[1]

3.4.2 *Compiling and Burning*

Building temperature is easy peasy, lemon squeezy. Shot the program pin (as you've chosen it in the bootloadHID firmware) and reset the device. Then just goto the temperature folder and fire

```
make
```

and it should even program the device. If something goes wrong, try

```
dmesg
```

and ensure your bootloadHID shows up by unplugging and replugging. Some pitfalls include

1. Not setting the fuse-bits to use the external clock. If you don't know what settings to use, you can use the ones given in 3.4.2.
2. Using an ATmega8L instead of ATmega8 (L is a low voltage variant that is not usually stable above 8 MHz, although in the lab it did work)

To verify you've programmed 'temperature' correctly (well you can't do it wrong; either you program it or you don't) then after removing

1 CAVEAT: You may need to change the pin you're using to indicate you want to program, in accordance with the hardware and recompile the firmware before burning

the program pin jumper (remove the shot) and resetting, when you try dmesg, you should get NPL temperature. Cool.

```
H-Fuses : 11000001 = C1
[Preserve EEPROM, Use custom Clock settings (for external crystal
    )]

L-Fuses : 00101111 = 2F
[Enable Brown Out Detection at 4.0V, ext. crystal high freq]
```

It would now be a good time to connect all the dipoles (put them far apart) to temperature using the connectors.

3.5 LIGHTS CAMERA ACTION[2]

Yes literally that. Make sure you have enough light falling on the dipoles, they're all in the view of the camera and then fire from the terminal

```
./latticeAnalyser
```

Hopefully, it would run without any issues. If it crashes, well we'll discuss that in a bit. Let's start with testing how well the temperature unit is performing.

3.5.1 *temperature Blind Test*

So called because we won't be using the camera at all for this functionality. We'll first fire the electromagnets in the dipoles one by one to ensure everything is functional.

```
temp 0
```

Change the zero sequentially to test the corresponding dipoles. At times, even when everything seems fine but the dipole doesn't move, the wiring of the electromagnet is wrong.

3.5.2 *Actual Experiment*

Once the temperature has been confirmed to be functional, setup the dipoles in a lattice configuration and then just type

```
0
```

The number signifies which camera to use. So if you have only one, zero is the way to go. Else you can experiment. A bunch of windows will open up. Let me list them out

2 this section onwards, its assumed that you have 'temperature' programmed and the dipoles setup ready

1. Source: This will show the raw footage from the camera, but the ROI (region of interest, the entire frame by default) will be 'undistorted' using prior calibration data.
    a) Selecting ROI: You can drag using the left mouse button to do so.
    b) Select Colour for the colour filter: Press 'C' (case sensitive) and click on the desired colour in the window.
2. Filter: This will show you the grayscale image after applying a colour filter.
3. Contours: This will display the contours detected and show the ellipses fit, along with the dipoles detected and their corresponding information.
4. Settings: This has various sliders to fine tune dipole detection. Here are the sliders and what they control (omitted when its obvious)
    a) Min Ang Vel Sq: If the time average of energy is below this value, energy is pumped in using temperature
    b) Hue Tolerance
    c) Saturation Tolerance
    d) Value Tolerance
    e) Minor Axis: Size of the minor axis
    f) Major Axis
    g) Radius: Of the only circle in a dipole pattern
    h) Ellipse Tolerance: Tolerance for all three dimensions specified earlier
    i) Brightness Inverse: Smaller this number, brighter will be the result in colour filter. Useful for better ellipse fitting.
5. Debug: Ignore it for now.
6. A massive graph like window

Set these to get all the dipoles visible in the ROI to be detected. When done, press 'p' (case sensitive) to start using the current frame as seed. TODO: explain a seed frame. You should now see the graph getting populated. Ensure that the dipole ids (written in square brackets) correspond to their physical positions. If they don't, press 'i' (case sensitive) and locate the terminal. Type the dipole id corresponding to the 0^{th} place, then the 1^{st} place and so on, for example

```
3 1 2 0
```

and press enter. The changes should be visible immediately. Now enter the coil angle of the dipoles (preferably keep them either perpendicular or parallel to the x or y axis as seen through the camera) by pressing 'o' and type the value in the terminal (in degrees). If after increasing energy, there's no movement, press 'I'. To record all the data since the selection of the seedframe into a file, press 'F'. The file is named "latticeAnalyserRENAMEorLOSEme" in the same folder as the program. To exit, get to the terminal and press ctrl C. You can press q but that will just bring you back to the terminal. And that's that.

NOTE: For the single letter commands, ensure one of the graphical windows (except the graph) is selected, else the interrupt won't trigger.

3.6 MISSING FEATURES

Following is by no means an exhaustive list of missing features

1. Ability to save settings
2. Ability to change the calibration file name
3. Very poor file name creation for saving the data

Part II

APPENDIX

A

LISTINGS

A.1 TIME LINE

Daily log for the project:

Time Line

```
SUMMER '13 TIMELINE
--
** July 26, Friday: Not well again. Tried to work on the
    documentation

* July 25, Thursday: Working on setting up latex on linux along
    with dOxygen for documentation. The latex bit failed. There
    were various problems even after painfully installing TeXlive
     to work with ST3, while compiling the classic thesis
    template.

* July 24, Wednesday: [Project Milestone] Completed testing with
    a 2x2 lattice, with energy pumping and it worked just fine!
    Obviously it required a lot of debugging, for electronics
    never works at once. Plus there were other software issues.

* July 23, Tuesday: [important day] Finalized the HSV colour
    filter.  Further added calculation of temperature using a
    timed average. Added support for making the pin
    configurations readable from file. Then temperature ran into
    issues and was fixed (int char problem) and put the last
    piece of the puzzle.

* July 22, Monday: Figured out the best way to make temperature
    take generic inputs for triggering the electromagnets.
    Further the ellipse detection was made more strict to dis-
    allow ellipse/circles that are not of suitable dimensions.
    Further worked on an HSV colour filter instead of the RGB
    model to support higher speeds.

** July 20 and 21, Saturday and Sunday: Working on the
    documentation.

** July 18 and 19, Thursday and Friday: (Not well) [at home]
    Tried to setup plplot in windows using CMake and VS11 (2012)
    failed. Tried with minGW that too failed. Tried upgrading
    Sublime Text 3 to work with latex, that too failed. Working
    on updating the documentation. Also worked on the secondary
    project with some success.
```

```
* July 17, Wednesday: (Not well in the morning) Physically
    completed construction of four new dipoles, ready to be
    tested as a 2x2 lattice.

* July 16, Tuesday: The modding problem was fixed (using modular
    arithmetic) in the dipole angle detection also. Dipole
    oscillations were setup with energy pumping using the
    temperature. Another dipole was also tested. Colour
    correction was changed from being based on RGB to HSV. This
    improved dipole detection at higher angular velocities.

* July 15, Monday: Attempted an upgrade of the gstreamer plugin
    which failed.'tempreature' was made functional again and
    damping was tested using the realtime graphs. The modding
    problem in the angle detection for deciding when to fire the
    electromagnet was also fixed.

** July 12-14, Friday to Sunday: Working on a secondary project.

* July 11, Thursday: A simple two point setup was created with
    one needle held from the top and bottom (used magnets to
    start with then removed them for testing damping) using glass
     slides and it seems to be better than what we've seen so far
    . Further, with plastic it is highly damped.  However, we
    tried to mount the needle on both sides with screws (for a
    metal cavity) and the damping was very sensitive to the
    amount of pressure. Further the idea is hard to implement
    cause the vision detection can't be done on it then, atleast
    the way we're doing it now. Also, camera calibration was
    implemented to correct for camera defects using a chessboard
    pattern. The systematic errors didn't disappear.

* July 10, Wednesday: A little more of error analysis and dipole
    air elevation setup for all the dipoles was done (the fourth
    was setup and left for drying). Used an interesting method
    for making the dipoles. It was found during tests that with
    the heavy bottom, the needle setup (with the pointed edge up
    !) had almost the same damping as that of the air elevated
    setup. Further it was concluded that more accuracy is
    required for the air elevation setup to work properly. It was
     hypothesized that the main source of 'angular friction' were
     the guides that held the needle upright and not the needle
    tip.

* July 9, Tuesday: Did the error analysis. Used a hell lot of
    GNUplot. Experimentally (that is in the physical world) setup
     the third dipole.

* July 8, Monday: Working on writing a method to record all the
    observations into a file. Further figured out what and how to
     do, for estimating errors and accounting for the systematic
```

```
    errors. Further started the dipole modifications for setting
    up the first dipole for air elevation. This was done and
    found to have worked rather well. Better than the needle
    counter-parts (but no reliable comparison can be made yet,
    the systems compared weren't 'identical enough')

** July 6 and 7, Weekend: Spent on an alternate project

* July 5, Friday: Two interesting things were done; one the long
    lingering bug in the modding was resolved using assistance
    from the atan2 approximation. Second, the aluminium disk was
    fitted on a needle and left to dry. Exciting cause that would
     finally conclude whether the air elevation method could work
     or not.

* July 4, Thursday: Cmake was fiddled around with and a graph for
     kinetic energy added. There was a periodic shoot up in the
    value which was caused by a mod problem in the angles. It was
     rectified. The dipole's hardware modification has also been
    initiated.

* July 3, Wednesday: Realised that a DIY air hockey table would
    be a better way of searching and it was. In the
    latticeAnalyser implimented dual graphs (one for the angular
    position and the other for the velocity). Also implemented
    the atan2 function for correcting the angle problem that used
     to occur.

* July 2, Tuesday: Figured how to use plplot, the basics. Got it
    up and running for 3d points and surfaces. Figured how to use
     grids, labels and enabling realtime support. Built and
    merged a minimalistic version of the same to the
    latticeAnalyser

* July 1, Monday: Air hockey tables were searched for some toy
    ones found too. Attempted the writing of an algorithm for
    auto numbering of points in a lattice. Figured it was not
    straight forward to implement with the usual C++ syntax. Will
     instead made provision (partially) for swapping manually.
    Also, plplot, a suitable plotting library was found and
    installed.

** June 9 - 21 : Monsoon School; Physics of Life, NCBS Bengaluru

* June 7, Friday: [Project Milestone] It was realized that
    neither the sachetIO was required (the data length can be
    increased to 128 bytes without any difficulty just by
    changing the report length in the USB HID configurations),
    nor was a current amplifier. However, today a dipole was made
     to turn continuously using the latticeAnalyser and the
    temperature. It was found that the current is more than
    sufficient for the current task. Further, for now, pins of
```

```
    the MCU can directly be used. Stage one has been accomplished
    . Now it remains to reduce friction (for the damping is far
    too high) and to finalize the circuit for the setup. Then the
     actual experiment begins.
* * June 6, Thursday: Added a buffer function to the sachetIO
    library. Looked up methods for current amplification
* * June 5, Wednesday: Created and tested sachetIO, a library for
     sending large arrays, required for communicating over HID
    USB. This was debugged and tested. Non buffered version is
    ready.
* June 4, Tuesday: Succesffully linked the latticeAnalyser (after
     compiling half of it with C, the rest with C++). Modified
    temperature to include a simple protocol and using that
    controlled the electromagnet of a single dipole from
    latticeAnalyser.
* June 3, Monday: Started with the hardware. Configured SP12 (the
     programmer) then bootloadHID, flashed a board to support a
    bootloadHID interface.

** June 1 and 2, Saturday and Sunday: Updated the documentation

* May 31, Friday: Tried all sorts of methods for air suspension
    which failed to work satisfactorily. Despite suspension, we
    couldn't achieve a state that had low enough torque (either
    perturbation or friction, atleast one was visible)
* May 30, Thursday: The vacuum cleaner setup had to be used. The
    box was drilled accordingly and the test failed partially
    anyway. Which is to say that if it is light enough, the
    suspension does take place, however the vertical oscillations
     can not be removed.
* May 29, Wednesday: The pump was built but it failed the test.
    The air pressure generated was negligible. i7 was configured
    alongside and the Lattice Analyser was built on it (had to
    make the multi threading optional using macros) using OpenCV
    with OpenTBB. Tests were run and it was found to be fast
    enough for an 8 x 8 dipole matrix.
* May 28, Tuesday: Multi-threading retried, wasn't quite
    functional last time. Looked up various techniques for multi-
    threading and froze an algorithm. The pump couldn't be built
    for parts weren't available.
* May 27, Monday: Multithreading attempted and succeeded,
    although not a very good release. Making progress in
    installing IPP.

* May 21 - 26, Tuesday to Sunday: Not well; Succeeded in
    compiling the code in windows. It's slower. Looked up
    multithreading using C++ 11

* May 20, Monday: Time Lag measured and found to be roughly 3 to
    4 frames behind. Not quite acceptable. Attempting to install
    OpenCV with IPP
```

```
** May 18 and 19, Saturday and Sunday: Completing the
    documentation for the same. Thought of a way of testing the
    time lag.

* May 17, Friday: The algorithm was successfully completed to
    measure 360 degrees. PLUS, completed the frame recording,
    identification of each dipole as unique and dumping the data
    out in file AND its testing with uniform motion which it
    passed with flying colours (which is to say in the visible
    range!, because proper standard deviation tests haven't quite
     been done yet) The vision part of the anlyser is almost done
    .

* May 16, Thursday: Working on dipole detection. The algorithm
    has started to work partially. It still does a mod 180
    detection.

* May 15, Wednesday: The magnetic lifting worked, but friction
    reduction failed. Rather interestingly the dipole would align
     to the suspension magnet's field. Plus, today the spot
    recognition algorithm was finalized and it seemd to be
    perfect.

* May 14, Tuesday: Trying to get the webcam to work, eventually
    acceded to installing everything on a desktop machine. Worked
     on reducing the friction further

* May 13, Monday: Completed the proof of concept version of the
    latticeAnlyser. Tomorrow we plan to print the coloured ovals
    and test

** May 11 and 12, Saturday and Sunday: Read the opencv tutorials
    when the algorithms started appearing and fitting the bill!

* May 10, Friday: Continued with the setup, finetuning,
    installing other applications, making a documentation
    alongside for better support next time, added a shared folder
     between windows and linux

* May 9, Thursday: Managed to get a few things up and running,
    still setting up ubuntu to run with hardware acceleration,
    failed at trying to get the webcam to work, installed the
    build tools, opencv etc.

* May 8, Wednesday: Met with Dr. X (forgot the name of the person
     at NPL I'm working with) and concluded OpenCV and linux are
    what I'll use. Initiated the downloading of required
    applications, including virtual box and an ubuntu image
```

A.2 GIT COMMITS

```
commit 8e385842ed187496bb320697afceea6ac735183f
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu Jul 18 22:23:15 2013 +0530

    Updating the documentation

commit 3fd0c2734e2cb00eaf4e036dee732ff0c38dfca7
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 16 17:28:58 2013 +0530

    Minor change, major fix (hopefully) fixed the modding problem
         (after rediscovery:P)

commit 6e6bf4f59a34b0e14246fd8e28845bdc2de401df
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 16 17:09:45 2013 +0530

    Converted colour detection to HSV from BGR

commit ce55da42951066b207320efe944169b50b82ab08
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 16 11:40:47 2013 +0530

    Slight failures in the modding problem also resolved..finally
         based on equivalent classes from math

commit a974f4c3b2aa02e06c3f7df8a6fad2fdd0f66223
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 16 11:02:29 2013 +0530

    Temperature: SOlved the modding problem for the axis

commit 33fda35ce344b612b786853b044e2229811fefcf
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu Jul 11 15:44:37 2013 +0530

    Completed addition of distortion correction and it has worked
         well, but hasn't solved the problem

commit 7ad448b39f301eceb69a8bdb1206c67f8f8196ca
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu Jul 11 12:48:02 2013 +0530

    Documentation: README updated

commit 3efc849a25294a881bf05e99af4afa02d6031e2c
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 10 18:05:57 2013 +0530

    More error analysis and found an error in the error analysis

commit 25debdf7a1a06aeab24b14f66eb597eda39ff7d9
```

```
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 9 17:39:49 2013 +0530

    Doing error analysis on the observations from the software to
         figure out what to optimize next

commit e2e8270db661e3e902626f188f32d1950445b404
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 9 12:01:56 2013 +0530

    Ready for error analysis, recorded all the data into a data
        file

commit 0f8893ca05b450d67c3a23ade21c6d74c693149a
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jul 8 17:42:23 2013 +0530

    minor or no change

commit 5c26dd130980019fa34f5700cc5512ddb80d60c7
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jul 8 15:24:19 2013 +0530

    latticeAnalyser: Functionality for re-assigning dipole
        numbers made functional :P

commit 55ea71e74a5416dd7f74171acf7ea1d1ff48d754
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jul 8 14:09:03 2013 +0530

    latticeAnalyser: Another finer improvement on the inst vel
        calculation

commit 6ed09058eb15263ae4ba74775f9b4cafafd9bc83
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jul 8 11:23:12 2013 +0530

    Documentation: README updated

commit 1c822e7fe983d18e84be3e3ebe9a8c084ab72f16
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 17:49:41 2013 +0530

    Angle modding problem finally resolved for the greater good :
        P

commit f83f81c76577417bef7725a040eb638f66a7906d
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 17:12:24 2013 +0530

    Error message in graph; its cause has been resolved and
        consequently they've been suppressed
```

```
commit 35fde5f24b524bc4bcb5ccbd9ad6a72608ed7f54
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 17:38:25 2013 +0530

    Multi threading now functional, though unstable, angle
        estimator still troublesome, blips

commit 9bda4ff89f6c343609f7a9be0a95d784a8b021c2
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 15:16:45 2013 +0530

    Control for min Angular velocity (for use with temperature)
        added. Angle estimator now even more stable. No glitches
        so far

commit 03154549ee38e6b4421f0a27797c4c00dc9626e7
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 14:27:20 2013 +0530

    Angle estimator now more stable, though still gives
        fluctuations

commit aa67fe4b451af9af9a12acb899e6d28c2e1359cc
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 14:05:55 2013 +0530

    Angle estimator now much more stable, but still peaks out

commit a0dfdae7cda0a9e35b90632d3513ab5a293be6fe
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jul 5 13:49:14 2013 +0530

    Angle estimator tested, poor precision with atan2, now using
        atan2 assisted modding to ellipse based angle detection

commit c515d301935b620fcc2004f5057c63a9881e1974
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu Jul 4 18:23:17 2013 +0530

    Documentation: Screenshot was missed out

commit 9f0faf13c588cbd325a2067f0c54005bdb9f0fe7
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu Jul 4 18:21:42 2013 +0530

    Documentation: README updated

commit a5880c4b1185ed0e5557d51181e3165e8345a7ce
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu Jul 4 18:14:21 2013 +0530
```

```
    Kinetic Energy graph added, problem located and calculation
        corrected

commit 10d125df11fbd250ee3a04e2d51a8bde97cd3f87
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 3 17:07:57 2013 +0530

    Documentation: Readme updated

commit 3b8a0fcb44b4be72622fb8bff6d7c226a60da80f
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 3 16:21:51 2013 +0530

    Documentation: READMEs updated

commit 16438bd90783665801e5f334421863af78544773
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 3 16:19:25 2013 +0530

    Added auto clear on overflow for the graphs

commit a37c4fdb0fce9d140c7ab2e169270392b46daf26
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 3 15:19:39 2013 +0530

    atan2 used for angles; problem of outlying angles solved

commit 132b1d4b32c794b9bca31ff575521290e30f67fd
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 3 15:00:05 2013 +0530

    2 graphs in one created, one for velocity, one for position

commit 282c04b3bd3824e77e6d57dcac5f393f2adbf988
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed Jul 3 11:48:52 2013 +0530

    3D graph plotting, updated to show angles

commit 8dedf79959fcb26c39e1ff7c0ec334fc8bf65732
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 2 17:07:12 2013 +0530

    Fagocytosis successful: Graphing including within the program

commit 5dee7e255f4ac1f281399937619af5db8c012a80
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 2 16:59:56 2013 +0530

    Fygocytosis done, implementing the rest of it

commit be11592758b03ab9487ffe985ee8de45005db691
```

```
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 2 15:47:53 2013 +0530

    Realtime update finally done for 3d graphics

commit a91a624ccfe896b9acfddd2efbeb24fc5b5d3b24
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 2 13:59:37 2013 +0530

    Now stable

commit bdcc662088707dd91789a53748d49fee1bef1d76
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jul 2 13:52:33 2013 +0530

    Working on using plplot and tried to write an algorithm to
        support auto grid numbering; but doomed

commit 7c8f275d2e1059b25e33fc34d381a74ed54a3e8e
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon Jul 1 11:28:29 2013 +0530

    After summer school

commit 4b1a5a3758caa379c2cdbc2a7e9e657953b5757a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sat Jun 8 00:00:05 2013 +0530

    Documentation: code listings updated

commit 261de8437cfc18f8f11d61964343d8529603aa1c
Merge: 803d298 502f050
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri Jun 7 23:57:40 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit 803d2984bea33f8fb0e206a2f378638048ec90b6
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri Jun 7 23:57:06 2013 +0530

    Documentation: Progress for the week added, including the
        milestone event

commit 502f050adf2d40f697e70b318a8f3705e8837020
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri Jun 7 18:02:18 2013 +0530

    [MileStone commit] latticeAnalyser and temperature: Sight
        based energy pumping into the dipole successfull
```

```
commit 2a24425f8af44d372d9a5b3887c0db0b7aa9989b
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri Jun 7 09:00:04 2013 +0530

    latticeAnalyser: trying to compile on windows

commit ae2c44716c144a80ce276959cdae2df1a92856bf
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu Jun 6 16:24:23 2013 +0530

    sachetIO: buffer function fixed and tested ok

commit efa7bdded264e2aeb81945a807e975637ae8e47a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu Jun 6 15:32:22 2013 +0530

    sachetIO: added the buffered read function

commit 22b0d4a24d5281dd1753aba4ab1b6a71b341bf7d
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu Jun 6 13:21:03 2013 +0530

    Documentation: Readme Updated with Wednesday's progress

commit ffcc38dffd780bb33e426f323e42626cd5db1f6b
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed Jun 5 22:42:27 2013 +0530

    sachetIO: bare minimum fixed and tested

commit ea3e045f05988e42d9eb2dd884a3ceff949e4abc
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed Jun 5 21:45:36 2013 +0530

    sachetIO: Fixing recieving

commit 07b8135ddeeed04430704625bc95cc3091d2098a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed Jun 5 21:13:32 2013 +0530

    sachetIO: Sending tested

commit 0cb0ee0d725288a914cdc7327b7d8271a4545a13
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed Jun 5 20:46:09 2013 +0530

    Debugging sachetIO

commit a0e26da38bc55571ae426e30f820aa3b83c0aa8a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed Jun 5 18:55:06 2013 +0530
```

```
    lattice and temperature: sachetIO added, not tested

commit 91ad9c62158050abf17d2845df6a73e459bd6dc6
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed Jun 5 11:13:13 2013 +0530

    Windows thumbnail updated, bad gitting

commit 971107d27875483c7da099fbc919bb3ff6e10cbc
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 17:38:28 2013 +0530

    Documentation: Readme updated

commit 1ae6231a7947ff2dfb60cde8c5ff6f04126fe6dd
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 17:17:29 2013 +0530

    latticeAnalyser: Brightness level added amongst other mods

commit 66e0a71c8883402d9f417ceee81fd90aae6b3978
Merge: 5197b0e f74225f
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 16:29:11 2013 +0530

    Sync between i7 and the laptop i3.

commit f74225f45621d6787eabb6aa8f1a6b8b2a2ed838
Merge: a9e3f77 0acd87a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue Jun 4 16:25:39 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit a9e3f773ce60d4d2208df7916ded4eb265dc3374
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue Jun 4 16:25:00 2013 +0530

    Dipoles for printing corrected

commit 5197b0e7cdf547ab1f2d0ac919d75a7374091b18
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 16:19:24 2013 +0530

    Completed Firing the electro using the console

commit 61a7d99bbaca3a1ff5bd242605a5d02931189fb1
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 12:28:48 2013 +0530

    Protocol tested | functional
```

```
commit d216f3253d72c34b41a37c3de40d3de4c7e4df61
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 12:10:52 2013 +0530

    lattice and temp, both about to be tested

commit 1c300638b1a4f72d4bd0c1adc871aa4e331f604a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue Jun 4 11:44:14 2013 +0530

    Dipoles for printing prepared

commit 5716ed3aac119b7f9ed73fdf91fd3cc5d582f3ad
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 10:20:38 2013 +0530

    temperature: modifying directory structures

commit 0acd87a44061dbad55c16cdaf3edeb83f0ba282e
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 10:12:00 2013 +0530

    Committing missed out changes

commit bc84259a0622c96483cd2a2e6192c64a50001387
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 10:10:06 2013 +0530

    Improving makefiles

commit 41bf3439997260e2ea985159dbeac26f3c932e09
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue Jun 4 09:59:29 2013 +0530

    latticeAnalyser and temperature: Now test built and stable
        together

commit 87e256921410d3b3dd64d3cf5c7bee2a07f2d07c
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jun 3 18:25:20 2013 +0530

    Unstable: Testing temperature with latticeAnalyser

commit 414bd316aee99f833fb0990d46ed8e8e812bb431
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jun 3 16:22:39 2013 +0530

    Completed making a directory structure and compiling with a
        makefile, the original program. Plus brought the
        latticeAnalyser up to pace, ready to be tested with a
        makefile
```

```
commit 45870e0b999e2a2fb5262c1724524c32344146e0
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon Jun 3 15:11:22 2013 +0530

    temperature: Brought to life

commit 45ec731b3acdd2453b9b398f2bab241f44a1d44c
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon Jun 3 09:06:32 2013 +0530

    Documentation: Minor changes

commit c9dba8a5a11417bcd88e5871d21447634d8dcdf8
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sun Jun 2 21:21:18 2013 +0530

    Documentation: Report completed for this week

commit a8caec9bc8d48e292500a67aaa4ab30307ce5f3c
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sun Jun 2 20:23:22 2013 +0530

    Documentation: Updating the report, air levitation bit

commit 81911d7db6ad61c78adb43cdf38f923463e99ea2
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sun Jun 2 13:59:47 2013 +0530

    Tested and functional in windows

commit 545b4b2a6f2fe5b479b2b195db3e33bf07dcbd05
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 31 17:49:46 2013 +0530

    Documentation: Updating the readme with todays update

commit 6ba72ab3722dde5108c24f36cf1f0b7a431a5473
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 31 17:47:33 2013 +0530

    latticeAnalyser: Working on the CLI to support hardware test
        and interface

commit c02f28b3cb63b80de999d6e48caf7d1c11ee134b
Merge: e799380 4124d6f
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 31 09:37:56 2013 +0530

    Merge between my laptop on the i7

commit 4124d6f02088e68eee26a717a2d311d878e94e14
```

```
Merge: e7fdc40 b9fa975
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 31 09:37:40 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit e79938071a4774aead45190065d4b89ec897f172
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 31 09:36:41 2013 +0530

    Documentation: Readme updated

commit e7fdc40f167da5cc37e4be6d00126a16139b6392
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 31 09:35:45 2013 +0530

    latticeAnalyser: Working on adding hardware support

commit b9fa9750312fbd712d4f67e412c6f8fa439d2e3f
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu May 30 14:48:47 2013 +0530

    Documentation: Updating the readme

commit e5ab8b0d86d65f425b6cb8882fe2e3e217ef32f8
Merge: 96497a8 8641fbb
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 30 11:14:22 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit 8641fbb08ea2310ef683cf273e246372083ca0fd
Merge: 947f4a9 eb3aa7b
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu May 30 12:03:26 2013 +0530

    Merge remote-tracking branch 'origin'

commit 947f4a92fdb40167b79fbcbf0703961068e9d928
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu May 30 12:03:04 2013 +0530

    Documentation: For installing the avr tool chain added

commit 96497a8bfdaa355470b28457c0c3258e58246727
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 30 11:13:55 2013 +0530

    Testing Air Levitation: Drawing for drilling corrected
```

```
commit eb3aa7ba02b8fe4e8fe461a1fcf3f48985637ac7
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 30 11:01:53 2013 +0530

    Testing Air Levitation: Drawing for drilling completed

commit 88a86f88320485cf29a4bd549fc596574efb5cd7
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 29 18:18:42 2013 +0530

    Compiled with linux, gcc and opencv built with openTBB and
        IPP, in i7, seems to be screeching fast so far

commit 97e92d0a1c358a48f6268620e8f27550fe4df930
Merge: a06ca8c 88a86f8
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed May 29 17:27:41 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit a06ca8c3c46b595a18fd2857b67181a58d426b3c
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed May 29 17:27:18 2013 +0530

    Adding the dipole image

commit dd49ef3da14113786804d091524dc9fe985f1ba2
Merge: 7118975 c5783f2
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 28 17:53:20 2013 +0530

    Resolving conflicts

commit c5783f2c0326b6f9932a79cb1ebcb6423eb5f3f4
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 28 17:49:26 2013 +0530

    Finalizing, now stable in windows, tested

commit 9094e4956634cf6552d4a13034681c9aa467ced4
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 28 17:40:52 2013 +0530

    Multi-Thread Atomic Sync Completed to satisfactory level

commit 54ef6c6562e5191d9a13fc28f4af5bb86aa0d9b8
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 28 17:12:15 2013 +0530

    Display atomic multi-thread, mysteriously functional
```

```
commit aea12d12ff7a7392a9a8aa6bfc24b9480fac2bed
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 28 16:10:41 2013 +0530

    Atomic Sync implemented for capture frame

commit 4c37bfb7a65726b0c5d4c0be78efa8adf308db20
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 28 15:22:27 2013 -0800

    Working on atomic aspects. Now switching to visual studio

commit ec9f99c1950c6590f14f011b5ced221a71bb44a4
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 28 11:18:46 2013 -0800

    Working on the multithreaded aspect, somehow still faster
        without multithreading

commit 7118975120137a637efbdfd00f4f37029dcb48f5
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 28 15:37:55 2013 +0530

    Minor changes

commit 1937a9b1a85d298adf71082c7d4d5d0b2e2459ed
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 27 16:58:04 2013 -0800

    Support for multi-threading added, also compiled with IPP

commit 6344695813909496233054173 60e3337e2c4b559
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 27 11:40:47 2013 -0800

    Multi threaded for reducing lag

commit 118b1118a3a9c9a2450c97857dfa1f3c7cdce17a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 23 15:42:56 2013 +0530

    Template Record modified and commited

commit b15b3b68ecf14b8f6412e68a0e32e07fafb8b320
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 23 14:17:21 2013 +0530

    Template Record zip added

commit 69cc3d0baf0c5593220504d6184379cf4f1eeaac
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 23 14:16:29 2013 +0530
```

```
    Adding the template amongst other things

commit 65be7489adf56a9d1ddb5429a812446b6d5a90fe
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 21 21:07:12 2013 +0530

    Minor bug, memory leak corrected.

commit d491b930c62d39d8d9032d7a7361fc302fccd29e
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Tue May 21 19:25:15 2013 +0530

    Windows Support Added, amongst other things

commit 76de5131ce3281f6a67c6fa4e2e04390eb9ffddf
Merge: 92b9031 5bd296f
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon May 20 17:16:48 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit 5bd296f6330bc6a994cdd3cb568944ddae03ddd4
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 20 17:16:10 2013 -0800

    Corrections

commit 7f19fcacfdd84895a66df61918983d26b9966f5c
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 20 14:02:50 2013 -0800

    latticeAnalyser functional again

commit 4720ab5b6ce0310a35c4e67392430d8248b488c4
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 20 14:02:04 2013 -0800

    Test results for speed added

commit f546e50310b6a9721430739bd0d1710939d4f98e
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 20 12:48:50 2013 -0800

    Testing: Computation time

commit 92b903140f65e1a1dc13fa340d2ba60f467a907d
Merge: 6b87a0b 7f19fca
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon May 20 14:03:18 2013 +0530
```

```
    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit 6b87a0bea18e78b97272fa4d2948add994f9be0b
Merge: c18ab4f f546e50
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon May 20 12:51:15 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit c18ab4f832ab6905bd110cb35f97f462ad4d64e3
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon May 20 12:51:01 2013 +0530

    Testing Overall speed

commit d902a319f303aa07b7744b28b10c2b7f7d9d4704
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon May 20 08:49:51 2013 +0530

    Documentation: Pushing final changes

commit 3961ae759ca5d8cd49ea0fc5b1f117808af64357
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Mon May 20 00:59:40 2013 +0530

    Documentation: For this week, completed

commit 600d76a3c5a0157d2704643c7606a6f4845015a0
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sun May 19 23:32:05 2013 +0530

    Documentation: Working on it

commit 97dfb751a1e095d12c2bd9b8bfdb692f2c7fc9f8
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sun May 19 21:33:03 2013 +0530

    Latex documentation initiated

commit 965a960cc01ba8c12828c9c7184002abfa254516
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Sat May 18 21:18:13 2013 +0530

    Documentation: Graph for the result added

commit 2b54dcdccda1674acd1e55b4811deef84af22bd6
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 20:05:53 2013 +0530

    Documentation: Graphs added plus the screenshots fixed
```

```
commit b451f3235971d518bdb8b598d7d0617b30d1905c
Merge: 24299e9 e6992b4
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 19:49:02 2013 +0530

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit e6992b4d4dfdd247e13cc26792e5ae2e967b71e7
Merge: b99fa82 e6a9613
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 17:29:37 2013 -0800

    Merge branch 'master' of github.com:toAtulArora/IISER_repo

commit b99fa823120154b9605e9c7295f7b1095788940c
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 17:28:39 2013 -0800

    junk files removed

commit 2f90706321725486350bf5e5cab471880024732c
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 17:28:05 2013 -0800

    Tested with movement, Passed so far atleast qualitatively

commit d0713d1bc4387bc86ffa97ffa70e7135461c1c8d
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 16:49:45 2013 -0800

    Data readout successful

commit 9562182c76cdd8bd3332369b077a537979b38226
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 14:38:46 2013 -0800

    Unstable: working on saving the data in frames

commit 24299e9b82d126498eb9fdb2a423b73dc10c956a
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 17:48:00 2013 +0530

    Completing documentation

commit e6a961387d67fc44b61d9d4a7ff8cdbf3c9300c8
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 17:07:59 2013 +0530

    Test animation added
```

```
commit c4b267c3aee1b4a250b35947909b9b66d840e124
Merge: 592f37f e76bdab
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 06:26:55 2013 +0100

    Merge branch 'master' of https://github.com/toAtulArora/IISER
        _repo

commit e76bdab03a3563f80b2b6a7df35a03cf5016cb08
Merge: c577396 139aa07
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 10:56:05 2013 -0800

    Merge branch 'master' of github.com:toAtulArora/IISER_repo

commit c5773969054079d95f2b6ffa6374a70bc37edae3
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 10:55:28 2013 -0800

    Stable and tested, the dipole angle measurement for each
        frame is successful

commit 1b512fc3f783b9ceab973241bac17de424fcf374
Merge: 17b9e98 7ed3367
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 10:21:26 2013 -0800

    Merge branch 'master' of github.com:toAtulArora/IISER_repo

commit 17b9e981d3d6eca9e1bcb5057c6e9223615685b0
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Fri May 17 10:21:09 2013 -0800

    Dipole detection successful and stable

commit 592f37fa548922654bfae0d0df77f873243a1501
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 06:26:44 2013 +0100

    Image Rotated

commit 139aa076a25cb555d7ea8148a5ffc5d16efac3ef
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 10:36:18 2013 +0530

    Dipoles modified, ellipse thinner

commit 7ed336730f865b7fa6dd42b966062c47b12414ce
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 09:51:06 2013 +0530

    Circle Ellipse for Dipole
```

```
commit a28114a6da1a2c41eea58773edbe7b36be55bfd1
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Fri May 17 09:17:00 2013 +0530

    STABLE upto mod 180. Updated a few things

commit 71f70879c14c7ee04accd4967eb30efadcfb503b
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 16 20:01:51 2013 +0530

    Dipole diagrams under development

commit 67091067fca83d9bd6b82435a147fa147b73a9ce
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu May 16 18:30:23 2013 -0800

    Stable and angle detection, Mod PI stable

commit f0331273de52a2cbc08375adf828c8c19c793b5a
Merge: 0a238d6 fbf2852
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu May 16 17:51:57 2013 -0800

    Merge remote branch 'origin'

commit 0a238d683fca9596f1f4e39b476239495fb431a0
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Thu May 16 17:50:47 2013 -0800

    Stable but not functional, the angle detection

commit fbf2852326d676b5249558b81bd9fb9359f5c050
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 16 11:23:54 2013 +0530

    Darker

commit 3e35bb152be886098e2ad3c627eb97d655ac4156
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 16 10:54:22 2013 +0530

    Ellipses, exported

commit 8a77a1cda5fedb8a31384aba38e7f9c59822d7d9
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Thu May 16 10:34:13 2013 +0530

    Ellipses: Dipoles now have ellipses

commit 1352099dd4b874d3c40feeaf1bb60ba2ecc0e0ac
Author: Atul Singh Arora <toatularora@gmail.com>
```

```
Date:   Wed May 15 16:46:49 2013 -0800

    Readme fixed

commit d4dde74debd2808ed61985555efed5e3f249faa8
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 16:45:34 2013 -0800

    latticeAnalyser now Stable

commit 8185f26452a568f24436dad15d7721c6abf09507
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 15:46:39 2013 -0800

    readme fix

commit b5c2f8ef6f30b41f8922595de45b2d21078ee535
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 15:45:36 2013 -0800

    readme fix

commit a1882f5b9ca8608c5f5078959990de0f4328d0af
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 15:44:17 2013 -0800

    Proof of concept for Hough

commit c4c85189bc2a33a21fa4bd18e746d5faff8f4cc3
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 14:54:41 2013 -0800

    Proof of concept is complete

commit 7989309295af080c63d2359e228548a83df384f0
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 14:32:34 2013 -0800

    Test version functional with contours, for circle detection

commit 48eb9ca66cd831e7622c3411e5d772dc48861853
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Wed May 15 13:19:45 2013 -0800

    Test with circles, using contours is fine

commit 38c0db3f5a80f766d6ce392874cfc6dbd0c98084
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed May 15 13:01:00 2013 +0530

    Plaussibly unstable, rewriting latticeAnalyser
```

```
commit ae3ed9a080a08f1293835340bbe27f67bb03a092
Author: Atul Sinh Aurora <toatularora@gmail.com>
Date:   Wed May 15 11:51:55 2013 +0530

    Added the dipole drawing

commit bbcf6834ea3ea5a8bc8beaccc111802da44ac762
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 14 16:39:23 2013 -0800

    Readme final update for tuesday

commit 7d37eb031af55077a3406fcb8636be9ff9c50688
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 14 14:47:01 2013 -0800

    Updating the readme

commit b468d8f06c6b1ce0a6ab8584c144766bb3c832e6
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 14 14:45:57 2013 -0800

    Updating the readme

commit 9050ca5cf5520a743c9fd15b1c62d1a4ebda7e01
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 14 14:10:03 2013 -0800

    Trying to push changes

commit 85749c48d6c87e4ae693daff193ff5c6682fe577
Author: root <root@ccf1.ccf.npl>
Date:   Tue May 14 12:58:47 2013 -0800

    Up and running in the new setup

commit b31da232eef661b854d86ea0f495f2f491a74a58
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Tue May 14 12:51:14 2013 +0530

    Migrating :(

commit a8cbef25fc90a3c878456f921b2a029dd5b0b058
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 13 12:09:08 2013 +0530

    Basic Colour Filters up and running

commit 118a3519f5123a64ac8049229462213f02f6052e
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Mon May 13 11:54:40 2013 +0530
```

```
    The lattice analyser is being built..has begun to 'see'
        things

commit 75bd2af30f3ee056eb2b840877782803f4a8bda5
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Sat May 11 15:25:51 2013 +0530

    Setting up the structure

commit f3751b794796ddb116da91dbf4d034faec6a5998
Author: Atul Singh Arora <toatularora@gmail.com>
Date:   Sat May 11 13:52:10 2013 +0530

    Readme Updated
```

B

SOURCE CODES

B.1 LATTICE ANALYSER

Following is the main C++ source code of Lattice Analyser.

latticeAnalyser.cpp

```
/*
  filename: latticeAnalyser.cpp
  description: This is the main application which analyses the
      lattice and
                  1. controls 'temperature'
                  2. records dipole position (thus angular
                      velocity) as a function of time
  baby steps (TM):
    1. Proof of concept stage
        a. Find a suitable algorithm
        b. Make the required modifications
            i. Add a colour filter
              Implement two colours [done]
              Add two sliders for adjusting tolerance [done, but
                  need to refresh something!]
              Add GUI for selecting the colours [done, but not a
                  gui so to speak]
              save settings [not doing it]
    2. Look at it grow!
        a. Enable screen cropping [done]
        b. Write an algorithm for ellipse to dipole conversion [
            completed]
        c. Save data for each frame using a circular array of
            sorts [done]
        d. Output the data perhaps in a text file  [done]
    3. Testing
        a. Test OpenCV's computation time [done]
                    b. Compiling it on Windows [done and as it
                        turns out, useless]
    4. Optimizing
        a. Multi-threading frame fethcing [done]
        b. Multi-threading display update [done, but failed to
            improve, infact made it worse]
        c. Multi-thread using mutex [done, no real difference but
             not slow either][it is actually very slow]
                    d. Atomic Sync for frame fetching [done]
        f. Atomic Sync for display update [done, and apparently
            faster! Yey]
            [There's something wrong though, since the frame
                jitters like a bad codec.]
```

```
            [resolved]
        g. Double Buffer for display [skipped]
    5. Hardware Interface
        a. Modify the CLI to include menus  [done]
        b. Fagocytosis of USB demo program [done]
        c. Working on the Proof of Concept for temperature
          i. Velocity calculation [done]
          ii. Realtime graphs [library installed; plplot; basic
              functionality tested]
            I. Test separate  [Done]
            II. Fagocytosis   [Done]
        d. Find the axis of the lattice to find the coil anlge [
            after experimentation, discontinued]
    6. Finalizing
        a. Temperature Interface generic
          i. Proof of concept [Done]
          ii. Import configurations form file [Done]
          iii. Test proper firing [Done]
          iv. Algorithm to find the best candidate for pumping
              energy [Done]
          v. Test the algorithm with 2 dipoles (not even a
              lattice :) [Done]
          vi. Ensure the temperature communication doesn't make
              the program wait [Done]
          vii. Test with a 2x2 lattice [Done]
        b. Improved colour filter based on HSV
          i. implement [Done]
          ii. allow colour picking (debug exisiting problem) [
              Done]
        c. Stricter ellipse detection (rejects most unwanted
            ellipses) [Done]
        d. Temperature Calculation and corresponding proof of
            concept triger [Done]

*/


#include "opencv2/highgui/highgui.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <thread>
#include <mutex>
#include <chrono>
#include <string>

#include <atomic>

using namespace cv;
using namespace std;
```

```
// #include <string.h>
// #include <array>

#define GRAPHS_ENABLED

#ifdef GRAPHS_ENABLED
    #include <plplot/plplot.h>
    #include <plplot/plstream.h>
    plstream *pls=new plstream();
    const char *legline[4];
    int colline[4],styline[4];
    int id1;
    bool plotSqKE=true;
#endif


//Configuration
//#define ATOMIC
// #define MULTI_THREAD_DISPLAY
// #define ATOMIC_DISPLAY
// #define MULTI_THREAD_CAMERA_UPDATE


#define TEMPERATURE_ENABLED


//TODO: Either calculate it at runtime, or allow the user to
    input
//This is the detected angle of the dipole when it is aligned
    with the coil (least energy configuration)
// #define COILANGLE 0
float coilAngle = 90;
#define preciseAngleTol 20
    //This is slightly twisted to explain; it is the difference
        allowed between the atan2 angle and the ellipse angle to
        resolve the mod, if this is not clear, refer to the code
const double version=0.6;
// #define MINANGULARVELOCITY 10000000
int minAngularVelocity=100;

int tempCandidate=0;  //This is the dipole that is accelerated
bool blind=false;           //This is the blind option, meaning
    hardware tracking is turned off
bool invertPush=false;      //This is to invert the moment of
    pushing
bool useCalibration=true;
#ifdef TEMPERATURE_ENABLED
  #ifdef TEMPERATURE_SINGLDIPOLE
    inline void fireElectro(long frame, int id);
  #else
    void fireElectro(int id, int intensity);
```

```
  #endif

  // for USB interface
  extern "C"
  {
    #include "DataTypes.h"
    #include "usbIO.h"
  }
  //
  vector<char> dipolePort;
  vector<char> dipoleBit;
#endif

#ifdef ATOMIC_DISPLAY
  #ifndef MULTI_THREAD_DISPLAY
    #define MULTI_THREAD_DISPLAY
  #endif
#endif

//#if (defined(ATOMIC) || defined(ATOMIC_DISPLAY) || defined(
    MULTI_THREAD_DISPLAY) || defined(MULTI_THREAD_CAMERA_UPDATE))
  //#include<atomic>
//#endif


#define CALIBRATION_ENABLED

#ifdef CALIBRATION_ENABLED
    Mat cameraMatrix;
    // = Mat(3,3,CV_32FC1);
    Mat distCoeffs;
    // = Mat(5,1,CV_32FC1);
#endif

#ifdef ATOMIC
        //#include <atomic>
  atomic<bool> threadsEnabled=false;
#else
  bool threadsEnabled=false;
#endif


#ifdef MULTI_THREAD_DISPLAY
  #ifndef ATOMIC_DISPLAY
    mutex drawnow;
  #endif
#endif

#ifdef ATOMIC_DISPLAY
    atomic<bool> updateDisplayRequested;
    atomic<bool> updateDisplayCompleted;
#endif
```

```
mutex processingImage;
mutex grabbingFrame;

//For computation time
double tCstart,tCdelta,tCend; //time for computation
vector <double> computationTime;

vector <Mat>  buf;
// ///////////////////////
//     Mat_<double> cameraMatrix(3,3), distCoeffs(5,1);
//     cameraMatrix =[ 5.6712925674714052e+02, 0.,
    3.1716566879559707e+02, 0., 5.6556813512152769e+02,
    2.1037221807058236e+02, 0., 0., 1. ];
// ///////////////////////


Mat grabbedFrame;
#ifdef MULTI_THREAD_CAMERA_UPDATE
  atomic<bool> frameGrabbed,frameRequested;
#else
  bool frameGrabbed=false,frameRequested=false;
#endif

#ifdef CALIBRATION_ENABLED
  void getCameraCalibrationParameters()
  {
    char calibrationFile[]="configurations/logitech2";
    FileStorage fs2(calibrationFile, FileStorage::READ);

    // first method: use (type) operator on FileNode.
    // int frameCount = (int)fs2["frameCount"];

    // std::string date;
    // second method: use FileNode::operator >>
    // fs2["calibrationDate"] >> date;

    // Mat cameraMatrix2, distCoeffs2;

    fs2["camera_matrix"] >> cameraMatrix;
    fs2["distortion_coefficients"] >> distCoeffs;


    // cameraMatrix = Mat(3,3,CV_32FC1);
    // distCoeffs = Mat(5,1,CV_32FC1);
    // //TODO: Input them off of a file instead..
    // //Not very certain how the pointer thing is working..found
         this on an implementation on the net
    //   cameraMatrix.ptr<float>(0)[0] = 5.6712925674714052e+02;
    //   cameraMatrix.ptr<float>(0)[1] = 0;
    //   cameraMatrix.ptr<float>(0)[2] = 3.1716566879559707e+02;
    //   cameraMatrix.ptr<float>(1)[0] = 0;
```

```
    //   cameraMatrix.ptr<float>(1)[1] = 5.6556813512152769e+02;
    //   cameraMatrix.ptr<float>(1)[2] = 2.1037221807058236e+02;
    //   cameraMatrix.ptr<float>(2)[0] = 0;
    //   cameraMatrix.ptr<float>(2)[1] = 0;
    //   cameraMatrix.ptr<float>(2)[2] = 1;

    //   distCoeffs.ptr<float>(0)[0]=1.0093652191470437e-01;
    //   distCoeffs.ptr<float>(1)[0]=-3.9155163783030478e-01;
    //   distCoeffs.ptr<float>(2)[0]=-8.1203753896884399e-04;
    //   distCoeffs.ptr<float>(3)[0]=4.7881016467320944e-03;
    //   distCoeffs.ptr<float>(4)[0]=2.8593695994355700e-01;
    cout<<"Camera Calibration Enabled"<<endl<<"Using file: "<<
        calibrationFile<<endl;
    cout<<cameraMatrix<<endl<<distCoeffs<<endl;
  }
#endif
void initializeMultithreadResources()
{
#ifdef MULTI_THREAD_CAMERA_UPDATE
  frameGrabbed=false,frameRequested=false;
#endif

#ifdef ATOMIC_DISPLAY
    updateDisplayRequested=true;
    updateDisplayCompleted=false;
#endif

#ifdef ATOMIC
  //#include <atomic>
  threadsEnabled=false;
#endif
}

Mat srcPreCrop; Mat cimg; Mat src; Mat src_gray; Mat
    srcColorFilter; Mat src_process; Mat srcColorA; Mat srcColorB
    ;Mat drawing;
//////////////
// bool isGood(float val,float expected,float tol)
// {
//   if(abs(val-expected)
// }
//////////////
//the following two functions are from http://www.cs.rit.edu/~ncs
    /color/t_convert.html
//0-360 and 0-1 (originally)
//r g b is also from zero to one
void RGBtoHSV( double r, double g, double b, double *h, double *s
    , double *v )
{
  r/=255;
  g/=255;
  b/=255;
```

```
  double min, max, delta;

  min = MIN( r, MIN(g, b) );
  max = MAX( r, MAX(g, b) );
  *v = max;       // v

  delta = max - min;

  if( max != 0 )
    *s = delta / max;   // s
  else {
    // r = g = b = 0    // s = 0, v is undefined
    *s = 0;
    *h = -1;
    return;
  }

  if( r == max )
    *h = ( g - b ) / delta;   // between yellow & magenta
  else if( g == max )
    *h = 2 + ( b - r ) / delta; // between cyan & yellow
  else
    *h = 4 + ( r - g ) / delta; // between magenta & cyan

  *h *= 60;       // degrees
  if( *h < 0 )
    *h += 360;

  (*h) *= (180.0/360.0);
  (*s) *= 255.0;
  (*v) *= 255.0;
}

void HSVtoRGB( double *r, double *g, double *b, double h, double
    s, double v )
{
  h *= (360.0/180.0);
  s /= 255.0;
  v /= 255.0;

  int i;
  double f, p, q, t;

  if( s == 0 ) {
    // achromatic (grey)
    *r = *g = *b = v;
    return;
  }

  h /= 60;      // sector 0 to 5
  i = floor( h );
```

```
  f = h - i;      // factorial part of h
  p = v * ( 1 - s );
  q = v * ( 1 - s * f );
  t = v * ( 1 - s * ( 1 - f ) );

  switch( i ) {
    case 0:
      *r = v;
      *g = t;
      *b = p;
      break;
    case 1:
      *r = q;
      *g = v;
      *b = p;
      break;
    case 2:
      *r = p;
      *g = v;
      *b = t;
      break;
    case 3:
      *r = p;
      *g = q;
      *b = v;
      break;
    case 4:
      *r = t;
      *g = p;
      *b = v;
      break;
    default:    // case 5:
      *r = v;
      *g = p;
      *b = q;
      break;
  }

  (*r)*=255;
  (*g)*=255;
  (*b)*=255;
}
//////////////////////
float findPrinciple(float val,float modVal)
{
  while(val<0)
    val-=modVal;
  return val;
}

//Think of a clock. I give you two positions on the clock. You've
     to tell me which ones ahead
```

```
/////basically modular arithmetic in some sense
bool IsClockwise(float final, float initial, float modVal)
{
  float delta,deltaA,deltaB;
  delta=final-initial;

  deltaA=delta;
  while(deltaA<0)
    deltaA+=modVal;

  deltaB=-delta;
  while(deltaB<0)
    deltaB+=modVal;

  if(deltaA<deltaB)
    return true;
  else
    return false;
}
//This is to find the shortest distance in two numbers in a
    modular arithmetic system
float shortestDistance(float final, float initial, float modVal)
{
  float delta,deltaA,deltaB;
  delta=final-initial;

  deltaA=delta;
  while(deltaA<0)
    deltaA+=modVal;

  deltaB=-delta;
  while(deltaB<0)
    deltaB+=modVal;

  return (deltaA<deltaB) ? deltaA: deltaB;
}


// int lastBuf=1;
//for the cropping
int cropped = 0;
Point origin;
Rect selection;
bool selectRegion;

// Mat cimg;
Mat_<Vec3b> srcTemp = src;
int thresh = 100;
int max_thresh = 255;
int canny=100;
int centre=30;
```

```
int minorAxis=7;
int majorAxis=35;
int radius=15;
int ellipseTolerance=8;


int mode=0;
double theta=3.14159;

//mode
//0 is screen select
//1 is colour select

RNG rng(12345);

//////////////DIPOLE DETECTION

class dipole
{
public:
  double angle,order; //angle is the angle, order gives a rough
      size of the dipole detected
  int x,y,id;  //centre, id tells where its mapped
  int e1,e2; //index number of ellipse
  static int count[2]; //double buffer
  static int current;
};

int dipole::count[2] = {0};
int dipole::current=0;

//Not using a dynamic array, doesn't matter for now
//TODO: Use a dynamic aray
// array<dipole,500> dipoles;
// array<dipole,500> lastDipoles;
#define DmaxDipoleCount 1000
#define DtemperatureAverageOverPeriod 1
dipole dipoles[2][DmaxDipoleCount];
// Colour read
// Point origin;



///////////////DIPOLE INFORMATION STORAGE IN RAM
bool dipoleRec=false;

class dipoleSkel
{
public:
    double angle;
    int id; //For the dipoleData, this refers to the id of
        seedDipole
```

```
    //in seedDipole, this should refer to the hardware ID
    vector<int> neighour;
    vector<double> neAngle;
    int x,y;
    double instAngularVelocity;
    bool detected;  //stores whether the dipole was detected at
        all
};

//This class is for storing dipole data of a given frame
class dipoleFrame
{
public:
  double time;   //time elapsed since the seed frame
  double order;  //gives the rough size of the dipoles
  int count,velValidCount;    //number of dipole detected in the
      frame, number of dipoles for which inst angular velocity
      could be calculated (essentially, that it was detected in
      two consecutive frames)
  double meanSquaredAngularVelocity,temperature; //mean of
      squares of angular velocities of each of the dipoles
  vector<dipoleSkel> data;
};

// #define DframeBufferLen  5000
dipoleFrame seedDipole;
//NOTE: You have to fix the numbering problem right here.
vector <dipoleFrame> dipoleData;

#define DframeBufferLen 5000
////THIS IS FOR STORING IN FILE
FILE * pFile;
char fileName[50];
////////////////////////////
#ifdef TEMPERATURE_ENABLED
  void getTemperaturePins()
  {

    char temperaturePinsFile[]="configurations/temperaturePins";
    FileStorage fs2(temperaturePinsFile, FileStorage::READ);
    cout<<endl<<"Temperature Enabled, using file: "<<
        temperaturePinsFile<<endl;

    char tempString[5];
    string extractedString;
    dipolePort.clear();
    dipoleBit.clear();

    int iMax;
    fs2["count"]>>iMax;

    // FileNodeIterator it = fs2.begin(), it_end = fs2.end();
```

```
    // int idx = 0;
    // std::vector<uchar> lbpval;

    // // iterate through a sequence using FileNodeIterator
    // for( ; it != it_end; ++it, idx++ )
    // {
    //     // cout << "feature #" << idx << ": ";
    //     cout << "x=" << (int)(*it)["x"] << ", y=" << (int)(*it
        )["y"] << ", lbp: (";
    //     // you can also easily read numerical arrays using
        FileNode >> std::vector operator.
    //     (*it)["lbp"] >> lbpval;
    //     for( int i = 0; i < (int)lbpval.size(); i++ )
    //         cout << " " << (int)lbpval[i];
    //     cout << ")" << endl;
    // }


    for(int i=0;i<iMax;i++)
    {
      sprintf(tempString,"p%d",i);
      fs2[tempString] >> extractedString;
      // cout<<tempString<<" "<<extractedString<<endl;
      if(extractedString.size()>1)
      {
        dipolePort.push_back(extractedString[0]);
        dipoleBit.push_back((extractedString[1]-'0'));
        //Not using dipole pin because obviously char
            corresponding to 0 is not '0'
        cout<<"Dipole "<<i<<" \t -- \t"<<dipolePort[i]<<
            extractedString[1]<<" (Port"<<dipolePort[i]<<" Bit"<<
            extractedString[1]<<")"<<endl;
      }
      else
      {
        dipolePort.push_back('Z');
        dipoleBit.push_back(9);
      }
    }

  }
#endif



//////////////////////
float candidateAptitude( int id)
{
  // int candidate=posPhysicalToDetected(id);
  int cf=dipoleData.size()-1;
  int candidate=id;
```

```
  float distanceFromCoil=shortestDistance(dipoleData[cf].data[
      candidate].angle,coilAngle,360);
  // if((dipoleData[cf].data[tempCandidate].angle - COILANGLE) >
      0)
  if(IsClockwise(dipoleData[cf].data[candidate].angle,coilAngle
      ,360))
  {
    if (dipoleData[cf].data[candidate].instAngularVelocity>0) //
        if it is going in the opposite direction
    {
      if (invertPush)
        // fireElectro(cf,tempCandidate);
        return distanceFromCoil;
    }
    else
    {
      if(!invertPush)
         // fireElectro(cf,tempCandidate);  //to increase speed,
             because the force will be towards the coil
        return distanceFromCoil;
    }


  }
  else  //if the angle is negaative, then
  {
    if (dipoleData[cf].data[candidate].instAngularVelocity<0) //
        if it is going twoards the coil
    {
      if(invertPush)
        // fireElectro(cf,tempCandidate);
        return distanceFromCoil;
    }
    else
    {
      if(!invertPush)
        // fireElectro(cf,tempCandidate);
        return distanceFromCoil;
    }
      // fireElectro(cf);
  }
  return 0; //this means the given dipole can't be used to pump
      in energy
}


//////////

int posPhysicalToDetected(int phyID)
{
  for(int i=0;i<seedDipole.data.size();i++)
  {
```

```
    if(seedDipole.data[i].id==phyID)
      return i;
  }
  cout<<endl<<"INVALID physical ID";
  return 0;
}

// This is a colour filter for improving accuracy
  // 20, 28, 41 [dark]
  // TODO: Allow the user to select the colour
  // Scalar colorB=Scalar(245,245,10);
  // Scalar colorB=Scalar(126,88,47);
  // Scalar colorA=Scalar(10,245,245);

  // HSV



  //This is experimental :P
  Scalar colorA((float)103.953, (float)151.59, (float)217);

  //The following's from GIMP
  // Scalar colorA((float)206*(360/180),74*(255/100),83*(255/100)
      );

  // int colorATol=30;
  // int colorBTol=35;
  // int brightInv=10;  //this is to increase the brightness
      after processing
  // H: 0 - 180, S: 0 - 255, V: 0 - 255

  int hueTol=20;
  int valueTol=193;
  int saturationTol=85;

  int colorATol=255;
  int colorBTol=255;
  int brightInv=74;  //this is to increase the brightness after
      processing

//
  const char* source_window = "Source";
  const char* filter_window = "Color Filter";
  const char* settings_window="Settings";


//////////TIMING
  long t,tLast,tickWhenGrabbed;
  double deltaT;
static void onMouse( int event, int x, int y, int, void* )
{
  if(mode==0)
```

```
{
  if( selectRegion )
  {
      selection.x = MIN(x, origin.x);
      selection.y = MIN(y, origin.y);
      selection.width = std::abs(x - origin.x);
      selection.height = std::abs(y - origin.y);

      selection &= Rect(0, 0, src.cols, src.rows);
  }

  switch( event )
  {
  case CV_EVENT_LBUTTONDOWN:
      cropped=0;
      origin = Point(x,y);
      selection = Rect(x,y,0,0);
      selectRegion = true;
      break;
  case CV_EVENT_LBUTTONUP:
      cropped=0;
      selectRegion = false;
      if( selection.width > 0 && selection.height > 0 )
          cropped = -1;
      break;
  }
}
else if(mode==1)
{
  switch( event )
  {
  case CV_EVENT_LBUTTONUP:
      cout<<"Color A is currently (HSV) "<<endl<<(float)colorA.
          val[0]<<endl<<(float)colorA.val[1]<<endl<<(float)
          colorA.val[2]<<endl;
      cout<<endl<<"x="<<x<<",y="<<y<<endl;
      // colorA=Scalar(srcPreCrop.at<Vec3b>(y,x)[0],srcPreCrop.
          at<Vec3b>(y,x)[1],srcPreCrop.at<Vec3b>(y,x)[2]);
      //THIS IS RIGHT, it had to be y,x and not x,y!!! damn..
      RGBtoHSV((int)srcPreCrop.at<Vec3b>(y,x)[2],(int)
          srcPreCrop.at<Vec3b>(y,x)[1],(int)srcPreCrop.at<Vec3b
          >(y,x)[0],&(colorA.val[0]),&(colorA.val[1]),&(colorA.
          val[2]));
      cout<<"BGR colours are "<<(int)srcPreCrop.at<Vec3b>(y,x)
          [2]<<","<<(int)srcPreCrop.at<Vec3b>(y,x)[1]<<","<<(
          int)srcPreCrop.at<Vec3b>(y,x)[0]<<endl;
      // printf("RGB colours are %f %f %f",srcPreCrop.at<Vec3b
          >(y,x)[0],srcPreCrop.at<Vec3b>(y,x)[1],srcPreCrop.at<
          Vec3b>(y,x)[2]);
      cout<<"Color A's been changed to "<<endl<<(float)colorA.
          val[0]<<endl<<(float)colorA.val[1]<<endl<<(float)
          colorA.val[2]<<endl;
```

```
        break;
    // case CV_EVENT_RBUTTONUP:
    //     cout<<x<<","<<y<<endl;
    //     colorB=Scalar(src.at<Vec3b>(y,x)[0],src.at<Vec3b>(y,x)
        [1],src.at<Vec3b>(y,x)[2]);
    //     cout<<"Color B's been changed to "<<endl<<colorB.val
        [0]<<endl<<colorB.val[1]<<endl<<colorB.val[2]<<endl;
    //     break;
    }
  }
}

#ifdef __GNUC__
        void tGrabFrame(VideoCapture&& capture)
#else
        void tGrabFrame(VideoCapture& capture)
#endif
{
  while(threadsEnabled)
  {

    capture>>grabbedFrame;
    if(frameRequested)
    {
      tickWhenGrabbed=getTickCount();
        grabbedFrame.copyTo(srcPreCrop);
        frameGrabbed=true;
    }

        //buf.push_back(frameGrabbed);

    // if(processingImage.try_lock())
    // {
    //   buf.copyTo(srcPreCrop);
    //   processingImage.unlock();
    // }
  }

}

void updateDisplay()
{
  if(!drawing.empty())
    imshow("Contours", drawing );
  if(!cimg.empty())
    imshow("Debug", cimg);
  if(!src_gray.empty())
    imshow( filter_window, src_gray);
  if(!srcPreCrop.empty())
    imshow( source_window, srcPreCrop );

}
```

```
#ifdef MULTI_THREAD_DISPLAY
  #ifndef ATOMIC_DISPLAY
    void tUpdateDisplay()
    {
      while(threadsEnabled)
      {
        drawnow.lock(); //this is to ensure it updates only once
            the processing has been done and not repetatively the
             same frame
        drawnow.unlock();

        updateDisplay();

      }
    }
  #else
    void tAtomicDisplay()
    {
      while(threadsEnabled)
      {

          // && updateDisplayCompleted==false)
        if(updateDisplayRequested)
        {
          updateDisplay();
          // this_thread::sleep_for(chrono::milliseconds(10));
          // updateDisplayCompleted=true;
          updateDisplayRequested=false;
        }
                    // else
          // this_thread::sleep_for(chrono::milliseconds (1));
                         // this_thread::yield();

      }
    }
  #endif
#endif
#ifdef GRAPHS_ENABLED

    void clearGraph()
    {
      pls->col0(1);
      // cout<<"1"<<endl;
      double xmin2d = -2.5;
      double xmax2d =  2.5;
      double ymin2d = -2.5;
      double ymax2d =  4.0;
      // pls->wind( 0.0, 1.0, 0.0, 1.0 );
      // pls->env(xmin2d, xmax2d, ymin2d, ymax2d, 0, -2);
      pls->adv(1);
      pls->clear();
```

```
pls->vpor( 0.0, 1.0, 0.0, 1.0 );
pls->wind( -2.5, 2.5, -3.0, 3.0 );

double basex = 2.0;
double basey = 4.0;
double height = 3.0;
double xmin = 0.0;
double xmax = 10.0;
double ymin = 0.0;
double ymax = 1000.0;
double zmin = 0.0;
double zmax = 360.0;
double alt = 45.0;
double az = 30.0;
double side = 1;
pls->w3d(basex, basey, height, xmin, xmax, ymin, ymax, zmin
    , zmax, alt, az);
pls->box3( "bnstu", "Dipole Count", 0.0, 0,
        "bnstu", "Frame Count", 0.0, 0,
        "bcdmnstuv", "Angular Position", 0.0, 4 );



// This is for selecting the second view
// Following are to get the points in the right location!
pls->adv(2);
pls->clear();
pls->vpor( 0.0, 1.0, 0.0, 1.0 );
pls->wind( -2.5, 2.5, -3.0, 3.0 );
// pls->env(xmin2d, xmax2d, ymin2d, ymax2d, 0, -2);
// pls->wind( 0.0, 1.0, 0.0, 1.0 );
zmin=-360;
zmax=360;
pls->w3d(basex, basey, height, xmin, xmax, ymin, ymax, zmin
    , zmax, alt, az);
pls->box3( "bnstu", "Dipole Count", 0.0, 0,
        "bnstu", "Frame Count", 0.0, 0,
        "bcdmnstuv", "Angular Velocity", 0.0, 4 );

pls->adv(3);


//the second false is the one!

legline[0] = "total energy";                    // pens
    legend
legline[1] = "not assigned";
legline[2] = "not assigned";
legline[3] = "not assigned";

pls->stripc( &id1, "bcnst", "bcnstv",
```

```
          0, 100, 0.3, 0, 10,
          0, 0,
          true, false,
          1, 3,
          colline, styline, legline,
          "t", "", "Average Kinetic Energy" );
    }
#endif

int process(VideoCapture& capture)
{

  #ifdef GRAPHS_ENABLED
    styline[0] = colline[0] = 2;      // pens color and line
        style
    styline[1] = colline[1] = 3;
    styline[2] = colline[2] = 4;
    styline[3] = colline[3] = 5;

    pls->init();
    pls->ssub( 1, 3 );
    // pls->adv(1);
    cout<<endl<<"Initializing the interface for graphing"<<endl;
    clearGraph();

  #endif


  #ifdef TEMPERATURE_ENABLED
      vInitUSB();
  #endif

  /// Create Window
  namedWindow( source_window, WINDOW_AUTOSIZE );
  setMouseCallback( "Source", onMouse, 0 );
  //Show the filtered image too

  namedWindow( filter_window, WINDOW_AUTOSIZE );

  //Show the settings window

  namedWindow(settings_window,WINDOW_AUTOSIZE  | CV_GUI_NORMAL);
  createTrackbar( "Min Ang Vel Sq", settings_window, &
      minAngularVelocity, 1000000, 0 );
  // H: 0 - 180, S: 0 - 255, V: 0 - 255
  createTrackbar( "Hue Tolerance", settings_window, &hueTol, 180,
       0 );
  createTrackbar( "Saturation Tolerance", settings_window, &
      saturationTol, 255, 0 );
  createTrackbar( "Value Tolerance", settings_window, &valueTol,
      255,0 );
```

```
  // createTrackbar( "ColorB Tolerance", settings_window, &
      colorBTol, 256, 0 );
  createTrackbar( "Minor Axis", settings_window, &minorAxis, 100,
       0 );
  createTrackbar( "Major Axis", settings_window, &majorAxis, 200,
       0 );
  createTrackbar( "Radius", settings_window, &radius, 200, 0 );
  createTrackbar( "Ellipse Tolerance", settings_window, &
      ellipseTolerance, 200, 0 );

  createTrackbar( "Brightness Inverse",settings_window, &
      brightInv, 255, 0);
  createTrackbar( "Canny (Hough)", settings_window, &canny, 200,
      0 );
  createTrackbar( "Centre (Hough)", settings_window, &centre,
      200, 0 );
  // createTrackbar( "Theta", settings_window, &thetaD, 3.141591,
       0 );

  /// Show in a window
  namedWindow( "Contours", WINDOW_AUTOSIZE );
  namedWindow( "Debug", WINDOW_AUTOSIZE );


  ////Voodoo intializations
  // dipoles[0][0].current=0;
  // dipoles[0][0].count[0]=0;
  // dipoles[0][0].count[1]=0;
  /// Load source image
  // src = imread( argv[1], 1 );

  // std::string arg = argv[1];
///////////////

  // cout<<capture.get(CV_CAP_PROP_FRAME_HEIGHT);
  // capture.set(CV_CAP_PROP_FRAME_HEIGHT,1080);
  // capture.set(CV_CAP_PROP_FRAME_WIDTH,1920);

  // capture.set(CV_CAP_PROP_FRAME_HEIGHT,720);
  // capture.set(CV_CAP_PROP_FRAME_WIDTH,1280);

  capture.set(CV_CAP_PROP_FRAME_HEIGHT,480);
  capture.set(CV_CAP_PROP_FRAME_WIDTH,640);

  thread t1(tGrabFrame,capture);
  #ifdef MULTI_THREAD_DISPLAY
    #ifndef ATOMIC_DISPLAY
      thread t2(tUpdateDisplay);
    #else
      thread t2(tAtomicDisplay);
    #endif
  #endif
```

```
for(;;)
{
  // processingImage.lock();
  //if the video feed has over 1 frame

  // if(buf.size()>1)
  // {
  //   buf.erase(buf.begin()); //remove the oldest frame
  //   srcPreCrop=buf[buf.size()-1]; //grab the latest frame
  // }

  // frameGrabbed.copyTo(srcPreCrop);
      frameRequested=true;

  if(frameGrabbed==true && !srcPreCrop.empty())
              // #ifdef ATOMIC_DISPLAY
              //      // if(updateDisplayCompleted)
              // #endif

  {
    frameRequested=false;     //this is so that the frame is
        not processed unless required
        frameGrabbed=false;   //this is so that we know the
            next time a frame is grabbed

    #ifdef MULTI_THREAD_DISPLAY
      // drawnow=true;
      #ifndef ATOMIC_DISPLAY
        drawnow.lock();
      #endif
    #endif

    {  //IMAGE CAPTURE and CROP
      // capture>>srcPreCrop;
      ////////////////COMPUTATION TIME CALCULATION
      tCstart=getTickCount();
      ////////////
      tLast=t;
      // t=getTickCount()/getTickFrequency();   //This is give
          time in seconds
      // t=getTickCount();
      t=tickWhenGrabbed;
      deltaT=(t-tLast)/getTickFrequency();

      if(dipoleRec)
      {
        //if this is not the last frame, add a frame
        dipoleData.push_back(seedDipole);
```

```
        //This is to avoid overflows
        // if (dipoleData.size()>DframeBufferLen)
          // dipoleData.erase(dipoleData.begin());
        //for the last frame
        dipoleData[dipoleData.size()-1].time=dipoleData[
            dipoleData.size()-2].time+deltaT;
        dipoleData[dipoleData.size()-1].count=0;  //No dipoles
            found before analysis!
        dipoleData[dipoleData.size()-1].
            meanSquaredAngularVelocity=0; //Initially zero
        dipoleData[dipoleData.size()-1].velValidCount=0;
                        //This is to set the number of dipoles
             for which velocity was calculated to zero. Very
            important
      }

      // long cfInit=dipoleData.size()-1;  //last frame

      // //test for current frame
      // if(dipoleData[cfInit].time!=t)
      // {
      //   //if this is not the last frame, add a frame
      //   dipoleData.push_back(seedDipole);
      //   //This is to avoid overflows
      //   if (dipoleData.size()>DframeBufferLen)
      //     dipoleData.erase(dipoleData.begin());
      //   //for the last frame
      //   cfInit=dipoleData.size()-1;
      //   dipoleData[cfInit].time=t;
      // }



      if(srcPreCrop.empty())
      {
        cout<<"Didn't get an image";
        break;
      }
      if(!cropped==0)
      {
        src=srcPreCrop(selection);
      }
      else
        src=srcPreCrop;

      #ifdef CALIBRATION_ENABLED
        if(useCalibration)
        {
          Mat temp=src.clone();
          undistort(temp, src, cameraMatrix, distCoeffs);
        }
      #endif
```

```
  // imshow( source_window, srcPreCrop );

}

//////////////////////
{//COLOR FILTER
  //Input src, output src_gray
  Scalar lowerBound;
  Scalar upperBound;
  Mat srcCloneTemp=src.clone();


  cvtColor(src,src,CV_BGR2HSV);
  // lowerBound = colorA-Scalar::all(colorATol);
  // upperBound = colorA+Scalar::all(colorATol);
  upperBound=colorA+Scalar((float)hueTol,(float)
      saturationTol,(float)valueTol);
  lowerBound=colorA-Scalar((float)hueTol,(float)
      saturationTol,(float)valueTol);

  // Now we want a mask for the these ranges
  inRange(src,lowerBound,upperBound, srcColorA);

  // lowerBound = colorB-Scalar::all(colorBTol);
  // upperBound = colorB+Scalar::all(colorBTol);
  // We do it for both the colours
  // cvtColor(src,)
  inRange(src,lowerBound,upperBound, srcColorB);

  // Now we create a combined filter for them
  // addWeighted(srcColorA, 1, srcColorB, 1, 0,
      srcColorFilter);
  addWeighted(srcColorA, 1, srcColorB, 0, 0, srcColorFilter
      );

  cvtColor(src,src,CV_HSV2BGR);
  // cvtColor(srcColorA,srcColorA,CV_HSV2BGR);
  /// Convert image to gray
  cvtColor( src, src_process, COLOR_BGR2GRAY );

  /// Now keep only the required areas in the image
  // // // multiply(src_process,srcColorFilter,src_gray,1);



  //Change the following to use greyscaled output
  src_gray=srcColorFilter.mul(src_process/brightInv);
  // src_gray=srcColorFilter;

  // // // src_gray=srcColorFilter;

  // NOw blur it
```

```
  blur( src_gray, src_gray, Size(3,3) );

  // imshow( filter_window, src_gray);
}

////////////////////////////

// BLANK PROCESSING
// medianBlur( src, src, 5 );
// cvtColor( src, src_gray, COLOR_BGR2GRAY );

// // // blur( src_gray, src_gray, Size(3,3) );

////////////////////////////
// This is contour Detection
////////////////////////////////
Mat threshold_output;
vector<vector<Point> > contours;
vector<Vec4i> hierarchy;

/// Detect edges using Threshold
threshold( src_gray, threshold_output, thresh, 255,
    THRESH_BINARY );
/// Find contours
findContours( threshold_output, contours, hierarchy,
    RETR_TREE, CHAIN_APPROX_SIMPLE, Point(0, 0) );

/// Find the rotated rectangles and ellipses for each
    contour
vector<RotatedRect> minRect( contours.size() );
vector<RotatedRect> minEllipse( contours.size() );

for( size_t i = 0; i < contours.size(); i++ )
   { minRect[i] = minAreaRect( Mat(contours[i]) );
     if( contours[i].size() > 5 )
       { minEllipse[i] = fitEllipse( Mat(contours[i]) ); }
   }

for( size_t i = 0; i< minEllipse.size(); i++ )
   {
    //You can add aditional conditions to eliminate
        detected ellipses
    float minor = (minEllipse[i].size.height<minEllipse[i].
        size.width)?minEllipse[i].size.height:minEllipse[i
        ].size.width;
    float major = (minEllipse[i].size.height>minEllipse[i].
        size.width)?minEllipse[i].size.height:minEllipse[i
        ].size.width;
    bool good=false;

    if(minor/major<1.3 && minor/major>0.7)  //if close
        enough to a circle
```

```
    {
      if((abs(minor-radius)<ellipseTolerance) && (abs(major
          -radius)<ellipseTolerance))
        good=true;  //VALID CIRCLE DETECTED
    }
    else //IF its an ellipse then
    {
      if(abs(minor-minorAxis)<ellipseTolerance && abs(major
          -majorAxis)<ellipseTolerance)
        good=true;  //VALID ELLIPSE FOUND
    }
    // if(
    //   !(
    //     (
    //     (minEllipse[i].size.height>minMinorAxis &&
        minEllipse[i].size.width>minMinorAxis)
    //     &&
    //     (minEllipse[i].size.height<maxMajorAxis &&
        minEllipse[i].size.width<maxMajorAxis)
    //     )
    //   )

    //   )
    if(!good)
    {
      // minEllipse[i]=RotatedRect(Point2f(0,0),Point2f
          (0,0),0);
      minEllipse.erase(minEllipse.begin()+i--);
    }
   }



//////////////////////////
//  Dipole Detection Algorithm
//////////////////////////////////////
// This detects form the ellipses detected, the dipoles!!
// This doesn't act on the dipoles itself!!!
// You yourself got confused about this :D
//////////////////////////
vector<bool> detected (minEllipse.size(),false);

int k = !(dipoles[0][0].current);
dipoles[0][0].current=k;
dipoles[0][0].count[k]=0; //this count is different from
    that used in frames, they correspond to a subset of
    these which are position wise close enough to the seed
    frame

// dipolesA[0].lastcount=0;
for (int i=0; i<minEllipse.size();i++)
{
```

```
        if(detected[i]==false)  //This detected corresponds to
            having been paired earlier
        {
          for (int j=0; j<minEllipse.size();j++)
          {
            if((i!=j) && detected[j]==false)  //This is so that
                you don't test with yourself and with others that
                 got paired
            {
                // if(abs(minEllipse[i].angle-minEllipse[j].angle
                    )< 15)  //if the orientation matches (less
                    than 5 degrees), then
                {
                  //Find the distance between minEllipses

                  Point differenceVector=Point(minEllipse[i].
                      center.x - minEllipse[j].center.x,
                      minEllipse[i].center.y - minEllipse[j].
                      center.y);  //find the difference vector
                  double distanceSquared=differenceVector.x*
                      differenceVector.x + differenceVector.y*
                      differenceVector.y; //find the distance
                      squared

                  //Find the major axis length
                  double majorAxis=MAX( MAX(minEllipse[i].size.
                      width, minEllipse[i].size.height) , MAX(
                      minEllipse[j].size.width, minEllipse[j].
                      size.height)); //find the max dimension

                  //The ratio is the ratio between the distance
                      between the ellipse and the small circle
                      and the length of the major axis
                  double errorPlusOne = distanceSquared /
                      ((11.348/30)*(11.348/30)*majorAxis*
                      majorAxis) ; //now to compare, just divide
                      and see if it's close enough to one

                  if (errorPlusOne>0.5 && errorPlusOne<2)  //if
                      the error is small enough, then its a match
                  {
                      // This is to ensure these don't get paired
                      detected[i]=true;
                      detected[j]=true;

                      // this is collection of the final result
                      int c=dipoles[k][0].count[k]++; //dont get
                          confused, count is static, so even
                          dipoles[0][0] would've worked, ro for
                          that matter, any valid index
```

```
// Note the ++ is after because the count
    is always one greater than the index of
     the last element!


// dipoles[k][c].angle=(minEllipse[i].angle
     + minEllipse[j].angle)/2.0;
// dipoles[k][c].angle=(minEllipse[i].angle
    );

//We're using two shapes, one ellipse and
    one circle.
RotatedRect largerEllipse =  ( MAX(
    minEllipse[i].size.width, minEllipse[i
    ].size.height) > MAX(minEllipse[j].size
    .width, minEllipse[j].size.height)  )?
    minEllipse[i]:minEllipse[j];
RotatedRect smallerEllipse =  ( MAX(
    minEllipse[i].size.width, minEllipse[i
    ].size.height) <= MAX(minEllipse[j].
    size.width, minEllipse[j].size.height)
     )?minEllipse[i]:minEllipse[j];


//CENTRE BASED ANGLE ESTIMATION
// double dCenterX=largerEllipse.center.x-
    smallerEllipse.center.x;
// double dCenterY=largerEllipse.center.y-
    smallerEllipse.center.y;

// dipoles[k][c].angle=((180/3.1415926535)*
    atan2(dCenterY,dCenterX)) + 180;
//////////////////////////////

//UNCOMMENT THIS PART FOR THE OLD ANGLE
    ESTIMATOR
// dipoles[k][c].angle=(largerEllipse.angle
    );

//Now we use the circle to remove the mod
    180 problem and get the complete 360
    degree position
// if((smallerEllipse.center.y -
    largerEllipse.center.y) < 0)
//   dipoles[k][c].angle+=180;
//////////////////////


////////////////////////////////////

//THIS IS ATAN ASSISTED RESOLOLVING ELLIPSE
     ANGLE
```

```
                    //This is because the dipole angle is more
                        accurate!
                    double preciseAngle=(largerEllipse.angle);

                    //This is calculated using the centres of
                        the ellipse and circle

                    double dCenterX=largerEllipse.center.x-
                        smallerEllipse.center.x;
                    double dCenterY=largerEllipse.center.y-
                        smallerEllipse.center.y;

                    //IN CODE NOTES:
                    //ELLIPSE is zero/180 when straight
                    //atan2 gives slope of the line, above the
                        axis is positive, below is negative

                    //The first + 180 is because atan2 returns
                        between plus and minus pi,
                    /////and the minus ninty is because the
                        ellipse is perpendicular to the line
                        joining centres of the two ellipses (
                        ellipse and circle)
                    double roughAngle=(((180/3.1415926535)*
                        atan2(dCenterY,dCenterX)) + 180);

                    //THIS DOESN'T GIVE PROPER RESULTS!!
                    // //THIS IS INTERESTING..
                    // // if (roughAngle>0 && roughAngle<180)
                    // if(roughAngle>350 && preciseAngle<10)
                    // {
                    //   // preciseAngle+=180;
                    //   ;
                    //   //do NOTHING!
                    //   //don't remove this bbecause there are
                         else cases also!
                    // }
                    // else if(roughAngle<10 && preciseAngle
                        >170)
                    // {
                    //   preciseAngle+=180;
                    // }
                    // else
                    // {
                    //   double equivalentAngle=roughAngle;  //
                        will always be between 0 and 180
                    //   //if the precise angle is anyway close
                         enough then dont do anything, else
                    //   //The commented didn't work
                    //   // if(( ((int)abs(preciseAngle-
                        equivalentAngle)) % 360)>
                        preciseAngleTol && ( ((int) abs((
```

```
    preciseAngle+180)-equivalentAngle)) %
    360)<preciseAngleTol)
//   if(abs(preciseAngle-equivalentAngle)>
    preciseAngleTol && abs((preciseAngle
    +180)-equivalentAngle)<=preciseAngleTol
    )
//   {
//     preciseAngle += 180;
//   }
// }
//THIS IS MATH POWER (actually miniscule
    manifestation of math's power)
if((shortestDistance(roughAngle,
    preciseAngle,360)-shortestDistance(
    roughAngle,preciseAngle+180,360))>5)
  preciseAngle+=180;

dipoles[k][c].angle=findPrinciple(
    preciseAngle,360);
// dipoles[k][c].angle=roughAngle;
//////////////////


// dipoles[k][c].x=(minEllipse[i].center.x
    + minEllipse[j].center.x)/2.0;
// dipoles[k][c].y=(minEllipse[i].center.y
    + minEllipse[j].center.y)/2.0;

dipoles[k][c].order=MAX(largerEllipse.size.
    height, largerEllipse.size.width);

dipoles[k][c].x=largerEllipse.center.x; //(
    minEllipse[i].center.x + minEllipse[j].
    center.x)/2.0;
dipoles[k][c].y=largerEllipse.center.y; //(
    minEllipse[i].center.y + minEllipse[j].
    center.y)/2.0;



dipoles[k][c].e1=i; //don't know why this
    is required
dipoles[k][c].e2=j;



// A NEW DIPOLE HAS BEEN DETECTED (
    HOPEFULLY)
// TIME TO PUT IT IN PLACE (IF ASKED TO)
//////////////THIS IS FOR RECORDING/SAVING
    THE DIPOLE MOVEMENT////////////
if (dipoleRec==true)
```

```
                    {

                      long cf=dipoleData.size()-1;  //last
                          frame


                      for(int q=0;q<seedDipole.data.size();q++)
                      {
                        //This is to test which dipole belongs
                            where in accordance with the
                            seedDipole frame
                        // if(MAX(abs(seedDipole.data[q].x -
                            dipoles[k][c].x), abs(seedDipole.
                            data[q].y - dipoles[k][c].y)) < (
                            seedDipole.order/2.0) )
                        //Or you could use the last fraame for
                            this
                        //CAVEAT: YOU MAY THINK WITH THE LAST
                            FRAME, THERE MIGHT ALREADY HAVE
                            BEEN MISSES!! BUT THAT'S ALRIGHT,
                            EVERY LAST FRAME COMES FORM THE
                            SEED FRAME, SO AT WORST, IT WILL BE
                        //MATCHED TO THE DIPOLE IN THE SEED
                            FRAME
                        if(
                          (MAX(abs(dipoleData[cf-1].data[q].x -
                               dipoles[k][c].x), abs(dipoleData
                              [cf-1].data[q].y - dipoles[k][c].
                              y)) < (dipoleData[cf-1].order
                              /4.0) )
                          &&
                          (dipoleData[cf].data[q].detected==
                              false)
                          )
                        {
                          dipoles[k][c].id=q;
                          // dipoleData.data[q] = dipoles[k][c]
                          //TODO: Make a function for
                              converting
                          dipoleData[cf].data[q].id=q;
                          dipoleData[cf].data[q].x=dipoles[k][c
                              ].x; //Copy the relavent data
                              from the dipole data collected
                              into the temp dipole
                          dipoleData[cf].data[q].y=dipoles[k][c
                              ].y;
                          dipoleData[cf].data[q].angle=dipoles[
                              k][c].angle;
                          dipoleData[cf].count+=1;
                          if(cf==0)
                            dipoleData[cf].data[q].
                                instAngularVelocity=0;
```

```
else
{
  if(dipoleData[cf-1].data[q].
      detected==true)
  {
    double deltaAngle=(dipoleData[cf
        ].data[q].angle - dipoleData[
        cf-1].data[q].angle);
    if(abs(deltaAngle)>300)  //eg.
        359 - 2
    {
      if(dipoleData[cf].data[q].angle
          <30) //roughly speaking, it
           couldn't couldn't have
          crossed 20!
      {
        deltaAngle=(dipoleData[cf].
            data[q].angle+360)-
            dipoleData[cf-1].data[q].
            angle;
      }
      else  //the other one must be
          close to zero!
      {
        deltaAngle=dipoleData[cf].
            data[q].angle-(dipoleData
            [cf-1].data[q].angle+360)
            ;
      }
    }
    dipoleData[cf].data[q].
        instAngularVelocity=
        deltaAngle/deltaT;
    dipoleData[cf].velValidCount+=1;
  }
  else
  {
    dipoleData[cf].data[q].
        instAngularVelocity=0; //this
         is NOT TRUE! but doen't
        matter, because the usual
        analysis will use angle vs
        time
    //this is used for angular
        velocity caclulation, in
        which it will be added, but
        not counted while dividing...
        so it is harmless
  }
}
```

```
                        // if(cf>1)
                          // if(dipoleData[cf-1].data[q].
                              detected)
                        dipoleData[cf].
                            meanSquaredAngularVelocity += (
                            dipoleData[cf].data[q].
                            instAngularVelocity*dipoleData[cf
                            ].data[q].instAngularVelocity) ;
                             //Add the sqr of inst angular
                            velocity, averaging is done later

                        dipoleData[cf].data[q].detected=true;
                               //This is true only when the
                            dipole's

                        dipoleData[cf].order=dipoles[k][c].
                            order; //This is bad programming
                            ..i should average, but doens't
                            matter
                        //Now that it has matched, terminate
                            the loop
                        q=seedDipole.data.size();
                      }
                    }


                  }


              }
              // magnitude(differenceVector.x,
                  differenceVector.y,distance);

              // point positionVector ((minEllipse[i].x +
                  minEllipse[j].x)/2.0,(minEllipse[i].y +
                  minEllipse[j].y)/2.0);

            }
          }
        }
      }
    }
    //////////////AFter the frame has been processed, evaluate
        physical quantities of interest
    if(dipoleRec==true)
    {
      long cf=dipoleData.size()-1;  //last frame
      if(dipoleData[cf].velValidCount>0)
        dipoleData[cf].meanSquaredAngularVelocity/=dipoleData[
            cf].velValidCount;
      //else it would be zero, the meanSquaredAngularVelocity
```

```
// dipoleData[cf].meanSquaredAngularVelocity=sqrt(
    dipoleData[cf].meanSquaredAngularVelocity);

int temperatureFrameCount=0;
double temperatureFrame=0;
for(int k=cf;k>0;k--)
{
  double dTime=(dipoleData[cf].time-dipoleData[k].time);
  // cout<<dTime;
  if(dTime>=0 && dTime<DtemperatureAverageOverPeriod)
  {
    temperatureFrame+=dipoleData[k].
        meanSquaredAngularVelocity;
    // cout<<temperatureFrame;
    temperatureFrameCount++;
  }
  else
    k=0;  //terminate the loop
}
if(temperatureFrameCount>0)
  temperatureFrame/=temperatureFrameCount;
else
  temperatureFrame=0;
dipoleData[cf].temperature=temperatureFrame;


#ifdef TEMPERATURE_ENABLED

  //If you want to put pid, put it here..
  #ifndef TEMPERATURE_SINGLDIPOLE
  if(dipoleData[cf].temperature<(minAngularVelocity))
  {
    static int lastFireIntensity=0;
    static long lastFireTime=0;
    int bestCandidate=0;    //Gotto pump some dipole! :P
    float lastMinDist=360;  //This is the how far it is
        from the coil, can't be greater than 180
    for(int i=0;i<dipoleData[cf].data.size();i++)
    {
      float minDist=candidateAptitude(i);
      if(minDist<lastMinDist)
      {
        lastMinDist=minDist;
        bestCandidate=i;
      }
    }

    int dipoleToFire=seedDipole.data[ dipoleData[cf].data
        [bestCandidate].id ].id;
    // fireElectro(dipoleToFire,abs(dipoleData[cf].data[
        bestCandidate].instAngularVelocity/10));
```

```
            cout<<( (tickWhenGrabbed - lastFireTime )/
                getTickFrequency() ) <<","<<(((float)
                lastFireIntensity)/1000)<<endl;
            if( ( (tickWhenGrabbed - lastFireTime )/
                getTickFrequency() )  > (((float)
                lastFireIntensity)/1000) )
            {
              lastFireIntensity=30;
              fireElectro(dipoleToFire,lastFireIntensity);
              lastFireTime=tickWhenGrabbed;
            }
          }
          #endif

          #ifdef TEMPERATURE_SINGLDIPOLE
          //THE SINGLE DIPOLE ALGORITHM
          if(!blind)
          {
            if(dipoleData[cf].temperature<(minAngularVelocity))
            {
              int candidate=posPhysicalToDetected(tempCandidate);

              // if((dipoleData[cf].data[tempCandidate].angle -
                  COILANGLE) > 0)
              if(IsClockwise(dipoleData[cf].data[candidate].angle
                  ,coilAngle,360))
              {
                if (dipoleData[cf].data[candidate].
                    instAngularVelocity>=0) //if it is going in
                    the opposite direction
                {
                  if (invertPush)
                    fireElectro(cf,tempCandidate);
                }
                else
                {
                  if(!invertPush)
                     fireElectro(cf,tempCandidate);  //to
                         increase speed, because the force will
                         be towards the coil
                }


              }
              else  //if the angle is negaative, then
              {
                if (dipoleData[cf].data[candidate].
                    instAngularVelocity<=0) //if it is going
                    twoards the coil
                {
                  if(invertPush)
                    fireElectro(cf,tempCandidate);
```

```
            }
            else
            {
              if(!invertPush)
                fireElectro(cf,tempCandidate);
            }
              // fireElectro(cf);
          }
          // fireElectro();
        }
      }
      else
        fireElectro(cf,tempCandidate);
      #endif
    #endif
  }
  //////////////
  #ifdef GRAPHS_ENABLED
    if(dipoleRec==true)
    {
      // cout<<endl<<"DID SOMETHING";
      static long cfRe=0;
      long cf=dipoleData.size()-1;  //last frame
      long t=(cf-cfRe);
      if(t>=1000)
      {
        cfRe=cf;
        clearGraph();
        t=0;
        // pls->bop();
        // pls->adv(1);
        // pls->clear();
        // pls->adv(2);
        // pls->clear();
      }

      for(int i=0;i<dipoleData[cf].count;i++)
      {
        if(dipoleData[cf].data[i].detected)
        {
          pls->adv(1);

          pls->vpor( 0.0, 1.0, 0.0, 1.0 );
          pls->wind( -2.5, 2.5, -3.0, 3.0 );
          pls->w3d(2,4,3,0,10,0,1000,0,360,45,30);
          double x = dipoleData[cf].data[i].id;
          double z = dipoleData[cf].data[i].angle;
          // double x = i;
          // double z=i;
          double y = t;
          pls->col0((i+1)%10);
          pls->poin3(1,&x, &y, &z,1);
```

```
            pls->adv(2);
            pls->vpor( 0.0, 1.0, 0.0, 1.0 );
            pls->wind( -2.5, 2.5, -3.0, 3.0 );
            pls->w3d(2,4,3,0,10,0,1000,-360,360,45,30);
            // x = dipoleData[cf].data[i].id;
            z = dipoleData[cf].data[i].instAngularVelocity;
            // double x = i;
            // double z=i;
            // y = cf;
            pls->col0((i+1)%10);
            pls->poin3(1,&x, &y, &z,1);
          }
        }

        pls->adv(3);
        pls->stripa(id1,0,cf,minAngularVelocity);
        //NOTE: The library apparently has issues if you stop
            printing both! it crashes at runtime :(
        if(plotSqKE)
          pls->stripa(id1,1,cf,(dipoleData[cf].
              meanSquaredAngularVelocity));
        else
          pls->stripa(id1,1,cf,(dipoleData[cf].temperature));

      }
    #endif

    //////////////DRAWING THE CONTOUR AND DIPOLE
    /// Draw contours + rotated rects + ellipses
    drawing = Mat::zeros( threshold_output.size(), CV_8UC3 );
    for( size_t i = 0; i< contours.size(); i++ )
       {
        // Scalar color = Scalar( rng.uniform(0, 255), rng.
            uniform(0,255), rng.uniform(0,255) );
        Scalar color = Scalar(0,0,255 );
        // contour
        drawContours( drawing, contours, (int)i, color, 1, 8,
            vector<Vec4i>(), 0, Point() );

        // ellipse
        if(i<minEllipse.size())
                            ellipse( drawing, minEllipse[i],
                                color, 2, 8 );

        // rotated rectangle
        // Point2f rect_points[4]; minRect[i].points(
            rect_points );
        // for( int j = 0; j < 4; j++ )
```

```
        //    line( drawing, rect_points[j], rect_points[(j+1)
            %4], color, 1, 8 );
        // }


        // int xx=dipoles[k][i].x;
        // int yy=dipoles[k][i].y;
        // int theta=dipoles[k][i].angle;

        // line(drawing, Point2f(xx,yy),Point2f(xx + 5*cos(
            theta), yy + 5*sin(theta)), Scalar(0,0,255),1,8);

       }

    for( int i=0;i<dipoles[0][0].count[k];i++)
    {

      int xx=dipoles[k][i].x;
      int yy=dipoles[k][i].y;
      double theta = (3.1415926535/180) * dipoles[k][i].angle;

      line(drawing, Point2f(xx - 5*cos(theta), yy - 5*sin(theta
          )),Point2f(xx + 5*cos(theta), yy + 5*sin(theta)),
          Scalar(0,255,255),5,8);



      // Use "y" to show that the baseLine is about
      char text[30];
      // dipoles[0][0].count[0]=1;
      // sprintf(text,"%f",dipoles[0][dipoles[0][0].count[k
          ]-1].angle);
      // sprintf(text,"%f",dipoleData[dipoleData.size()-1].
          meanSquaredAngularVelocity);
      int fontFace = FONT_HERSHEY_SCRIPT_SIMPLEX;
      double fontScale = 0.5;
      int thickness = 1;

      int baseline=0;
      Size textSize = getTextSize(text, fontFace,
                                  fontScale, thickness, &
                                      baseline);
      baseline += thickness;

      // center the text
      Point textOrg((drawing.cols - textSize.width)/2,
                    (drawing.rows + textSize.height)/2);
        // // draw the box
        // rectangle(drawing, textOrg + Point(0, baseline),
        //           textOrg + Point(textSize.width, -textSize.
            height),
        //           Scalar(0,0,255));
```

```
          // // ... and the baseline first
          // line(drawing, textOrg + Point(0, thickness),
          //      textOrg + Point(textSize.width, thickness),
          //      Scalar(0, 0, 255));

          // then put the text itself
          // putText(drawing, text, textOrg, fontFace, fontScale,
               Scalar::all(255), thickness, 8);
        sprintf(text,"%1.1f",dipoles[k][i].angle);
        putText(drawing, text, Point(dipoles[k][i].x,dipoles[k][i
            ].y), fontFace, fontScale, Scalar::all(0), thickness
            *3, 8);
        putText(drawing, text, Point(dipoles[k][i].x,dipoles[k][i
            ].y), fontFace, fontScale, Scalar::all(255),
            thickness, 8);


        sprintf(text,"%d,%d",dipoles[k][i].id,i);

        if(dipoleRec==true)
        {
          if(dipoles[k][i].id < seedDipole.data.size())
            sprintf(text,"[%d],%d",seedDipole.data[dipoles[k][i].
                id].id,i);
        }


        putText(drawing, text, Point(dipoles[k][i].x,dipoles[k][i
            ].y-10), fontFace, fontScale, Scalar::all(0),
            thickness*3, 8);
        putText(drawing, text, Point(dipoles[k][i].x,dipoles[k][i
            ].y-10), fontFace, fontScale, Scalar(255,255,0),
            thickness, 8);

        //DEBUG ONLY
        if(i==0)
        {
          cimg = Mat(src.rows,src.cols+500, CV_8UC3, Scalar
              (0,0,0));
          // sprintf(text,"%1.1f",dipoles[k][i].angle);
          sprintf(text,"Press p to seed");
          // sprintf(text,"%1.1f",dipoleData[dipoleData.size()
              -1].data[0].instAngularVelocity);

          // if(dipoleData[dipoleData.size()-1].
              meanSquaredAngularVelocity >0)
          if(dipoleRec==true)
            sprintf(text,"%1.1f",dipoleData[dipoleData.size()-1].
                meanSquaredAngularVelocity);
          putText(cimg, text, Point(src.rows/4,src.cols/4),
              fontFace, fontScale*12, Scalar::all(255), thickness
              *4, 8);
```

```
  }
}


//////////////////////////
//  THIS IS HOUGH
//////////////////////////
// // cvtColor(img, cimg, CV_GRAY2BGR);
// // cimg=src_gray;
// // Mat cimg();
// Mat cimg(src.rows,src.cols, CV_8UC3, Scalar(255,255,255)
    );

// vector<Vec3f> circles;
// HoughCircles(src_gray, circles, CV_HOUGH_GRADIENT, 1,
    10,
//              canny, centre, minMinorAxis, maxMajorAxis
    // change the last two parameters
//                              // (min_radius & max_radius)
     to detect larger circles
//              );

// // src_gray:s Input image (grayscale)
// // circles: A vector that stores sets of 3 values: x_{c
    }, y_{c}, r for each detected circle.
// // CV_HOUGH_GRADIENT: Define the detection method.
    Currently this is the only one available in OpenCV
// // dp = 1: The inverse ratio of resolution
// // min_dist = src_gray.rows/8: Minimum distance between
    detected centers
// // param_1 = 200: Upper threshold for the internal Canny
     edge detector
// // param_2 = 100*: Threshold for center detection.
// // min_radius = 0: Minimum radio to be detected. If
    unknown, put zero as default.
// // max_radius = 0: Maximum radius to be detected. If
    unknown, put zero as default

// for( size_t i = 0; i < circles.size(); i++ )
// {
//     Vec3i c = circles[i];
//     // Scalar color = Scalar( rng.uniform(0, 255), rng.
    uniform(0,255), rng.uniform(0,255) );
//     Scalar color = Scalar( 255,255,0 );
//     circle( cimg, Point(c[0], c[1]), c[2], color, 3,
    CV_AA);
//     circle( cimg, Point(c[0], c[1]), 2, color, 3, CV_AA)
    ;
// }

// imshow("Debug", cimg);
```

```
#ifndef MULTI_THREAD_DISPLAY
  updateDisplay();
#endif

#ifdef MULTI_THREAD_DISPLAY
  // drawnow=true;
  #ifndef ATOMIC_DISPLAY
    drawnow.unlock();
  #else
    updateDisplayRequested=true;  //Both are required to
        trigger an update
    // updateDisplayCompleted=false; //The request flag is
        used to indicate processing is done, the second
        indicates the same here
    //however, once the frame has been updated, the update
        flag avoids unecessary refreshing of the frames.
  #endif

#endif


//////////////////////
// CLI
//////////////////////
char key = (char) waitKey(5); //delay N millis, usually
    long enough to display and capture input
int kMax; //sorry, bad programming, but relatively
    desparate for results..
switch (key)
{
    #ifdef GRAPHS_ENABLED
      case 'o':
      {
        plotSqKE=!plotSqKE;
        cout<<endl<<"Plot measure of inst. average kinetic
            energy:"<<plotSqKE;
        break;
      }
    #endif
    case 'o':
    {
      cout<<endl<<"Input the coil angle"<<endl;
      cin>>coilAngle;
      cout<<endl<<"CoilAngle updated to "<<coilAngle;
      break;
    }
    case 'i': //Initialize the indices of the seed
    {
      //you've to use brackets in case if you want local
          variables!
      if(dipoleRec==true)
      {
```

```
    cout<<endl<<"Input the indices in order to
        calibrate "<<endl;
    for (int newLocation=0;newLocation<seedDipole.data.
        size();newLocation++)
    {
        int val;
        cin>>val;
        seedDipole.data[val].id=newLocation;
    }
  }
  else
  {
    cout<<endl<<"Start dipole recording using the key '
        p' and try again";
  }
  break;

}
case 'c':
  #ifdef GRAPHS_ENABLED
    clearGraph();
  #endif
  break;
case 'C':
  mode=1;
  cout<<"Mouse will capture color now. Right click for
      one, left for the other"<<endl;
  break;
case 'S':
  mode=0;
  cout<<"Screen crop mode selected. Mouse will capture
      start point at left click and the other point at
      right click"<<endl;
  break;
case 'p':
  {
    cout<<"Look what you've done!"<<endl<<"Just kidding
        : This frame will be used as a seed"<<endl;
    dipoleRec=true; //Enable dipole recording
    seedDipole.data.clear();  //clear the data
    dipoleSkel tempDipole;  //create a temporary dipole
         skeleton
    k=dipoles[0][0].current;  //find the current buffer
         of dipoles detected (double buffered for
        possible multithreading)
    kMax=dipoles[0][0].count[k]; //find the number of
        dipoles detected in the last scan

    for(int c=0;c<kMax;c++)
    {
```

```
            tempDipole.x=dipoles[k][c].x; //Copy the relavent
                data from the dipole data collected into the
                temp dipole
            tempDipole.y=dipoles[k][c].y;
            tempDipole.angle=dipoles[k][c].angle;
            tempDipole.instAngularVelocity=0;
            tempDipole.detected=false;   //This is to ensure
                the dipole was detected, but for the seed
                frame, it is left false.
            tempDipole.id=c;
            seedDipole.data.push_back(tempDipole);  //Add the
                data in the seedframe's data stream

            seedDipole.order+=dipoles[k][c].order;  //to get
                teh average order
            if(c>0)
            {
              seedDipole.order/=2.0;
            }
          }
          seedDipole.time=0; //Initial time is to be stored
              as zero
          dipoleData.push_back(seedDipole);
          break;
        }
      case 'w':
        {
          cout<<endl<<"Writing angle vs time for the first
              dipole to file"<<endl;
          if(dipoleRec==true)
          {
            sprintf(fileName,"latticeAnalyserBeta_%d",
                getTickCount());
            pFile = fopen (fileName,"w");

            //Loop through all the frames
            for (vector<dipoleFrame>::iterator dD =
                dipoleData.begin() ; dD != dipoleData.end();
                ++dD)
            {
              //Within each frame, loop through all dipoles?
              // for(vector<dipoleSkel>::iterator dS = dD.
                  data.begin() ; dS!=dD.data.end() ; ++dS)
              // {

              // }
              //or just print the first dipole
              if(dD->data[0].detected)
                fprintf (pFile, "%f,%f,%f\n",dD->data[0].
                    angle,dD->time,dD->
                    meanSquaredAngularVelocity);
                  //->data[0].instAngularVelocity);
```

```
            // fprintf (pFile, "%d,%d\n",dD->data[0].angle,
                dD->time);
          }


          // for (int p=0;p<dipoleData.size();p++)
          // {
          //   fprintf(pFile,"%d,%d\n",dipoleData[p].data.
              size(),dipoleData[p].time);
          // }
          fclose (pFile);
          // fprintf (pFile, "Name %d [%-10.10s]\n",n,name)
              ;

        }
        break;
      }
    case 'F':
    {
      // Table Description
      //  Time    Dipole 0    Dipole 1    Dipole 2 ...
      // cout<<endl<<"Writing angle of all dipoles for the
          given frame"

      if(dipoleRec==true)
      {
        cout<<endl<<"Writing all the data collected so far
            into a file";
        sprintf(fileName,"latticeAnalyserRENAMEorLOSEme");
        pFile=fopen(fileName,"w");
        //Write the column names (very important to figure
            out which dipole corresponds to which spatial
            location)
        fprintf(pFile,"Time");
        for(int fi=0;fi<seedDipole.data.size();fi++)
        {
          fprintf(pFile,"\t Dipole %d",seedDipole.data[fi].
              id);
        }
        fprintf(pFile,"\n");

        //Now write the data
        for (vector<dipoleFrame>::iterator dD=dipoleData.
            begin(); dD!=dipoleData.end();dD++)
        {
          //The first entry is time
          fprintf(pFile,"%f",dD->time);
          for(vector<dipoleSkel>::iterator dData=dD->data.
              begin(); dData!=dD->data.end(); dData++)
          {
            //Second entry onwrds, we have the dipole
                angles
```

```
            if(dData->detected==true)
              fprintf(pFile,"\t %f",dData->angle);
            else
              fprintf(pFile,"\t");
          }
          fprintf(pFile,"\n");
        }
        fclose (pFile);
        cout<<endl<<"Written! Unless something broke..";
      }
      else
        cout<<endl<<"Data recording is off. Turn it on
            using 'p'.";
      break;
    }
    case 'b':
      //This is to make blind
      blind=!blind;
      cout<<"Blind:"<<blind<<endl;
      break;
    case 'd':
      useCalibration=!useCalibration;
      cout<<endl<<"Calibration Use:"<<useCalibration<<endl;
      break;
    case 'I':
      //invert push
      invertPush=!invertPush;
      cout<<"Push"<<invertPush<<endl;
      break;
    case 'W':
      {
        pFile=fopen("TestComputation","w");
        for(vector<double>::iterator d=computationTime.
            begin();d!=computationTime.end();++d)
        {
          fprintf(pFile,"%f\n",*d);
        }
        fclose(pFile);
        break;
      }
    case 'q':
    case 'Q':
    case 27: //escape key
        destroyWindow(source_window);
        destroyWindow(filter_window);
        destroyWindow(settings_window);
        destroyWindow("Contours");
        destroyWindow("Debug");

        #ifdef TEMPERATURE_ENABLED
            vCloseUSB();
        #endif
```

```
            threadsEnabled=false;
            t1.join();
            #ifdef MULTI_THREAD_DISPLAY
              t2.join();
            #endif
            // destroyAllWindows();
            // this_thread::sleep_for( chrono::milliseconds
                (5000) );
            // waitKey(1000);
            return 0;
        // case ' ': //Save an image
        //     sprintf(filename, "filename%.3d.jpg", n++);
        //     imwrite(filename, frame);
        //     cout << "Saved " << filename << endl;
        //     break;
        default:
            break;
      }
      tCend=getTickCount();
      tCdelta=tCend-tCstart;
      computationTime.push_back(tCdelta/getTickFrequency());
    }
    // processingImage.unlock();
  }
  #ifdef GRAPHS_ENABLED
    delete pls;
  #endif
  return 0;


}

// double distSq(int x1,int y1,int x2,int y2)
// {
//   return (pow(x1-x2,2) + pow(y1-y2,2));
// }
// void latticeAxis(vector<dipoleSkel>& data,vector<double>&
    angles)
// {
//   int lastDistance=10000;
//   // Calculating the shortest distance squared, between the
    first point and all other points
//   for(int i=1;i<data.size();i++)
//   {
//     int distance=pow((data[0].x-data[i].x),2) + pow((data[0].y
    -data[i].y),2);
//     lastDistance=MIN(distance,lastDistance);
//   }


//   double tolLower=0.5;
//   double tolHigher=1.5;
```

```
//   for(int i=0;i<data.size();i++)
//   {
//     for(int j=0;j<data.size();j++)
//     {
//       if(i!=j)
//       {
//         double neiDist=distSq(data[i].x,data[i].y,data[j].x,
    data[j].y)/(lastDistance)
//         if( neiDist > tolLower && neiDist<tolHigher ) //Yup,
    found a neighbour!
//         {
//           double angleFound=atan2( (data[i].y - data[j].y) / (
    data[i].x - data[j].x) );
//           angles.push_back(angleFound); //Added the angle
    found to an array
//         }
//       }
//     }
//   }
// }

// inline void latticeAxisTest()
// {
//   // Calculate shortest distance squared between the first
    point and all other pointss
//   // for each point, find all neighbours, store the angle
    determined in an array
//   // TODO: In the previous step, add a method to avoid
    incorrect counting

// }

// void latticeAxis(vector<dipoleSkel>& data)
// {
  //Find neighbours
  //Find the ones with only 3 neighbours
  //Within these, find the ones with a particular slope missing


// double a=3.14;  // The value you seek
// std::find_if(v.begin(),v.end(),[a](double b) { return a>b-
    epsilon && a<b+epsilon; });

// }
#ifdef TEMPERATURE_ENABLED
#ifdef TEMPERATURE_SINGLDIPOLE
  inline void fireElectro(long frame,int id)
  {
    //this is to avoid too many fires within a short time
    static long lastFrame=0;
    // static bool alternate=false;
    if(frame-lastFrame>3)
```

```
    {
      // alternate=!alternate;
      // if(alternate)
      // {
        char usbBuf[REPORT_LEN]={dipolePort[id],dipoleBit[id
            ],0,200};

        nWriteUSB((unsigned char*)usbBuf,14);
        lastFrame=frame;
        cout<<endl<<">> Temperature: Electro Fired for dipole "<<
            id;

      // }
    }
  }
#else
  void fireElectro(int id, int intensity)
  {
        char usbBuf[REPORT_LEN]={dipolePort[id],dipoleBit[id],0,
            intensity};

        nWriteUSB((unsigned char*)usbBuf,14);
        // lastFrame=frame;
        cout<<endl<<">> Temperature: Electro Fired for dipole "<<
            id<<" with intensity "<<intensity<<" ";
  }
#endif
#endif

void temperatureTest()
{
#ifdef TEMPERATURE_ENABLED
  cout<<"Which dipole to fire?";
  int i=0;
  cin>>i;

  char usbBuf[REPORT_LEN];
  for(int q=0;q<REPORT_LEN;q++)
  {
    usbBuf[q]=0;
  }

  // usbBuf[0]='B';
  // usbBuf[1]=5;
  usbBuf[0]=dipolePort[i];
  usbBuf[1]=dipoleBit[i];
  usbBuf[2]=255;
  usbBuf[3]=255;


  cout<<"temperature Test"<<endl<<endl;
  cout<<"Initializing Hardware"<<endl;
```

```
  vInitUSB();
  cout<<endl;
  // cout<<"Initialization Successful"<<endl<<endl;

  cout<<"Writing to hardware:"<<usbBuf<<endl;
  int usbLen;
  if( (usbLen=nWriteUSB( (unsigned char *) usbBuf,14)) )
  {
    cout<<"Writing Successful"<<endl<<endl;
  }

  cout<<"Reading from hardware"<<endl;
  usbLen=nReadUSB( (unsigned char*) usbBuf);
  if(usbLen==0)
    cout<<"Failed!"<<endl<<endl;
  else
  {
    usbBuf[usbLen]= '\0';
    cout<<"Data Read: "<<usbBuf<<endl;
    for(int i=0;i<usbLen;i++)
      printf("%4d",usbBuf[i]);
    cout<<endl<<endl;
  }
  vCloseUSB();

#else
        cout<<"Temperature Support not enabled";
#endif
}

/**
 * @function main
 */
int main( int ac, char** argv )
{
  initializeMultithreadResources();
  #ifdef TEMPERATURE_ENABLED
    getTemperaturePins();
  #endif

  cout<<"Loading";
    cout<<endl<<endl
        <<"Lattice Analyser | version "<<version<<endl
        <<"_____________"<<endl
        <<"Created at the National Physical Laboratory, New Delhi
            "<<endl
        <<endl
        <<"Project Repository Folder: github.com/toAtulArora/
            IISER_repo/Summers_2013/NPL"<<endl
        <<endl
        <<"For help type"<<endl
        <<"help"<<endl
```

```
      <<"(Like you couldn't guess!)"<<endl<<endl
      <<"\t now what?  ";


for(;;)
{
  string a;
  cin>>a;

  if(!a.compare("help"))
  {
    cout<<endl; //for multi line, beauty stuff

    cout<<"Command \t Description"<<endl
    <<"------- \t ----------"<<endl
    <<"temp    \t Launches hardware test"<<endl
    <<"temperature \t same as temp"<<endl
    <<"<number> \t Initiates analysis of dipoles using the
        corresponding camera"<<endl
    <<"q        \t exit or quit"<<endl;

    cout<<endl; //again for multi line console outputs, to
        maintain beuty
  }
  else if(!a.compare("exit") || !a.compare("quit") || !a.
      compare("q"))
  {
    break;
  }
  else if(!a.compare("temperature") || !a.compare("temp"))
  {
    temperatureTest();
  }
  else
  {
    // if(atoi(a.c_str())!=0 || !a.compare("0"))
    cout<<"Commands for this mode:"<<endl
    <<"c \t Will clear the graphs"<<endl
    <<"C \t Enable mouse capture of colour"<<endl
    <<"S \t Screen crop is selected, viz Drag with left click
        to select a sub window"<<endl
    <<"p \t This frame will be selected as the seed frame"<<
        endl
    <<"w \t Write the angles and dipoles, of the first dipole
        to file"<<endl
    <<"W \t Write computation times to file"<<endl<<endl;

    threadsEnabled=true;
    getCameraCalibrationParameters();
    VideoCapture capture; //try to open string, this will
        attempt to open it as a video file
```

```
      // if (!capture.isOpened()) //if this fails, try to open as
           a video camera, through the use of an integer param

      capture.open(a);
      if(!capture.isOpened())
      {
          capture.open(atoi(a.c_str()));
      }

      if (capture.isOpened())
      {
        process(capture);

      }
      else
      {
          cerr << "Failed to open the video device specified" <<
              endl;
          // return 1;
      }
    }
    // else if(!a.compare("latticeAxis"))
    // {
    //   latticeAxisTest();
    // }
    cout<<endl<<"\t now what? ";
  }



  return 0;
}
```

B.2 TEMPERATURE

Following is the main C code for 'temperature'

tempereture.c

```
/*
        filename: temperature
        description: This is the firmware of the hardware that
            simulates temperature on the dipole lattice

        baby steps(TM):
                1. Bare Minimum Tests: Communication Test with
                    the lattice analyser     [done]
                2. Proof of Concept:
                        The target of this would be to keep a
                            dipole continously rotating
```

```
                    a. Test the kind of currents and voltages
                         required to fire up the magnet
                    b. Make a program and fire up the magnet
                        from the lattic analyser

        communication protocol [clear text]
                fixed length, 10 characters (8 bit each)

                xxxxx xxxxx
                \---/ \---/
                  |     |
                  |     +------- Field Intensity
                  +-------------Dipole ID
                for
        Binary Protocol for testing
                fixed length, 10 charactes, 8 bit each
                0 1 2 3 4   5 6 7 8 9
                x x x x x   x x x x x
                | | | |     \ - - - /
                | | | |         |
                | | \ |         +------------Undefined for now
                \ |  -|---------------------Dipole Intensity
                    (65536 max)
                 -|-----------------------Dipole ID (65536 max)
        Binary Protocol V0.2
                fixed length, 4 characters for now
                0 1 2 3 4
                x x x x x
                | | | |
                | | | |
                | | \ |
                \ |  -|---------------------Dipole Intensity
                    (65536 max)
                 -|-----------------------Dipole ID
                                                                    |

                                                                /

                                                                    \

                                                            Port


                                                        Bit


*/

#include "usbIO.h"
#include "avriomacros.h"
// #include "temperatureConfig.h"
```

```
// #define DEBUG_STAGE_ZERO

// This is to ease programming, reduce codesize
// #define port(x) port##x
// #define bit(x) bit##x
#define fireElectro(port,bit,duration) \
{ \
        out(bit,port);  \
        off(bit,port);  \
        for(U16Bit j=0;j<duration;j++) \
                _delay_us(1); \
        in(bit,port);   \
        on(bit,port);   \
}

#ifndef DEBUG_STAGE_ZERO
        // U16Bit acknowledge[10]="Ok";
        U16Bit* dipoleID;
        U16Bit* intensity;
        U8Bit   len;
        U8Bit*  data=0;
#endif

int main(void)
{
        #ifdef DEBUG_STAGE_ZERO
                U8Bit   i;
                U8Bit   ucInDataLen;
                U8Bit   *pucInData = 0;

                usbInit();

                // ---------------- enforce re-enumeration.
                usbDeviceDisconnect();  // enforce re-enumeration
                    , do this while interrupts are disabled!
                i = 255;
                while(--i)
                    // fake USB disconnect for > 250 ms
                        _delay_ms(1);
                usbDeviceConnect();

                sei();
                for(;;)
                {
                        usbPoll();
                        ucInDataLen = ucGetUSBData( &pucInData );
                        if ( ucInDataLen > INBUFFER_LEN )
                                ucInDataLen = INBUFFER_LEN;
                        if ( ucInDataLen )
                        {
                                    // copy indata and send back
                            ascii values incremented by 1
```

```
                                                                    //
                                                                        also
                                                                        ,
                                                                        make
                                                                        the
                                                                        first
                                                                        value
                                                                        ,
                                                                        the
                                                                        length
                                                                        of
                                                                        data
                                                                        recieved
                            pucOutBuf[0] = ucInDataLen;
                            for ( i = 0; i < ucInDataLen; i++
                                 )
                                pucOutBuf[i+1] =
                                    pucInData[i] + 1;
                            vSendUSBData( ucInDataLen+1 );
                    }
            }
            return 0;
    #else
            // for(;;)
            // {
            //      DDRB |= (1<<5);
            //      PORTB |= (1<<5);
            //      _delay_ms(1000);
            //      PORTB &= ~(1<<5);
            //      _delay_ms(1000);

            //      // DDRB=0xff;
            //      // // DDRC=0xff;
            //      // // PORTD=0xff;
            //      // PORTB=0xff;
            //      // _delay_ms(1000);
            //      // // DDRB=0x0;
            //      // // DDRC=0x0;
            //      // // PORTD=0x0;
            //      // PORTB=0x0;
            //      // _delay_ms(1000);
```

```
// }
DDRB=0;
DDRB &= ~(1<<5);
    //Make it high impedence again
PORTB |= (1<<5);
    //

usbInit();
                                //Initialize USB
usbDeviceDisconnect();
            //Disconnect and re-connect
_delay_ms(255);
usbDeviceConnect();

sei();
                                //Enable Interrupts
for(;;)
{
        usbPoll();
                                                //
            This has to be called frequently
            enough
        len=ucGetUSBData(&data);
                                //Get data
        if(len>INBUFFER_LEN)
                len=INBUFFER_LEN;
                                                //
                    truncate if overflew
        if(len)
                                                //Got
             data?
        {
                dipoleID=(U16Bit*)&data[0];
                                        //point the
                    dipoleID to its location in
                    the data
                intensity=(U16Bit*)&data[2];
                                //point the intensity
                     pointer to its location
```

```
if(*dipoleID==256)
                                //if
    the id is for the zeroth
    dipole (testing)
{
        pucOutBuf[0]=(U8Bit) (*
            intensity);
        // pucOutBuf[0]=(U8Bit)
            (*(intensity[0]))
        vSendUSBData(1);

                    //Acknowledge
             receiving data

        //TODO: ADD PORT CODE
                                //
            Fire up the magnet
            for a time
            proportional to the
            intensity

        DDRB |= (1<<5);

                    //Define as
            output
        PORTB &= ~(1<<5);

                    //Make port D
            .4 low, to fire the
            magnet
        for(U16Bit j=0;j<(*
            intensity);j++)
                _delay_us(1);
        DDRB &= ~(1<<5);

                    //Make it
            high impedence again
        PORTB |= (1<<5);

                    //

        // _delay_us((U8Bit) (*
            intensity/2) );
        // _delay_us((U8Bit) (*
            intensity/2) );
        //TODO: ADD CODE TO TURN
            OFF
}
else
{
        char pin=data[1];
```

```
                                        pucOutBuf[0]=':';
                                        pucOutBuf[1]='(';

                                        // if(data[0]=='A')
                                                // fireElectro(A,
                                                    pin,*
                                                    intensity)
                                        if(data[0]=='B')
                                        {
                                                fireElectro(B,pin
                                                    ,*intensity)
                                        }
                                        else if(data[0]=='C')
                                        {
                                                fireElectro(C,pin
                                                    ,*intensity)
                                                pucOutBuf[0]='C';
                                                pucOutBuf[1]=data
                                                    [1];
                                        }
                                        else if(data[0]=='D')
                                                fireElectro(D,pin
                                                    ,*intensity)
                                        // don't even think of
                                            missing the brackets,
                                             the macro is a set
                                            of statements!

                                        vSendUSBData(2);

                                                //Acknowledge
                                             receiving data

                                }
                        }
                }
                return 0;
        #endif


}
```

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede, for LaTeX.
The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*".

The latest version of this document is available online at:

`https://github.com/toAtulArora/dipoles`