Title:

**Introduction:**

Mitochondria is a double membrane bounded organelle found in many different types of cells in both plants and animals. It controls many different kinds of cellular activities. Some researchers also find that it may implicated some human diseases like heart problem, mitochondrial disorder, autism ,etc. Our project is aim at develop an algorithm and implement it with a graphical user interface for doing mitochondria tracking to help biologist and physicians study mitochondria’s behavior and obtain useful information from the data.

**Methodology:**

We have divided our project into few modules, Pre-tracking modules, tracking modules and data processing modules. We have finished the pre-tracking modules and still working on tracking modules. In this section, we will talk about the algorithm and some implementation work we have done on those finished parts and give some overview and planning to other parts that we are still working on it.

Before we drift into our work on this module, we will have a brief introduction to image processing. Any image can be defined as a 2D function f(x,y), where x and y represent both x and y axis in Cartesian plane. The function carries the intensity or gray value information of the image at that point and each point is called a pixel. Many pixels together will form a digital image. We can treat a picture just like a matrix in like fig1.

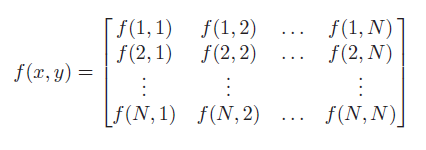


Fig1. A matrix view of a digital image

A video is a collection of a set of pictures or frames. In our project, the input is video. Through the application programming interface (API) of the OPENCV library, we can convert our input into various array consist of grey values of each pixel. Based on above basic information, we can start the introduction to our work.

**Pre-tracking modules:**

In the pre-tracking module, we have divided into three process, **construction of the tracking rail,** **refinement of the obtained path** and **elimination of unwanted information**.

1. **Construction of the tracking rail**

Since our input can be view as a black white video, we setup a black color and same size picture first. By run through every frame of the video and make use of the conversion equation; we can obtain the intensity value. If the obtained intensity value at certain point (x,y) in certain frame is greater than that in the black color pic, we replace the intensity by the latest greatest value at point (x,y) . In fig2, it is the first frame of our sample input and fig 3 is the trajectory of the movement of the mitochondria within our sample video.

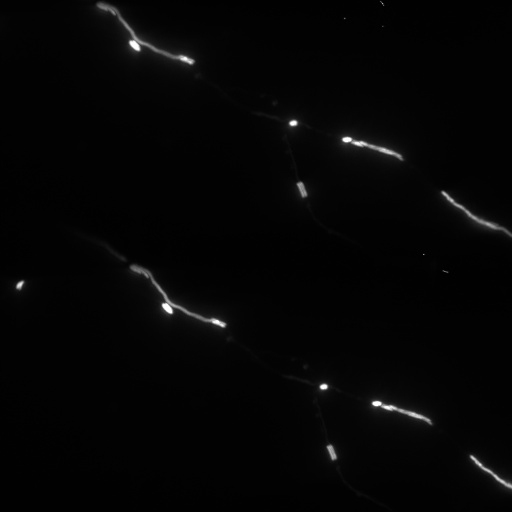
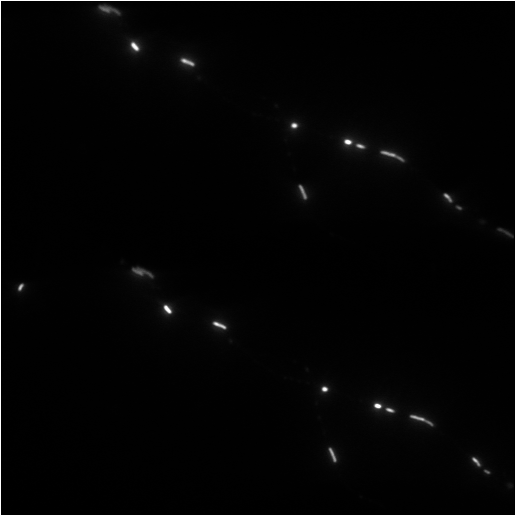


Fig2 The first frame of our sample Fig3 Trajectory of mitochondria in the video

With the help of this trajectory, we can use the drawing board function to select which trajectory we are going to track. Since the image can view as a matrix , we can also view it as a 0-1 Boolean matrix .We use the mouse pointer to select the points at the rail for tracking. If point (x’,y’) has been selected as a point within our targeting track it will be marked as 1 in our Boolean trajectory matrix ,else is 0. Fig 4 is demo of our resulting track after selection.

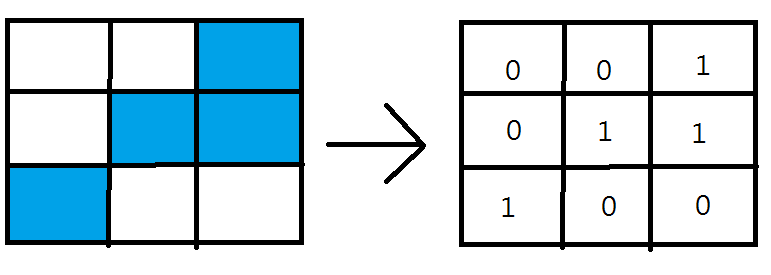


Fig 4a A simulation of digital image Fig 4b Representation in Boolean

with blue color represent the matrix

selected region

**Refinement of the obtained path**

After we obtained the targeting track, in most case it shouldn’t be any ‘hole’ within the track (see Fig5 example). However, if there is some mechanical failure on your mouse or whatever reasons, holes problem may occur. In order to ensure our quality of tracking and preventing any fault happen while calculation of center location, we will carry out a search similar to depth first search within the region of our ‘rail’ located and fill those zero with 1 in out Boolean trajectory matrix.

Fig5

**Elimination of unwanted information**.

Finally, before we start on tracking the mitochondria, we would first remove other points in the video that isn’t within our target rail. With the Boolean value matrix we did on previous section, we can sieve away the points (mitochondria) that aren’t our target easily. The reason to do this is because of preventing overlapping issue and causing calculation error for some frame within the video hence causing outliers in our tracking data.

It is quite abstract to state with only words so an example is shown on Fig 6 for the case of overlapping. In fig 6. Assume that the black solid line is the track we selected previously, and yellow is our target mitochondria, green is other mitochondria within the track and red one is some mitochondria not within our track. In some testing (details will be given in next part) we found that, if there is something similar happen, red and yellow might consider as same object depended on the color coefficient we used. To eliminate all the possibility to get error, we simply remove all content other than the track.

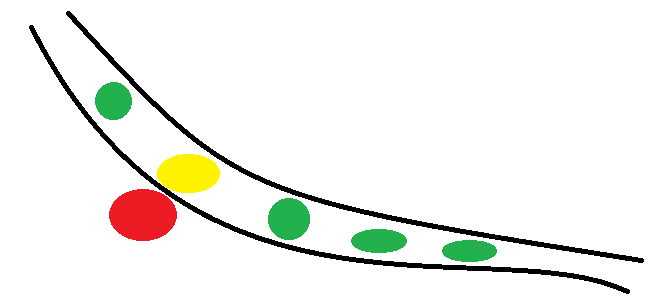


Fig 6 An illustration of the reason need to clear unwanted information

**Tracking modules:**

In this part, we only finished up to detection of mitochondria at each frame of the video, other parts are still working on it. So, I will give a brief summary to what I have read and give present some possible ideas that may works here.

In OpenCV library, there is a function called ‘contour’, our tracking will mainly base on this function with different interpretation method.

From the OpenCV library official definition, contour is a curve joining the boundary of an object which have some color or intensity, which match our target for tracking of mitochondria.

Firstly, we use function to map intensity value or RGB value into HSV values for carry out the detection. HSV is a cylindrical-coordinate representation of points in RGB model. Where h stands for hue, s for saturation and v for value. Fig7 is an illustration chart of HSV adopted from Wikipedia. Fig 8 is the mapping equation we used to map the value from RGB into HSV values. For more details, you can refer to the Wikipedia -HSL AND HSV page.

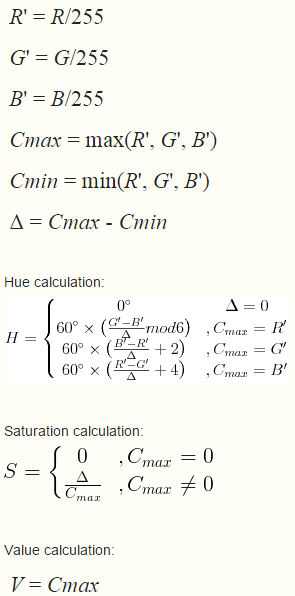
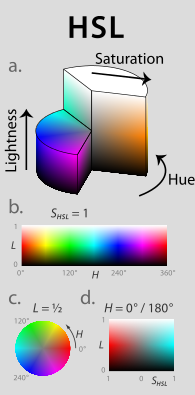


Fig 7 Illustration chart for Fig8 Mapping equation to convert

HSV coloring system RGB to HSV

From our testing, we found that the detection range of this kind of black and white mitochondria video is from HSV value (0,0,20) to (0,0,90) .This is a rough approximation with our samples, in real tracking ,this value should be set according to the target mitochondria we are going to track. As we can see from fig 9 to 12, with different HSV value, the contour will have different boundary on the same mitochondria.

Since we are using color for tracking, if two object very close to each other like Fig 6, possibility of being detected as one object is quite high, that’s why we eliminated all the point except those in the region we selected.

After we processed each frame with the color we want to detect, we would use the Canny edge detector build in the library to create contour for the objects.

The algorithm of this detector can be divided into few stages, noise reduction, intensity gradient within the frame, non-maximum suppression and hysteresis thresholding. Firstly, every frame will carry out noise reduction by a low-pass filter formulated by Gaussian distribution to reduce the picture from noise signal into a much tidy distribution. A sample adopted from John Canny ‘s paper is in Fig13 to illustrate this idea.

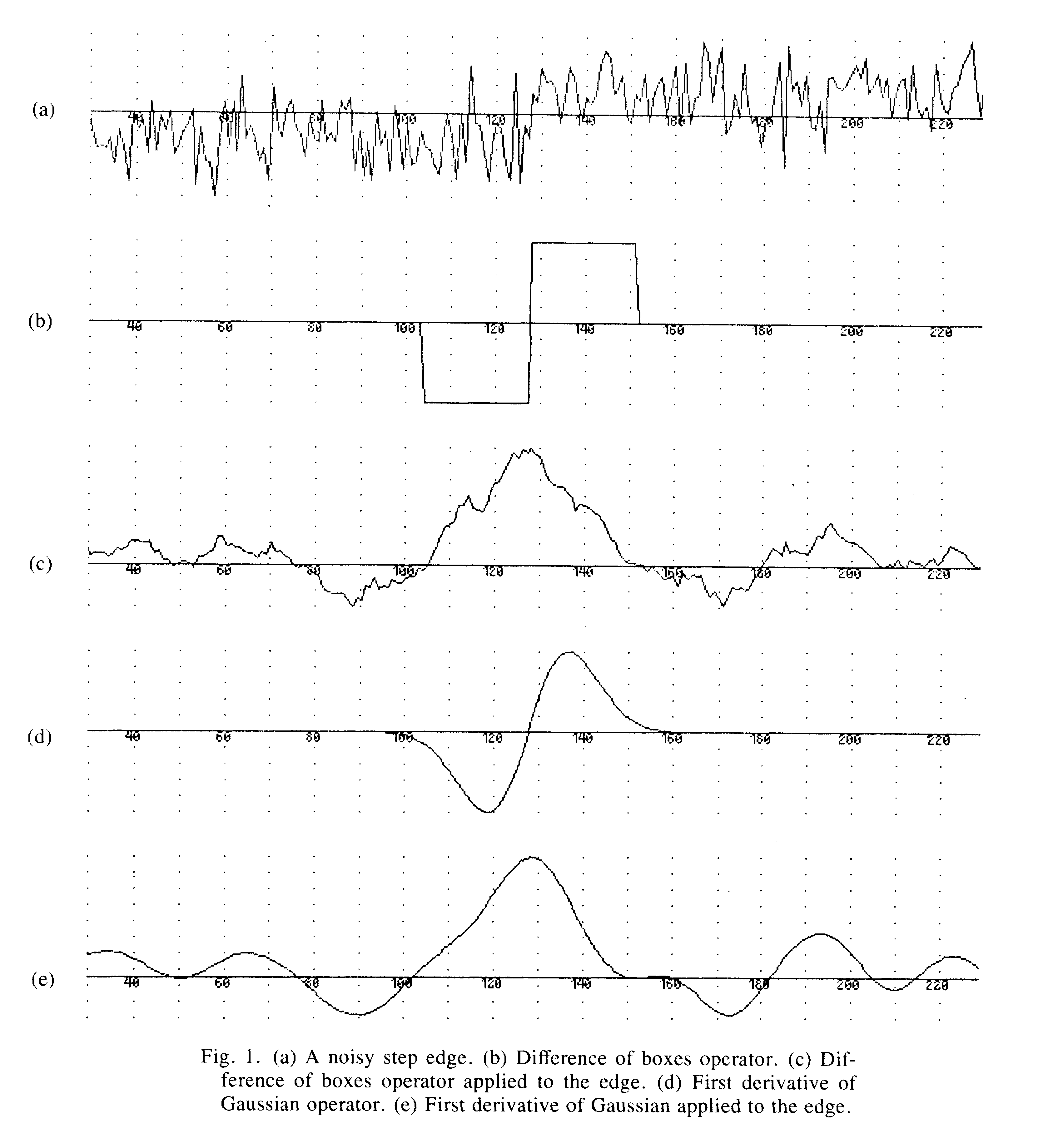


Fig 13 A chart adopted from John Canny’s paper to show the effect after applying Gaussian filter to the signal

Then, with the HSV value in each frame we obtained previously, we would plug the image matrix into the Sobel operator in both x and y direction to get the first derivative in x direction(Gx) and y direction(Gy). We can obtain the edge gradient of every pixel by the equation:



With the direction by this:



After obtained the magnitude and direction of the edge, the image will be scanned again to remove unwanted edges, and obtain a thin edge, an illustration obtained from OpenCV documentation is in Fig 14.



Fig 14 Illustration of suppression adopted from OpenCV documentation

Before carry out the non-maximum suppression, there will have some non-black pixel within the region between C to B. And assume that point A is on the edge and point B, C line in the same gradient direction. If point A is the local maximum, we will process to hysteresis thresholding else we will suppress it.

In the hysteresis thresholding process, the library has a build in threshold parameters for max intensity(maxm) and minimum intensity(minm). If the intensity gradient or HSV value gradient is more than the maxm, it will certainly be the edge and those below minm will certainly not.

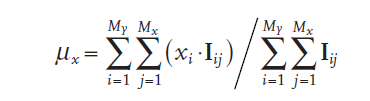
For those values in between maxm and minm, if that pixel is connected with some edge, it will be also consider as edge, else it isn’t belongs to the group of edge. For implantation details, you may refer to the official documentation of OPENCV and also the paper published by John Canny.

Starting from materials here, they are mainly ideas and information we got from research paper without any actual implementation.

Before we carry on the tracking process, we will classify the mitochondria into few different kinds of shape in order to obtain a better calculation of the center coordinate. They are circle, elipse and also some other irregular shape.

For circle, this would be the best case of locating of the center. By using the Hough Circle transform, we can obtain its center and also radius easily. Since we don’t have the parameter and radius information of certain mitochondria in certain frame, the algorithm will first iterate the radius and got a value R\*. In every point on the circle, it will draw a circle with radius R\*. The center would be located in the intersection of all the circle centered on every point of our targeted circle.

For irregular shape, we will use the idea of center of mass of object in physics and apply here to calculate the center of mass of image (Laurent Holtzer & Thomas Schmidt). The equation is follows:



,where I represent the intensity or signal in an arbitrary point (I,j). If we replace the Xi with Yi , we can obtain the y-axis of the center through the same equation.

For elipse , we didn’t come out of any ideas yet, so we just treat it as an irregular shape object at this moment.

Apart from center calculation problem, another issue that may happen in our tracking is cell division, cell fusion process and also two and more organelles just pass through.

For first two issues, we might solve them by five frame difference algorithm (SHU Xin& LI Dong-Xing&XUE Dong-Wei,2014) or three frame difference algorithm (Gan Ming-gang&Chen Jie&Liu Jin&Wang Ya-nan,2010) and combine the number of contour found in global and also that within a small interested region to judge whether cell fission or fusion has occupied or not.

After we have obtained the center of our target object in every frame of the video, cell fission or fusion happened or not information we would process to calculation of the slope by using the fundamental theorem of calculus to do it. To speed up the calculation process, we might use Queue data structure to store our center point and by dequeue process to extract points to do the calculations.

**Things to do:**

Apart from continue our work on doing the tracking of mitochondria, we hope to think of some better algorithm of approach for doing detection of elliptical object and also the irregular shape object to archive a better efficient as well as accuracy in calculation of the center.

Also, since the video maybe consist of many frame, optimization is needed in order to finish our job faster, and this might archive by using the GPU programming to shorten the runtime by using different hardware architectures.

Reference:

<https://en.wikipedia.org/wiki/HSL_and_HSV>

<http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html>

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf