# MDM2 Research

* Convert image to pixel values
* survos

## Motivation

Identifying bone breaks from X-rays (hairline fracture)

## Possible models

### Finite differences:

Refs:

* <https://www.researchgate.net/publication/357351763_Finite_difference_methods_in_image_processing> - initial finite differences method source [1]
* <https://aitskadapa.ac.in/e-books/CSE/DIGITAL%20IMAGE%20PROCESSING/Digital%20Image%20Processing%20(%20PDFDrive%20)%20(1).pdf> – more general source (section 12.4.3 applies to finite differences)[2]

Brief Summary:

* The simplest method to compute a gradient vector for a point
* Approximations are used for backward, forward and symmetric (central) differences. These approximations correspond to their respective filter masks

Kernels

* Kernels can be used to detect edges in images and the Laplacian filter is one example (shown through formulas) [1]:
* A white paper with black text and numbers

  Description automatically generated
* Convolution can be used to obtain the laplacian filter from
* A math equations and numbers

  Description automatically generated with medium confidence

Formulas:

* Forward, backward and central differences (sometimes referred to as symmetric ) for one variable (1st order)[1]:

A math equations on a white background

Description automatically generated

* Central differences for two variables x, y [1]:

A math equations on a white background

Description automatically generated

* Magnitude of the gradient between adjacent positions[1]:

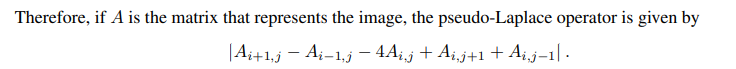
A math equations on a white background

Description automatically generated

A math equations with black text

Description automatically generated with medium confidence

* From this you can imply [1]:



## Possible images to use

## General research

* In RGB colour model, an image is represented with three grayscale images corresponding for each colour component, i.e. a matrix for red intensity, a matrix for green intensity, and a matrix for blue intensity, these matrices are known as masks.(see below from <https://www.researchgate.net/publication/357351763_Finite_difference_methods_in_image_processing> (finite differences 1) )

A group of different colors

Description automatically generated

* <https://aitskadapa.ac.in/e-books/CSE/DIGITAL%20IMAGE%20PROCESSING/Digital%20Image%20Processing%20(%20PDFDrive%20)%20(1).pdf> (see section 12 for edge detection, 12.4 for gradient based edge detection)

### Canny Algorithm:

Method (<https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm> ):

* Image **smoothed by Gaussian convolution**
* A simple **2-D first derivative operator** (somewhat like the [Roberts Cross](https://homepages.inf.ed.ac.uk/rbf/HIPR2/roberts.htm)) is applied to the smoothed image to **highlight regions of the image with high first spatial derivatives**. **Edges give rise to ridges in the gradient magnitude image**
* The algorithm then **tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top** to give a thin line in the output, a process known as *non-maximal suppression*.
* Hysteresis tracking:
  + The tracking process exhibits **hysteresis controlled by two thresholds: *T1* and *T2*, with *T1 > T2***.
  + Tracking can **only begin at a point on a ridge higher than *T1***. Tracking then **continues** in both directions out from that point **until the height of the ridge falls below *T2***.
  + This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

Implementation:

* OpenCV in python has a function cv.Canny that can be used to perform the Canny algorithm on a chosen image passed in as a parameter.
* Parameters of Canny(image, threshold1, threshold2, apertureSize, L2gradient)

A white background with black text

Description automatically generated

Improved Canny Algorithm? (<https://ieeexplore.ieee.org/abstract/document/6885761?casa_token=_YIQnFDHAAAAAAAA:UAQnKmHWtz9DOOK1_RnJ4VMwEpAz1tDcWZmFJklDWDqG0ZaxlIVU-zyYw5AwoFbkWxcFdStIyQ> )

* Using Newton’s law of universal gravity where the greyscale values replace the masses of the bodies. In the equation for ‘Gravitational intensity (E)’:

A mathematical equation with numbers and symbols

Description automatically generated

(where G is a constant and r is the vector from the pixel to each of its neighbours)

* This value for E\_total is understood as an image gradient and when this intensity reaches above a certain threshold, the point is an edge.
* Try implementing this and visualising this
* How to set threshold bounds:

A white background with black text

Description automatically generated

* + A text on a page

    Description automatically generated
  + The threshold is set from a combination of the gradient magnitude and other quantities (like standard deviation)
  + When implemented with G = /2 the following results were obtained:

A person with a hat and a number of lines

Description automatically generated with medium confidence

* + The 3 segmented images refer to the varied values for k used (1.2, 1.4 and 1.6)

### Frequency

Look at 2nd paper :

### Savitz Golay:

* Used for smoothing data points using Least Squares (sum of distance between polynomials sampled)
* Anti-symetrtric, purely imaginary component, at the highest frequency digitally possible (Nyquist), the SGDDs response will be 0

1. Take out neigborhood of N neighbours and fit a polynomial to it
2. Evaluate the polynomial at its center o, and continue with next neigbhourhood
3. It will find a lower oder polynomial that fits the original sequwence the best in least squares

**Advantages:**

* Least squares fitting is clear and easy to implement
* Convolution operation is easier
* Filter coefficients easily observed (integer values)
* To compute the derivative at a given point just find the derivative of the fitted polynomial
* It is a low pass differentiator, so is ideal at low frequency but attenuates at high frequencies which filters noise

**Disadvantages:**

* Not great for filtering blue noise in images, all samples have equal weight

A collage of graphs

Description automatically generated A collage of a person wearing a life jacket

Description automatically generated

KNN segmentation, good example of filtering noise code: <https://www.kaggle.com/code/naim99/knn-image-segmentation>

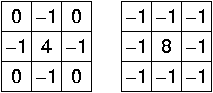
Derivative Operators in segmentation: <https://cmrcet.ac.in/files/ECE/ececoursefile/16.pdf>

### Gaussian Filters:

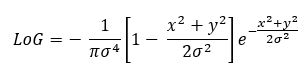
* Gaussian Smoothing
  + Make use of the gaussian lowpass filter, eliminating the higher frequencies feature along the frequency domain, thus blurring the image
  + The Gaussian has the form:



* + Two most common convolution kernels are:



* Application of Gaussian
  + Laplacian of Gaussian Filter
    - Combination of the Laplacian algorithm and gaussian filter
    - To improve the Laplacian's subject to noise
    - In general, a gaussian smoothing filter is applied to smooth the image before using the Laplacian



Gray scale:



<https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-7-201103-I!!PDF-E.pdf>

Python file is in github

Fourier transform:

Fourier transform: convert the spatial domain into the Fourier or frequency domain. Mainly used in image filtering, image de-noised.([Image Transforms - Fourier Transform)](https://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm#:~:text=The%20Fourier%20Transform%20is%20an%20important%20image%20processing,the%20input%20image%20is%20the%20spatial%20domain%20equivalent.)

Fourier transform and shift -> Gaussian Low-Pass Filter -> Inverse Fourier transform-> Canny algorithm (first three step has python code)

First three steps can remove high-frequency noise, and the parameters of the Gaussian low-pass filter can be adjusted to change the range of retained frequencies, which will make the edge of the image clearer.

Canny edge detection can make the edges be detected more accurately, the combination of using Fourier transform and Canny reduces the errors and avoids false edges.

Step 1(Fourier transform and shift):

f\_transform = fft2(image)

f\_shift = fftshift(f\_transform)

Step 2: I saw someone finished the python code of Gaussian filter part

Step 3:

f\_ishift = ifftshift(f\_shift\_filtered)

image\_filtered = ifft2(f\_ishift)

image\_filtered = np.abs(image\_filtered)

Step 4: I think other people is doing this part

Basic formula for 2D Fourier transform and reverse Fourier transform(used in report):

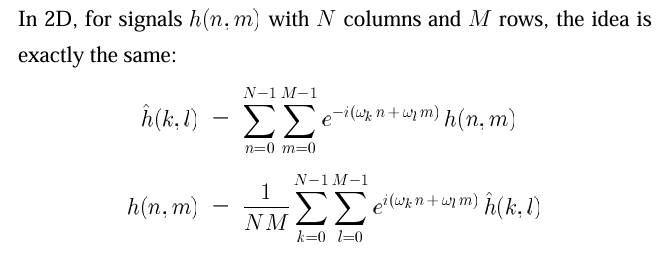
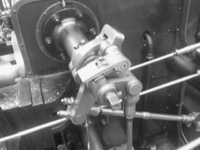
[linearFilters2.dvi](https://www.cs.toronto.edu/~jepson/csc320/notes/linearFilters2.pdf)

Image edge detection:

<https://en.wikipedia.org/wiki/Canny_edge_detector#Process> (canny edge detection, basic algorithm)



<https://ieeexplore.ieee.org/document/8667063> (other people’s report about edge detection and it has a good example of introduction)

<https://www.researchgate.net/publication/357351763_Finite_difference_methods_in_image_processing> - finite differences method

how to define good segmentation:

* try salt and pepper noise or Gaussian noise to corrupt the image
  + how much noise can the algorithm deal with
* test case
  + try tilted circle
  + decreasing intensity block colours
  + standard databases for testing (Lena etc.)