Internship Report

on

HUMAN POSE ESTIMATION

Submitted by

A. LOHITH

(20691A3128)

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL INTELLIGENCE

MADANAPALLE INSTITUTE OF TECHNOLOGY & SCIENCE
(UGC – AUTONOMOUS)

(Affiliated to JNTUA, Ananthapuramu)
Accredited by NBA, Approved by AICTE, New Delhi)
AN ISO 9001:2008 Certified Institution
P. B. No: 14, Angallu, Madanapalle – 517325

2023-2024

DEPARTMENT OF ARTIFICIAL INTELLIGENCE & DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that the internship work entitled **"HUMAN POSE ESTIMATION"** is a bonafide work carried out by

Submitted in partial fulfillment of the requirements for the award of degree **Bachelor of Technology** in the Department of **Artificial Intelligence**, **Madanapalle Institute of Technology and Science, Madanapalle,** affiliated to **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu** during the academic year 2023-2024

Faculty	**Internship Co-ordinator**	**Dr. K. Chokkanathan**
R.Ashok Kumar	**R.Ashok Kumar**	**Head & Associate Professor**
Assistant Professor,	**Assistant Professor**	**Department of AI**
Department of AI	**Internship Coordinator**	
	Department of AI	

ACKNOWLEDGEMENT

I sincerely thank the **Management** of **Madanapalle Institute of Technology and Science** for providing excellent infrastructure and lab facilities that helped me to complete this project.

I sincerely thank **Dr. C. Yuvaraj, M.E., Ph.D**., **Principal** for guiding and providing facilities for the successful completion of our project at **Madanapalle Institute of Technology and Science,** Madanapalle.

I express our deep sense of gratitude to **Dr. K. Chokkanathan, M. Tech., Ph.D.,** Associate Professor & Head, Department of AI for his continuous support in making necessary arrangements for the successful completion of the project.

I express our sincere thanks to the **Internship Coordinator**, **R. Ashok Kumar, M.S.,** Assistant Professor, Department of AI for his tremendous support for the successful completion of the internship Project.

.

I express our deep gratitude to **R.Ashok Kumar**, Assistant Professor, Department of AI for his guidance and encouragement that helped us to complete this internship project.

I also wish to place on record my gratefulness to other **Faculty members of Department of AI** and also to our friends and our parents for their help and cooperation during our internship.

CERTIFICATE

{Code Clause};

Code Clause

To Whom So IT May Concern

Date - 04 / 09 / 2023

This is to certify that **A. Lohith** , pursuing Computer Science & Engineering - Artificial Intelligence at **Madanapalle Institute of Science and Technology** has successfully completed an internship with CodeClause from **1-Jul-2023 To 1-Sep-2023**.

During this tenure he handled **Data Science Intern** position.

During the tenure of the Internship, **A. Lohith** has shown a great amount of responsibility, sincerity and a genuine willingness to learn and zeal to take on new assignments and challenges. In particular, his coordination skills and communication skills are par excellence and his attention to details is impressive.

We wish all the very best for your future.

with regards,
CodeClause

SUDAKU CODECLAUSE PVT LTD

MINISTRY OF CORPORATE AFFAIRS
GOVERNMENT OF INDIA

ISO 9001:2015

MSME

Certificate No - CC-CL84712

DECLARATION

I, the undersigned hereby declare that the results embodied in this Internship **"HUMAN POSE ESTIMATION"** is a bonafide record of the work done by me in partial fulfillment of the award of **Bachelor of Technology** in **Artificial Intelligence** from **Jawaharlal Nehru Technological University Anantapur, Ananthapuramu. The content of this report is** not submitted to any other University/Institute for award of any other degree.

Place: Madanapalle
Date:01-11-2023

A. LOHITH
Roll No. (20691A3128)
Department of Artificial Intelligence
Madanapalle Institute of Technology & Science
Madanapalle.

ABSTRACT

Human pose estimation aims at predicting the poses of human body parts in images or videos. Since pose motions are often driven by some specific human actions, knowing the body pose of a human is critical for action recognition. This survey focuses on recent progress of human pose estimation and its application to action recognition. We attempt to provide a comprehensive review of recent bottom-up and top-down deep human pose estimation models, as well as how pose estimation systems can be used for action recognition. Thanks to the availability of commodity depth sensors like Kinect and its capability for skeletal tracking, there has been a large body of literature on 3D skeleton-based action recognition, and there are already survey papers such as [1] about this topic. In this survey, we focus on 2D skeleton-based action recognition where the human poses are estimated from regular RGB images instead of depth images. We summarize the performance of recent action recognition methods that use pose estimated from color images as input, then show that there is much room for improvements in this direction.

CONTENTS

CHAPTER **Page No.**

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM STATEMENT

Since a normal image is not clear and it may not show some useful information or its brightness level seems inappropriate, then to get rid of such problems, we perform image enhancement to get a better and clearer image.

1.2 OBJECTIVE

The aim of image enhancement is to improve the interpretability or perception of information in images for human viewers, or to provide `better' input for other automated image processing techniques.

1.3 DIGITAL IMAGE PROCESSING

Image enhancement is the process of adjusting digital images so that the results are more suitable for display or further image analysis. For example, you can remove noise, sharpen, or brighten an image, making it easier to identify key features.

Digital Image Processing Tutorial provides basic and advanced concepts of Image Processing. Our Digital Image Processing Tutorial is designed for beginners and professionals both. Digital Image Processing is used to manipulate the images by the use of algorithms. For processing digital images, the most common software that used widely is Adobe Photoshop. Our Digital Image Processing Tutorial includes all topics of Digital Image Processing such as introduction, computer graphics, signals, photography, camera mechanism, pixel, transaction, types of Images, etc.

1.4 ART AND PHOTOGRAPHY

In the context of an "image enhancement project," the industry vertical would depend on the specific application or use case for which image enhancement is being employed.

Examples like Medical Imaging, Satellite Imaging, Media and Entertainment.

Digital image processing plays a significant role in the fields of art and photography, enabling artists, photographers, and designers to manipulate and enhance images to achieve specific creative goals. Photographers often use image enhancement techniques to improve the quality and aesthetics of their photos, whether for artistic purposes or commercial photography.

1.5 GITHUB

We need a dataset of low-resolution and high-resolution image pairs for training your GAN. Commonly used datasets include DIV2K, COCO, or custom datasets specific to your application.

1.6 PURPOSE

The central aim of Super-Resolution (SR) is to generate a higher resolution image from lower resolution images. High resolution image offers a high pixel density and thereby more details about the original scene. The need for high resolution is common in computer vision applications for better performance in pattern recognition and analysis of images. High resolution is of importance in medical imaging for diagnosis. Many applications require zooming of a specific area of interest in the image wherein high resolution becomes essential, e.g. surveillance, forensic and satellite imaging applications.

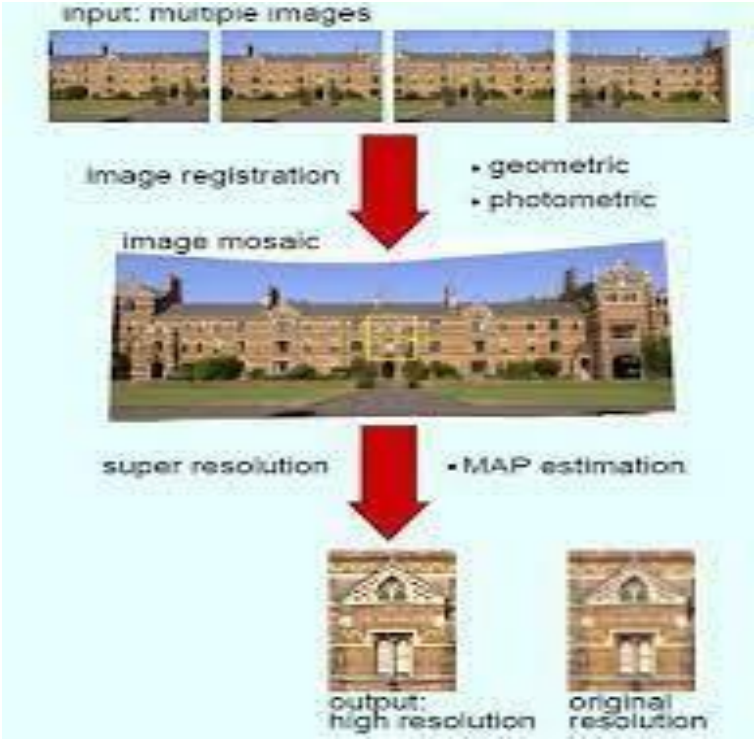

Fig1.1 Stages in Super-Resolution

1.7 METHODS

Digital image processing involves a wide range of methods and techniques to manipulate, enhance, analyse, and extract information from digital images. These methods are used in various fields, including photography, medicine, remote sensing, computer vision and more.

1. Super-Resolution:
   Enhances image resolution, creating higher-quality images from lower-resolution sources.
2. Deep Learning for Image Processing:
3. Utilizes deep neural networks, including Convolutional Neural Networks (CNNs) and Generative Adversarial Networks (GANs), for various image processing tasks such as image generation, style transfer and segmentation.

CHAPTER 2

HARDWARE AND SOFTWARE

2.1 PLATFORM AND HARDWARE USED

This section is to present a detailed description of the Platforms. It explains the hardware and software requirements for developing the application and its interface, tested features of the program

- A powerful GPU (NVIDIA GPUs, like the GeForce series or Tesla series, are commonly used for deep learning tasks).
- Sufficient RAM (at least 16 GB is recommended).
- A fast CPU to handle data preprocessing and model training.

2.2 SOFTWARE REQUIREMENT

- Python: We need Python installed on our system. Python 3.6+ is recommended or platforms like jupyter or colab.
- Deep Learning Framework: We should choose either TensorFlow or PyTorch, as per our preference.
- CUDA and cuDNN: If using NVIDIA GPUs, install CUDA and cuDNN for GPU acceleration.
- Popular Python libraries: Numpy, Matplotlib, PIL (Pillow), etc.

IDE	TYPE	PLATFORM	URL
ANACONDA is a free and open-source distribution of the Python and R Programming Languages for Data science and machine learning related applications	Free and open source	Windows Mac OS, Linux	https://www.anaconda.com/download/

Table 2.1 Tools and Platform us

CHAPTER 3

PROJECT ANALYSIS

3.1. ARCHITECTURE DIAGRAM

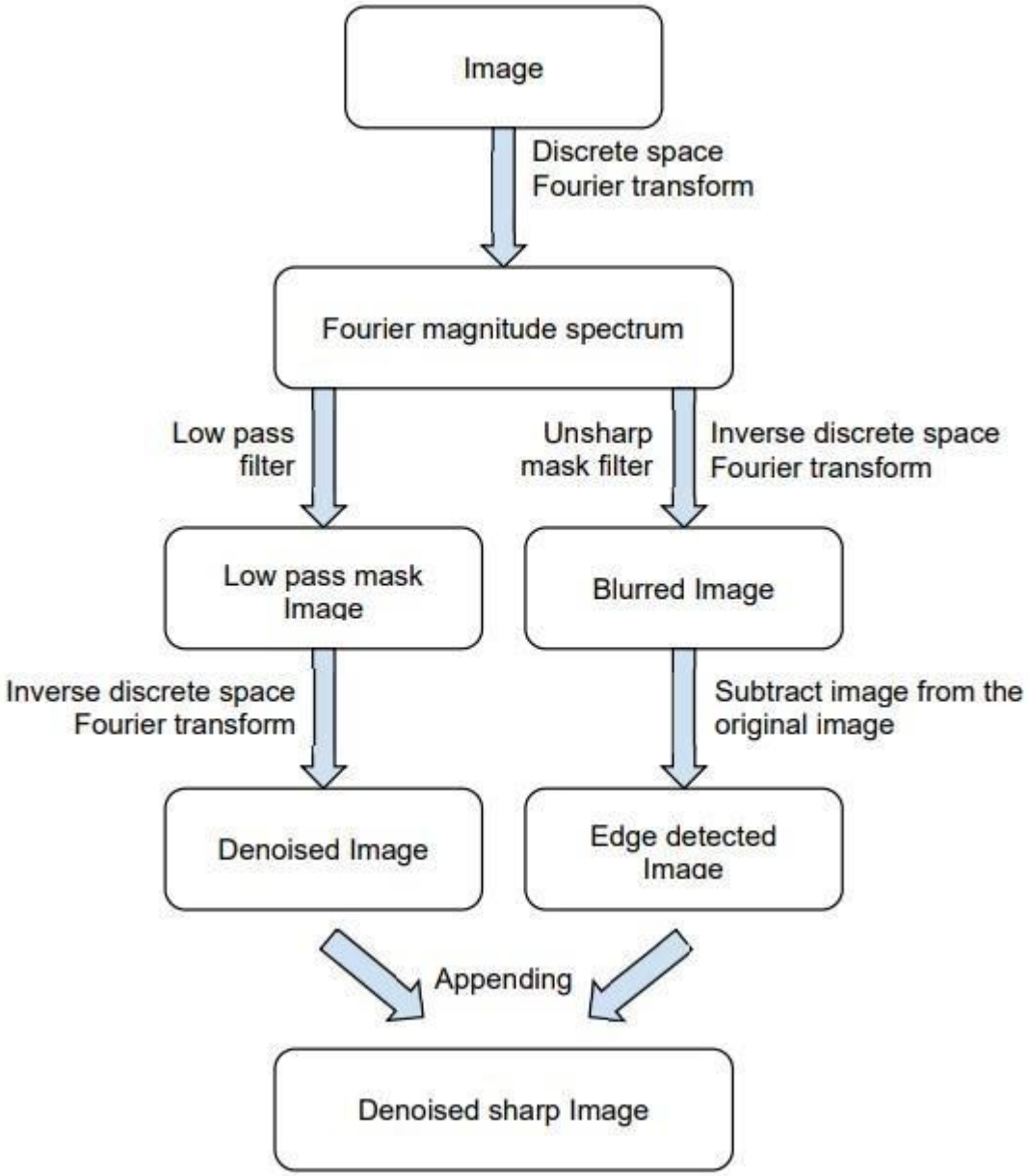

Fig3.1 Architecture Diagram

3.2. IMPLEMENTATION DATA

PREPARATION:

- I have collected and pre-processed our dataset, ensuring that we have corresponding pairs of low-resolution and high-resolution images.
- Crop, resize, and augment our data if needed.

BUILD THE GAN:

- Create a GAN architecture consisting of a generator and a discriminator.
- The generator takes low-resolution images as input and generates high-resolution images.
- The discriminator tries to distinguish between real high-resolution images and those generated by the generator.

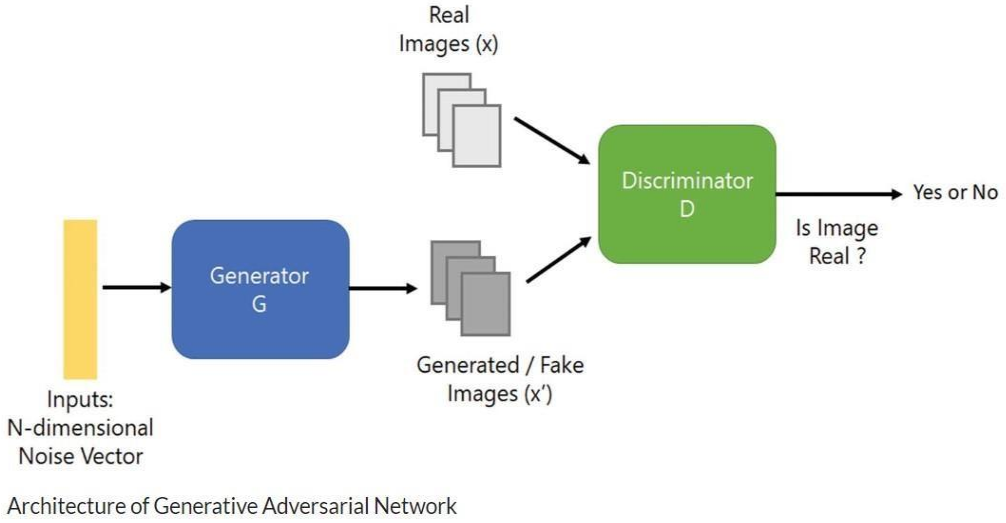

Fig3.2 GANS

TRAINING:

Training our GAN on the prepared dataset. This typically involves many epochs of training. Monitor loss functions to ensure convergence. Experiment with hyperparameters and network architectures to improve results.

TESTING AND EVALUATION:

Evaluate your model using a separate validation or test dataset. Common metrics for super-resolution tasks include PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index).

FINE-TUNING AND OPTIMIZATION:

Iterate on our model, fine-tuning hyperparameters and network architecture for better results.

DEPLOYMENT:

Once we are satisfied with our GAN's performance, we can deploy it to generate superresolution images on new, unseen data.

POST-PROCESSING:

Depending on our application, we might need to apply post-processing techniques to further enhance the generated images.

- An intelligent drawing interface for automatically generating images inspired by the color and shape of the brush strokes.
- An interactive visual debugging tool for understanding and visualizing deep generative models. By interacting with the generative model, a developer can understand what visual content the model can produce, as well as the limitations of the model.

3.3 ALGORITHM USED

3.3.1 Non-uniform Interpolation:

This approach consists of three stages i) registration, ii) non-uniform interpolation and iii) de-blurring

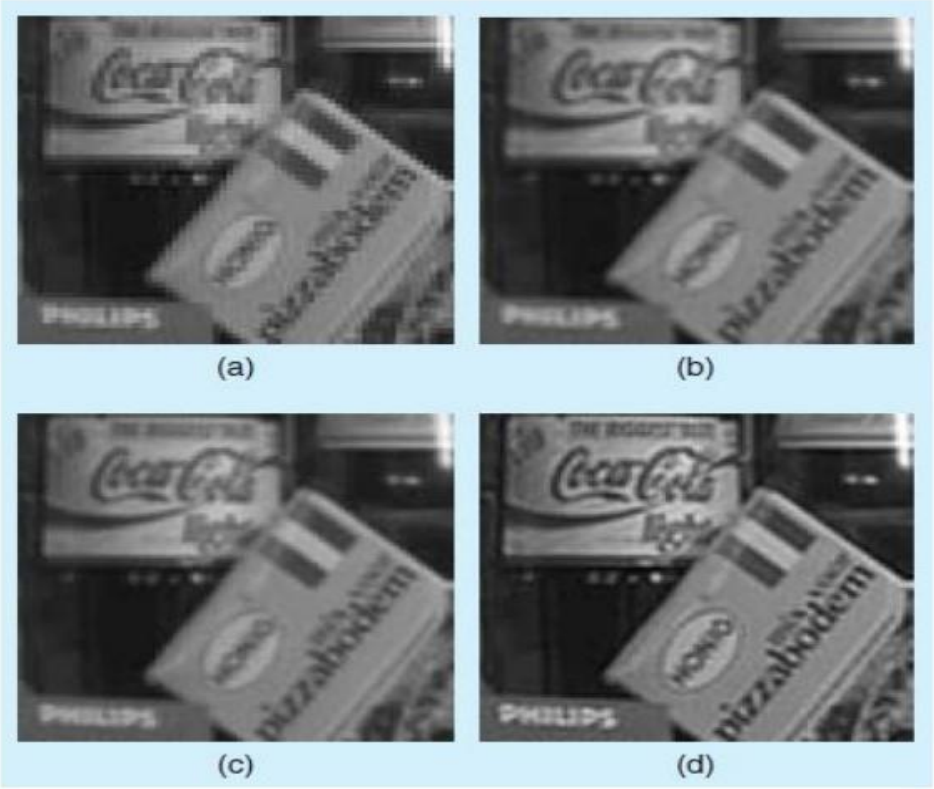

Fig 3.3 Nonuniform interpolation SR reconstruction results by (a) nearest neighbor interpolation, (b) bilinear interpolation, (c) nonuniform interpolation using four LR images, and (d) debluring part

With registration of input images, a composite image on non-uniformly spaced sampling points is obtained. In the second step, uniformly spaced sampling points are obtained by direct or

iterative reconstruction procedure. After getting a high-resolution image with non-uniform interpolation, restoration is carried out to remove blurring.

3.3.2 Frequency Domain Approach:

This approach was proposed by Tsai and Huang where aliasing in the low-resolution images is used to reconstruct high resolution image. The relationship between low resolution images and the high-resolution image is described by them using relative motion between the low-resolution images. This approach is based on following three principles, i) the shifting property of Fourier transform ii) the aliasing relationship between the continuous Fourier transform (CFT) of an original HR image and the discrete Fourier transform (DFT) of observed LR images iii) the assumption that an original HR image is band-limited It is thus possible to formulate the system equation relating the aliased DFT coefficients of the observed low- resolution images to a sample of the CFT of an unknown image.

3.3.3 Regularized Image Reconstruction Approach:

The super-resolution image reconstruction approach can be an ill-posed problem because of an insufficient number of low-resolution images and ill-conditioned blur operators. Regularization is the procedure adopted to stabilize the inversion of ill-posed problem. This is achieved by imposing prior knowledge on the solution. There are two types of approaches, viz. i) deterministic approach ii) stochastic approach.

3.3.4 Super-Resolution

Super-resolution is based on the idea that a combination of low resolution (noisy) sequence of images of a scene can be used to generate a high-resolution image or image sequence. Thus, it attempts to reconstruct the original scene image with high resolution given a set of observed images at lower resolution. The general approach considers the low resolution images as resulting from resampling of a high-resolution image. The goal is then to recover the high resolution image which when resampled based on the input images and the imaging model, will produce the low resolution observed images. Thus the accuracy of imaging model is vital for super-resolution and an incorrect modeling, say of motion, can actually degrade the image further. The observed images could be taken from one or multiple cameras or could be frames of a video sequence. These images need to be mapped to a common reference frame. This process is registration. The super-resolution procedure can then be applied to a region of interest in the aligned composite image. The key to successful super-resolution consists of accurate alignment i.e., registration and formulation of an appropriate forward image model.

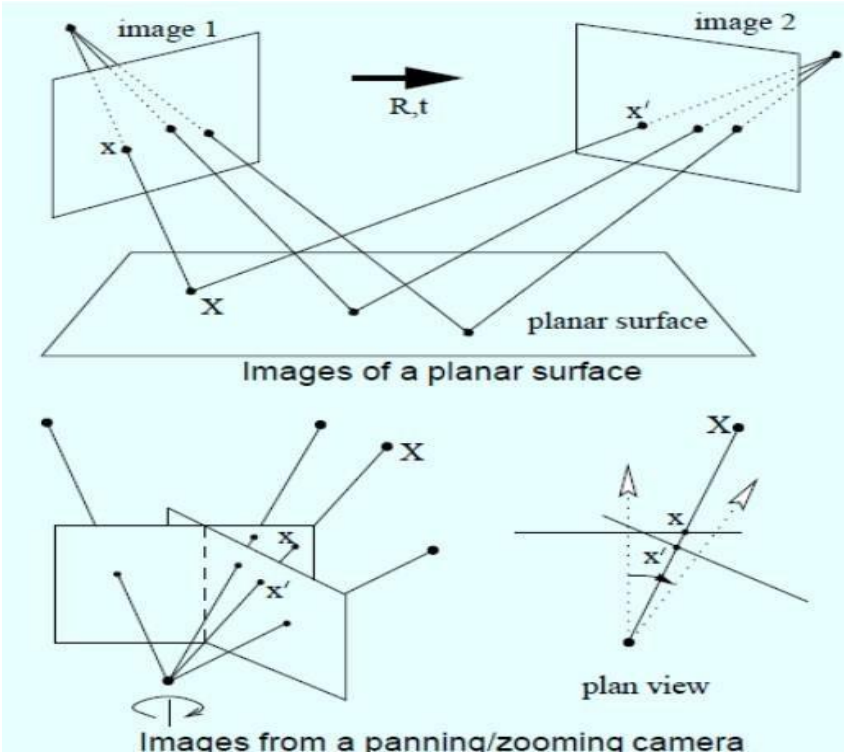

Fig 3.4 Two scenarios for homography

3.4. Result

Once an image has been captured at a particular resolution, it's not possible to increase its resolution without affecting its quality. While it's completely possible to add pixels by increasing the dimensions of the image in software, this is essentially a process of guesswork using existing pixels, rather than something that can retroactively add details that weren't there to begin with. So, once again, the best thing to do is to make sure you're capturing images at the highest-quality settings available on your camera. It should be clear which option this is: it will typically be marked 'Large', or with the specific megapixel count that corresponds to the sensor inside it.

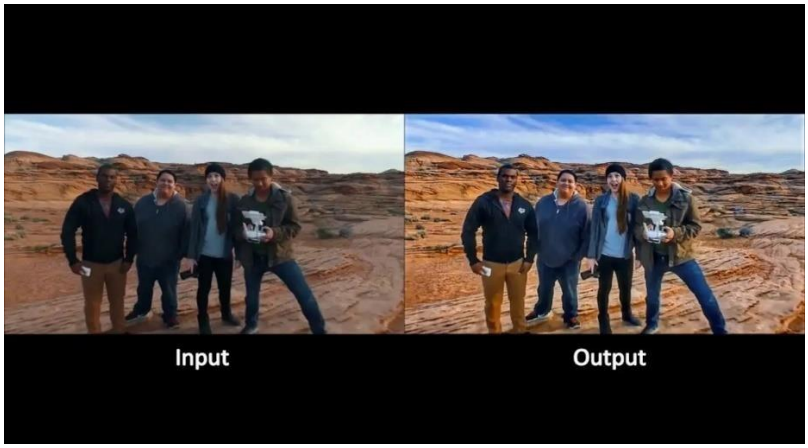

Fig 3.5 Input and Output

CONCLUSION

In most of the applications higher resolution is desired to get detailed information of the captured image. This can be achieved either by better image sensors or advanced optics. But it has cost implications and also hardware limitations. We can opt signal processing, specially, image processing to overcome these limitations of the sensors and optics manufacturing technology. An economical and effective solution to obtain a HR image from low resolution images is the use of image super resolution algorithms. . The super resolution methods combine the details of low-quality multiple images to extract information on unknown pixels of high-quality image. Thus, the fusion of information from multiple LR images enables the reconstruction of HR image easier as compared to single LR image. This is because multiple LR images contain different information and this additional information can be exploited to obtain a HR image. Thus, it remains a challenging task to reconstruct a HR image from single LR image which has less amount of information at hand. The proposed work is primarily concentrated on single image super resolution.

APPENDICES

CODE

```
import argparse from

pydoc import locate from

lib import utils import

cv2

from lib import AlexNet import lasagne
from scipy import optimize import
argparse from PIL import Image from
pydoc import locate from lib import
activations def def_feature(layer='conv4',
up_scale=4):    print('COMPILING...')    t
= time()    x = T.tensor4()    x_t =
AlexNet.transform_im(x)
   x_net = AlexNet.build_model(x_t, layer=layer, shape=(None, 3, 64, 64), up_scale=up_scale)
AlexNet.load_model(x_net, layer=layer)    x_f = lasagne.layers.get_output(x_net[layer],
deterministic=True)
   _ftr = theano.function(inputs=[x], outputs=x_f)    print('%.2f
seconds to compile _feature function' % (time() - t))    return _ftr
def def_predict(model_P):    print('COMPILING...')    t = time()
x = T.tensor4()    z = model_P(x)
   _predict = theano.function([x], [z])    print('%.2f seconds to compile _predict function' %
(time() - t))    return _predict def def_invert_models(gen_model, layer='conv4', alpha=0.002):
bfgs_model = def_bfgs(gen_model.model_G, layer=layer, npx=gen_model.npx, alpha=alpha)
ftr_model = def_feature(layer=layer)    predict_model = def_predict(gen_model.model_P)
return gen_model, bfgs_model, ftr_model, predict_model def predict_z(gen_model, _predict,
ims, batch_size=32):    n = ims.shape[0]    n_gen = 0    zs = []    n_batch = int(np.ceil(n /
float(batch_size)))    for i in range(n_batch):
```



```
      imb = gen_model.transform(ims[batch_size * i:min(n, batch_size * (i + 1)), :, :, :])
zmb = _predict(imb)        zs.append(zmb)        n_gen += len(imb)        zs =
np.squeeze(np.concatenate(zs, axis=0))        if np.ndim(zs) == 1:        zs =
zs[np.newaxis, :]        return zs if __name__ == "__main__":
   args = parse_args()    if not args.model_file:  # if the model file is not
specified        args.model_file = './models/%s.%s' % (args.model_name,
args.model_type)    if not args.output_image:  # if the output image path is not
specified
```

```
        args.output_image = args.input_image.replace('.png', '_%s.png' % args.solver)

    for arg in vars(args):
        print('[%s] =' % arg, getattr(args, arg))

    # read a single image          im =
Image.open(args.input_image)
    [h, w] = im.size          print('read image: %s (%dx%d)' %
(args.input_image, h, w))
    # define the theano models              model_class =
locate('model_def.%s' % args.model_type)
    gen_model          =          model_class.Model(model_name=args.model_name,
model_file=args.model_file, use_predict=True)
    invert_models = def_invert_models(gen_model, layer='conv4', alpha=0.002)
    # pre-processing steps      npx =
gen_model.npx      im =
im.resize((npx, npx))      im =
np.array(im)      im_pre =
im[np.newaxis, :, :, :]
    # run the model      rec, _, _ = invert_images_CNN_opt(invert_models, im_pre,
solver=args.solver)
    rec = np.squeeze(rec)
    rec_im = Image.fromarray(rec)      # resize the
image (input aspect ratio)      rec_im =
rec_im.resize((h, w))      print('write result to %s'
% args.output_image)
rec_im.save(args.output_image)
app.setWindowIcon(QIcon('pics/logo.png'))
window.setWindowTitle('Interactive GAN')
window.show()      app.exec_()
```


SCREENSHOTS


```
def def_feature(layer='conv4', up_scale=4):
    print('COMPILING...')
    t = time()
    x = T.tensor4()
    x_t = AlexNet.transform_im(x)
    x_net = AlexNet.build_model(x_t, layer=layer, shape=(None, 3, 64, 64), up_scale=up_scale)
    AlexNet.load_model(x_net, layer=layer)
    x_f = lasagne.layers.get_output(x_net[layer], deterministic=True)
    _ftr = theano.function(inputs=[x], outputs=x_f)
    print('%.2f seconds to compile _feature function' % (time() - t))
    return _ftr
def def_predict(model_P):
    print('COMPILING...')
    t = time()
    x = T.tensor4()
    z = model_P(x)
    _predict = theano.function([x], [z])
    print('%.2f seconds to compile _predict function' % (time() - t))
    return _predict
def def_invert_models(gen_model, layer='conv4', alpha=0.002):
    bfgs_model = def_bfgs(gen_model.model_G, layer=layer, npx=gen_model.npx, alpha=alpha)
    ftr_model = def_feature(layer=layer)
    predict_model = def_predict(gen_model.model_P)
    return gen_model, bfgs_model, ftr_model, predict_model
```

```
def predict_z(gen_model, _predict, ims, batch_size=32):
    n = ims.shape[0]
    n_gen = 0
    zs = []
    n_batch = int(np.ceil(n / float(batch_size)))
    for i in range(n_batch):
      imb = gen_model.transform(ims[batch_size * i:min(n, batch_size * (i + 1)), :, :, :])
        zmb = _predict(imb)
        zs.append(zmb)
        n_gen += len(imb)
        zs = np.squeeze(np.concatenate(zs, axis=0))
        if np.ndim(zs) == 1:
          zs = zs[np.newaxis, :]
        return zs
if __name__ == "__main__":
    args = parse_args()
    if not args.model_file:  # if the model file is not specified
        args.model_file = './models/%s.%s' % (args.model_name, args.model_type)
    if not args.output_image:  # if the output image path is not specified
        args.output_image = args.input_image.replace('.png', '_%s.png' % args.solver)

    for arg in vars(args):
        print('[%s] =' % arg, getattr(args, arg))
```


```
def predict_z(gen_model, _predict, ims, batch_size=32):
    n = ims.shape[0]
    n_gen = 0
    zs = []
    n_batch = int(np.ceil(n / float(batch_size)))
    for i in range(n_batch):
      imb = gen_model.transform(ims[batch_size * i:min(n, batch_size * (i + 1)), :, :, :])
        zmb = _predict(imb)
        zs.append(zmb)
        n_gen += len(imb)
        zs = np.squeeze(np.concatenate(zs, axis=0))
        if np.ndim(zs) == 1:
          zs = zs[np.newaxis, :]
        return zs
if __name__ == "__main__":
    args = parse_args()
    if not args.model_file:  # if the model file is not specified
        args.model_file = './models/%s.%s' % (args.model_name, args.model_type)
    if not args.output_image:  # if the output image path is not specified
        args.output_image = args.input_image.replace('.png', '_%s.png' % args.solver)

    for arg in vars(args):
        print('[%s] =' % arg, getattr(args, arg))
```


BIBLIOGRAPHY

1. List of Super-Resolution publications http://decsai.ugr.es/~jmd/superresolution/publications.html [Date accessed: 31/03/2011]
2. Super-resolution SD-to-HD up-converter super-resolution (Avarex.ru) http://www.youtube.com/watch?v=wxCleSGnji8 [Date accessed: 31/03/2011]
3. YUV Super Resolution vs plain static upscaling of video http://www.youtube.com/watch?v=181c6DxDs6k [Date accessed: 31/03/2011]
4. SR Overview (Visual Geometry Group) http://www.robots.ox.ac.uk/~vgg/research/SR/ [Date accessed: 31/03/2011