Computer Graphics: Converting Images To hand Drawn Sketches Using Novel Orthogonal Gaussian Lattice Method

ANISH SACHDEVA

DELHI TECHNOLOGICAL UNIVERSITY (DTU)

DTU/2K16/MC/13

Details

- Project Title: Converting Images To hand Drawn Sketches Using Novel Orthogonal Gaussian Lattice Method
- Subject: B Tech Project – I (MC-401)
- Project Supervisor 1: Dr. S. Sivaprasad Kumar (MC)
- Project Supervisor 2: Dr. Rajiv Kapoor (ECE)
- Candidate Name: Anish Sachdeva
- Candidate Roll No: DTU/2K16/MC/013
- Project Link: https://github.com/anishLearnsToCode/image2sketch

Introduction

Introducing a new method of feature extraction from Images ; "The Orthogonal Gaussian Lattice Method"

This new feature extraction method canbe used everywhere current feature extraction is used:

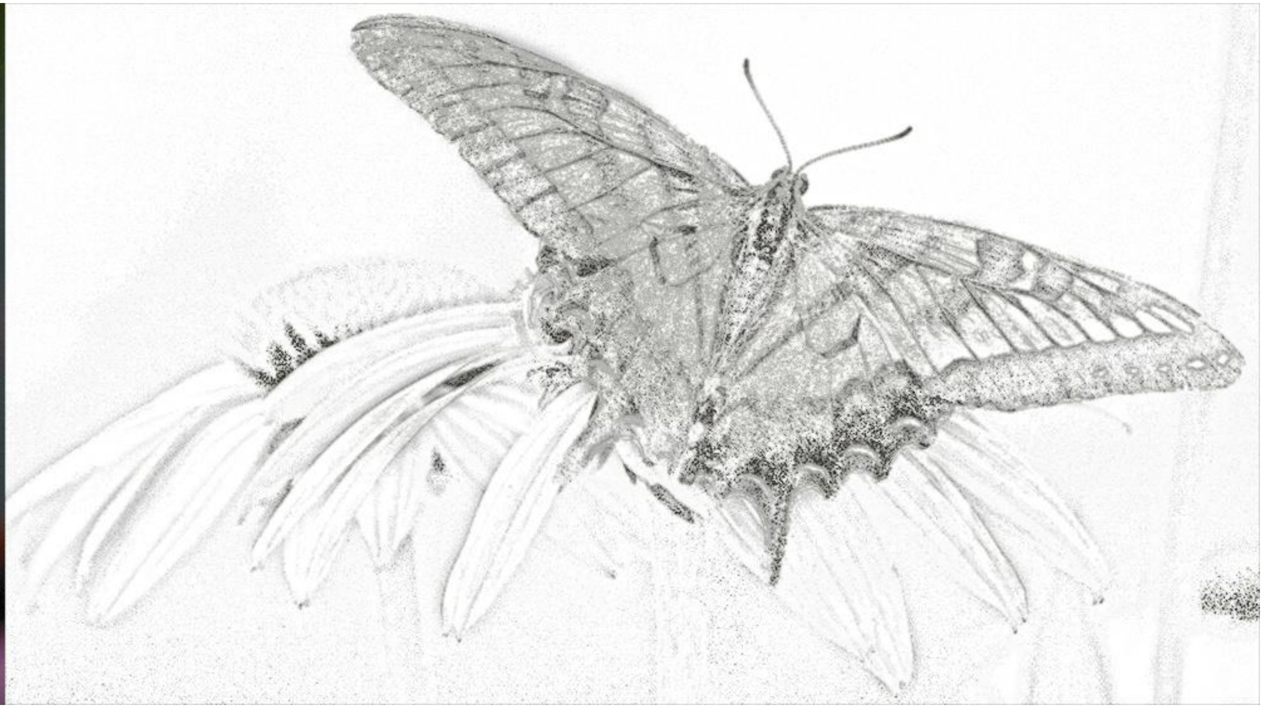
1. ML/Deep learning applications
2. Object Identification
3. Object tracking
4. Anomaly Detection
5. Computer Graphics

Application Showcased in this Project

In this this project I use my feature extraction method for the folllowing Application in Computer Graphics:

Creating a hand drawn sketch composite from a Given image *I*.

Examples

Current Methods For Creating a Sketch Composite From Images

1. Basic Canny Edge Detection
2. Fixed Texture Application
3. Guassian Blur/Blend Method (Current State-of-the-Art Method (SOA))
4. Novel Gaussian Lattice Method

Comparison Between Methods

Canny Edge Detection

Algorithm 1 Canny Edge Detection Algorithm for a Given Image I

$kernel \leftarrow$ 5x5 Gaussian Kernel

$I \leftarrow I * kernel$

Create Sobel Kernel to calculate Image Derivatives I_x and I_y

$$k_x \leftarrow \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$k_y \leftarrow \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

Compute the Derivatives I_x and I_y using k_x and k_y

$I_x \leftarrow I * k_x$

$I_y \leftarrow I * k_y$

Compute Magnitude G and angle Θ as

$G = \sqrt{I_x^2 + I_y^2}$

$\Theta = \arctan \frac{I_y}{I_x}$

We now perform Non Maximum Suppression to reduce the variation in Edge Thickness

for all pixels p in I **do**

  for all permitted angles θ in $0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°, 360°$ **do**

    Threshold all pixels with this gradient angle to 255 and the rest to 0

  end for

end for

$result \leftarrow I$

return $result$

Fixed Texture Application

Algorithm 2 Applying Texture Mask on Image I

Require: The Image I
Require: The mask M
Reshape the Image I to match the ratio of M, crop the image if you must
$I \leftarrow$ Grayscale of I
$result \leftarrow I \cdot M$
return $result$

Gaussian Blur Blend Technique

The steps are very simple,

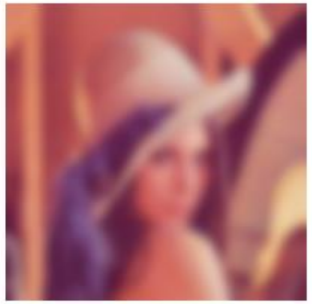
1. Take the Image I.
2. Take the Grayscale of this Image I.
3. Take the Negative of the grayscale obtained in step 2.
4. Apply a Gaussian Blur to the Negative Obtained in Step 3.
5. Blend the grayscale image from step 2 with the blurred negative from step 4 using a color dodge (5).

Guassian Blur

$$G(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

(a) Standard Lenna Image with No Gaussian Blur

(b) Gaussian Blur with $\sigma = 1$



(c) Gaussian Blur with $\sigma = 10$

(d) Gaussian Blur with $\sigma = 50$

Algorithm for Gaussian Blur Blend

Algorithm 3 Creating the Sketch Composite from I using Gaussian Blur and Blend Method

Require: The Image I

Require: *Sketch Density*: A Hyperparameter which will be used in The Gaussian Blurring Kernel Size and will decide the number of sketch lines to appear in the final result.

We create an Kernel of odd size, for convolution

Kernel Size $\leftarrow 2 * (\mathit{Sketch\ Density}, \mathit{Sketch\ Density}) + 1$

$J \leftarrow$ Grayscale of Image I

$B \leftarrow$ Gaussian Blur of J with kernel of size $k =$ Kernel Size

We divide the Grayscale image with the Gaussian Blur of the Grayscale Image. The following is a pixel by pixel level operation which can be parallelized using a standard numerical matrix package.

$result \leftarrow J/B$

return $result$

Orthogonal Gauss Lattice Method

The steps performed for computing using this method are:

1. We Take 3 different Gaussian with different mean μ and std. deviation σ.

2. We compute the Grayscale of the Image I, therefore reducing the Image pixel value from 3-dimensional to 1 dimensional.

3. We normalize the pixel values in our grayscale Image.

4. We compute 3 Gaussian Inverses using the 3 different Gaussian we took initially from the grayscale Image.

5. Using the Gaussian Inverses we computed, we now take a sliding window of size w and compute deviation spread Vectors (explained later) between a central and surrounding pixel.

6. We take 3 different bounds, denoted in this project by a for the 3 different Gaussian.

7. We compute 3 Simple Graphs from the 3 Gaussian Inverse using the deviation spread vectors and connectivity parameters a we took in step 6.

8. We compute the Different Components in the 3 different Simple Graphs that we calculated in Step 7 and the separate components in a single Frame (Simple Graph) is called a Lattice.

9. We can vary the type of lattices we create, the density of lattices and change the different features we discover by changing our 3 initial Gaussian that we select and also changing the connectivity bound parameter a.

Gaussian Inverse

Hello

$$G(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

$$G^{-1}(y;\mu,\sigma) = \sigma * \sqrt{-2 * \log(y\sigma\sqrt{2\pi})} + \mu$$

$$G^{-1}(y;\mu,\sigma) = \mu + \begin{cases} 0 & y > \frac{1}{\sigma\sqrt{2\pi}} \\ \sigma * \sqrt{-2 * \log(y\sigma\sqrt{2\pi})} & \text{otherwise} \end{cases}$$

$$G_1 \leftarrow (\epsilon, \frac{4}{\sqrt{2\pi}})$$

$$G_1 \leftarrow (\epsilon, \frac{2}{\sqrt{2\pi}})$$

$$G_1 \leftarrow (\epsilon, \frac{1}{\sqrt{2\pi}})$$

Deviation Spread Ratio Vector

Algorithm 4 Calculating the Deviation vectors from a Given image I

Require: Image matrix I

I ← grayscale(I)
I ← I / 255
We create 3 Gaussian
$G_1 \leftarrow (\mu_1, \sigma_1)$
$G_2 \leftarrow (\mu_2, \sigma_2)$
$G_3 \leftarrow (\mu_3, \sigma_3)$
We compute the Inverse Gaussian of the Image with the 3 Gaussian
$IG_1 = InverseGaussian(I, G_1)$
$IG_2 = InverseGaussian(I, G_2)$
$IG_3 = InverseGaussian(I, G_3)$
We now create a 3×3 or $w \times w$ sliding window and slide over our image to compute the Deviation Vectors.
The Deviation Vector of the surrounding and central Pixel Value are the ratio of
the deviation spread if the 2 pixels with the 3 Gaussian.
for all windows w in I **do**
  for all surrounding pixels p_s in window w for central pixel p_c **do**
    deviation(p_c, p_s) = DeviationVector(p_c, p_s, IG_1, IG_2, IG_3)
  end for
end for
Here a single deviation vector is a 3×1 row vector
return DevaiationVectors as D

Computing The Simple Graph From Deviation Ratio Vectors

Algorithm 5 Computing the Simple Graphs from the Deviation Vectors D and Connectivity Bounds α

Require: Deviation vectors D
Require: Connectivity Bounds α

```
Create 3 Simple Graphs U_1, U_2 and U_3 with no. of vertices = number of pixels and no edges
for all deviation vectors d^(i) in D do
  for all ratios d_j^(i) in d^(i) do
    if d_j^(i) bounded by (α_j, 1/α_j) then
      Add an edge between the pixels for this deviation vector d^(i) in the Simple Graph U_j
    end if
  end for
end for
return Simple Graphs U_1, U_2 and U_3
```

Extracting Lattices From Graph

Algorithm 6 Computing the Lattices from the Graphs U_1, U_2 and U_3 using Standard Graph Theory Extracting Components List Algorithm

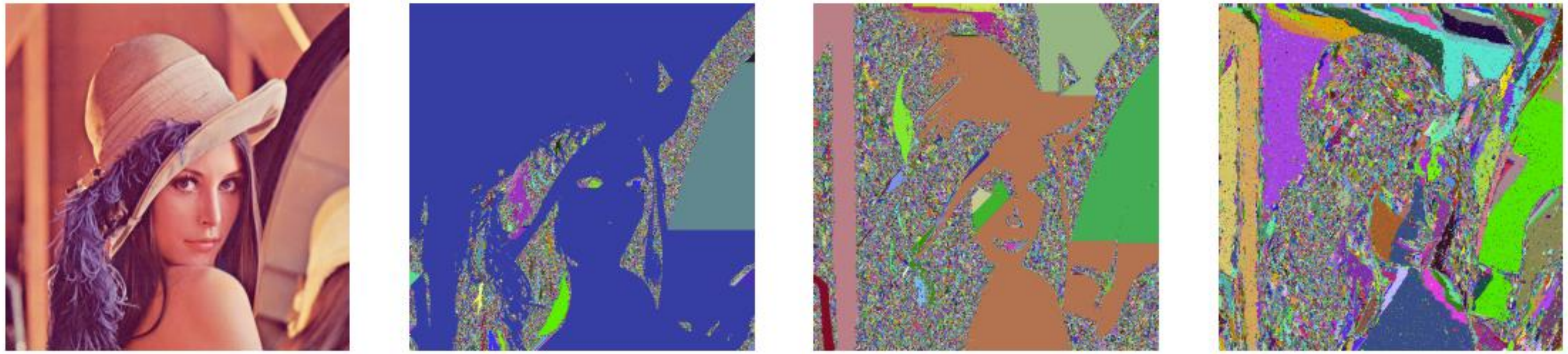
Require: Simple Graphs U_1, U_2 and U_3

$L_1 \leftarrow$ lattices from U_1

$L_2 \leftarrow$ lattices from U_2

$L_3 \leftarrow$ lattices from U_3

return Lattices L_1, L_2 and L_3



(a) Original Image

(b) Lattices in Image with G_1 and $\alpha = 0.98$

(c) Lattices in Image with G_2 and $\alpha = 0.98$

(d) Lattices in Image with G_3 and $\alpha = 0.98$

Figure Lattices in Different Simple Graphs with the connectivity parameter $<\alpha>= (0.98, 0.98, 0.98)$

Vertex Coloring In The Lattices

Algorithm 8 Computing The Vertex Shaded Image Given The Simple Graphs U_i for different Gaussian for image I

Require: Image I
Require: Simple Graphs U_i for $\{1, 2, 3\}$ for the 3 Gaussian
  Create 3 new Images J_1, J_2 and J_3 which will be the result
  for all Simple Graph U in U_i and resulting Image J in J_i **do**
    for all Vertices v in U **do**
      $pixel\_color \leftarrow PixelColorFromDegree(v.degree)$
      Assign pixel v in J color $pixel\_color$
    end for
  end for
  return Resulting Images J_1, J_2 and J_3

Results of Vertex Coloring

(a) Original Image

(b) Vertex Shading in G_1 with $\alpha = 0.86$

(c) Vertex Shading in G_2 with $\alpha = 0.90$

(d) Vertex Shading in G_3 with $\alpha = 0.94$

Linear Fusion Of Multi-Guassian Technique

Algorithm 9 Computing The Sketch Composite from Image I

Require: Image I
Require: Gaussian Parameters (μ_1, σ_1), (μ_2, σ_2) and (μ_3, σ_3)
Require: Connectivity Parameters $< \alpha >$
Require: Fusion Weights W
Compute The Deviation Vectors Using Algorithm-4
Compute The Simple Graphs from the Deviation Vectors Using Algorithm-5
$< VS > \leftarrow$ The Vertex Shaded Images From The Simple Graphs using Algorithm-8
$R_{gb} \leftarrow$ Gaussian Blended Result from Algorithm-3
return $w_1 \cdot VS_1 + w_2 \cdot VS_2 + w_3 \cdot VS_3 + w_{gb} \cdot R_{gb}$

Results

(a) Original Image

(b) Weighted Result
$W = (0.03, 0.03, 0, 0.94)$

(c) Weighted Result
$W = (0.03, 0.03, 0.15, 0.79)$

(d) Weighted Result
$W = (0.03, 0.03, 0.30, 0.64)$

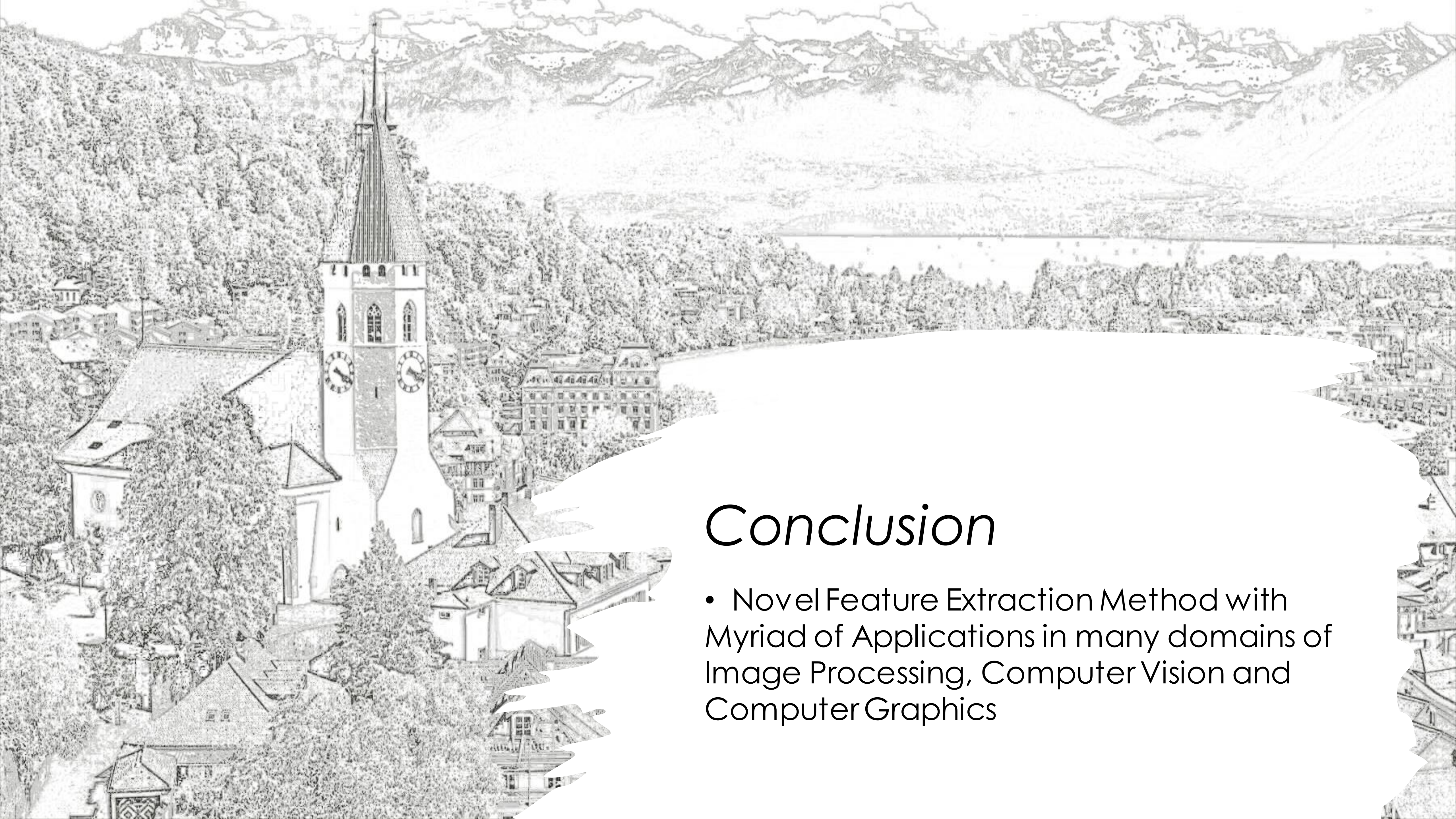
(e) Weighted Result
$W = (0.03, 0.03, 0.45, 0.49)$

(f) Weighted Result
$W = (0.03, 0.03, 0.60, 0.34)$

Figure 30: Weighted Mean Results with Gaussian Parameters $G_1 = (\epsilon, \frac{4}{\sqrt{2\pi}}), G_1 = (\epsilon, \frac{2}{\sqrt{2\pi}}), G_1 = (\epsilon, \frac{1}{\sqrt{2\pi}})$ and $\alpha = (0.86, 0.94, 0.94)$

Future Scope

1. Novel Feature Extraction
2. Object Detection in Images
3. Object Tracking in Image Sequences
4. Anomaly Detection in Image Sequences



Conclusion

- Novel Feature Extraction Method with Myriad of Applications in many domains of Image Processing, Computer Vision and Computer Graphics