B. Tech Major Project

Under Guidance of Dr. S. Sivaprasad Kumar (MCE) &
Dr. Rajiv Kapoor (ECE)

Anomaly Detection in Image Sequences

Delhi Technological University

Anish Sachdeva
DTU / 2K16 / MC / 13

1 Introduction

Anomaly Detection in Image Sequences or videos is a very important problem and one which has many different approaches, but no definitive solution which delivers high accuracy. This is one of the leading open problems in unsupervised learning.

The following is an novel approach for solving said problem.

2 The algorithm

We are given an image Sequence or a Video V denoted by a time dependent function $f(x, y, t)$. We start by building a Lattice for each individual frame of this video. We build a lattice for a singe image frame using the following steps:

We firstly convert the image to grayscale, and the normalize it. We take 3 Gaussian with different standard deviation, where one Gaussian will have less deviation σ and will be used to observe small changes in the pixel values, whereas the flatter Gaussian's with larger σ will be used to observe smaller changes in pixel values.

We find the Gaussian Inverse of the image using these 3 Gaussian and then use these inverses to compute the z-score for each pixel for each Gaussian. We now take a sliding window of size 3 and convolute on our Image to compute the deviation spread vector for every surrounding pixel value p_s for a given central pixel p_c. The deviation spread ratio is computed by dividing the z-score for the surrounding pixel by the z-score for the central pixel.

After computing the deviation spread vectors, we will compute the Lattice for the given image using the deviation vectors. We will create a simple graph where every pixel is considered as a vertex and any 2 adjacent pixels will be connected *iff* their deviation spread vector lies in a particular range.

3 Computing the Gaussian Inverses of an Image

A Gaussian $G(x, \mu, \sigma)$ is denoted as

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

and the inverse Gaussian is denoted by

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

Let us have a look at a few Gaussian Inverse of a sample image. The Gaussian inverse may not always be defined for all pixel values, as the highest point in a Gaussian is $\frac{1}{\sigma\sqrt{2\pi}}$ and for 2 of the Gaussian for which we have taken large deviation (σ), for some pixel values an inverse might not exist and hence in the Inverse Gaussian below we take binary images, where 1 represents pixel whose inverse exists.

Figure 1: Lenna: Original Image I

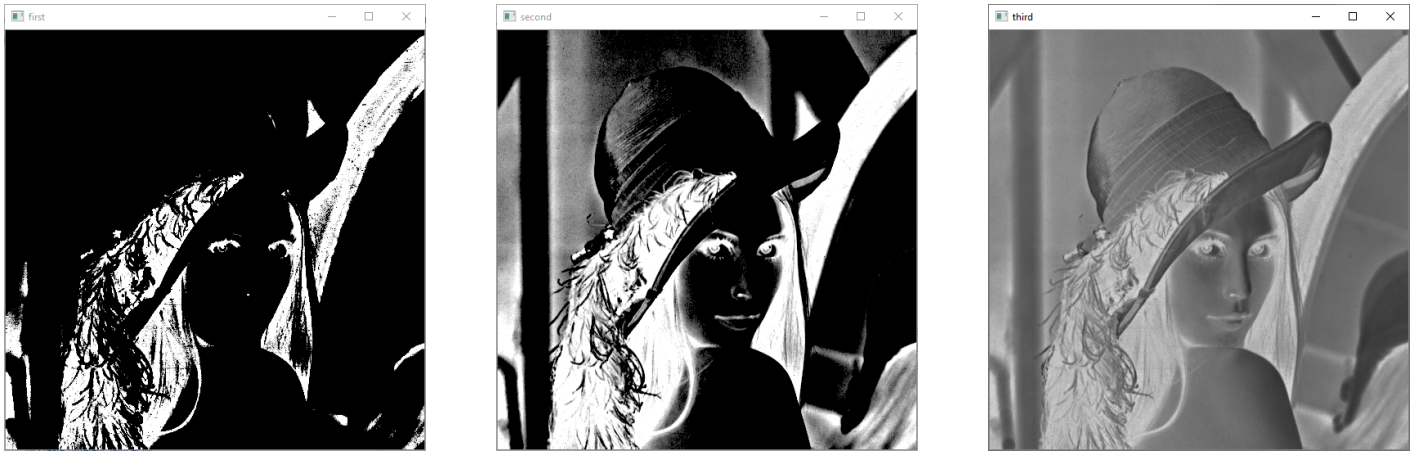


Figure 2: The Gaussian Inverse of the Image with the 3 Gaussian

Figure 3: A Dolphin Jumping Over a Wave

Figure 4: The Gaussian Inverse of the Image with the 3 Gaussian

4 Computing the Deviation Vectors

We use the Gaussian inverse to compute the Deviation Vectors of the image pixel values.

Algorithm 1 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 G
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

5 Creating the Lattice

We will now create a Lattice, which will basically be an un-directed Simple Graph where each pixel will be represented as a Vertex and we will add edges between pixels on the basis of the deviation spreads we have computed. This will form components inside the Lattice, and we will compute all these disconnected components in the lattice.

Algorithm 2 Computing the Lattice from the Deviation Vectors D

Require: Deviation vectors D
  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 (l_j, u_j) **then**
        Add an edge between the pixels for this deviation vector $d^{(i)}$ in the Lattice
      end if
    end for
  end for
  return Lattice L

6 Disconnected Components in a Lattice

We now compute the individual disconnected components from this lattice and plot a few images to show results of this algorithm with the Lower Bound (l_i) and Upper Bound (u_i) as $(0.86, 0.94, 0.94)$ and $(1.162, 1.063, 1.063)$.

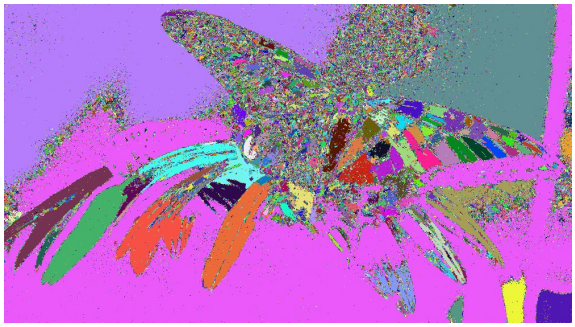
All components have been given different colors for representation purposes and the colors have no significance.

(a) Image of a Rose Flower

(b) Corresponding Lattice

Figure 5: Disjoint Components in a Lattice

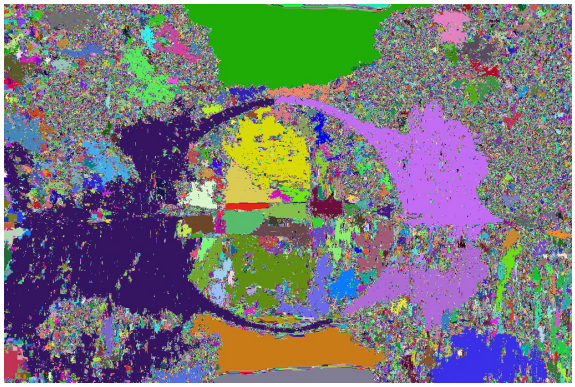
(a) Image of a Butterfly



(b) Corresponding lattice

Figure 6: Disjoint Components in a Lattice

(a) Image of a Bridge



(b) Corresponding Lattice

Figure 7: Disjoint Components in a Lattice

(a) Standard Lenna Image

(b) Corresponding Lattice

Figure 8: Disjoint Components in a Lattice

(a) Dolphin Jumping over a Wave

(b) Corresponding Lattice

Figure 9: Disjoint Components in a Lattice