



# Deteksi Tepi Menggunakan Metode Laplacian of Gaussian Pada Citra Bola Futsal

Endri Dwi Prasetyo

Prodi Teknik Informatika, Universitas Budi Darma, Medan, Indonesia

Email@endri@gmail.com

**Abstrak**—Bola adalah sebuah benda bulat yang dipakai sebagai alat olahraga atau permainan. Umumnya bola terisi dengan udara. Terdapat bermacam-macam bola salah satunya adalah bola futsal. Bola futsal adalah bola yang sama dengan bola sepak tetapi yang membedakan adalah ukuran bola, berat bola dan teksturnya. Sering kali bola futsal yang baru tidak bisa dimainkan karena ukuran bola tidak sesuai atau bola cacat (Bola baling). Akibatnya saat terselenggaranya turnamen atau liga, ditengah pertandingan pemain sering sekali meminta pergantian bola dengan wasit karena bola yang dimainkan kurang nyaman. Bola baling memang sering menjadi kendala bagi pemain. Hal ini membuat para pemain kesulitan dalam mengontrol bola, *passing* dan menendang bola tersebut. Maka dari itu perlu dilakukan penelitian untuk mengidentifikasi kecacatan bola menggunakan citranya.

**Kata Kunci:** Deteksi Tepi, Laplacian of Gaussian, Citra, Bola Futsal

**Abstract**—A ball is a round object that is used as a sport or game. Generally the ball is filled with air. There are various kinds of balls, one of which is a futsal ball. A futsal ball is the same ball as a soccer ball but what distinguishes it is the size of the ball, the weight of the ball and its texture. Often times new futsal balls cannot be played because of the wrong size of the ball or the ball is defective (propeller ball). As a result, when a tournament or league is held, in the middle of a match players often ask for the replacement of the ball with the referee because the ball is not comfortable. The propeller ball is often an obstacle for players. This makes it difficult for the players to control the ball, pass and kick the ball. Therefore it is necessary to conduct research to identify ball defects using its image.

**Keywords:** Edge Detection, Laplacian of Gaussian, Image, Futsal Ball

## 1. PENDAHULUAN

Bola adalah sebuah benda bulat yang dipakai sebagai alat olahraga atau permainan. Umumnya bola terisi dengan udara. Terdapat bermacam-macam bola salah satunya adalah bola futsal. Bola futsal adalah bola yang sama dengan bola sepak tetapi yang membedakan adalah ukuran bola, berat bola dan teksturnya. Sering kali bola futsal yang baru tidak bisa dimainkan karena ukuran bola tidak sesuai atau bola cacat (Bola baling). Akibatnya saat terselenggaranya turnamen atau liga, ditengah pertandingan pemain sering sekali meminta pergantian bola dengan wasit karena bola yang dimainkan kurang nyaman. Bola baling memang sering menjadi kendala bagi pemain. Hal ini membuat para pemain kesulitan dalam mengontrol bola, *passing* dan menendang bola tersebut. Maka dari itu perlu dilakukan penelitian untuk mengidentifikasi kecacatan bola menggunakan citranya.

Dalam mendeteksi kecacatan bola biasanya dilakukan dengan cara panca indra saja, namun cara itu dianggap kurang efektif dan sering kali terjadi kesalahan. Maka dari itu perlu dilakukan deteksi tepi pada citra bola futsal untuk mendeteksi kecacatan bola futsal itu sendiri. Deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra. Tepi-tepi ini akan menandai bagian detail citra. Tepi-tepi pada gambar tersebut terletak pada titik-titik yang memiliki perbedaan tinggi. Dengan perbedaan tinggi tersebut tercipta suatu pola atau guratan yang membentuk suatu objek dapat diperoleh menggunakan *High Pass Filter* (HPF), tujuannya adalah untuk menandai bagian yang menjadi detail citra dan untuk memperbaiki detail dari citra yang kabur yang terjadi akibat error atau adanya efek dari proses akuisisi citra.

Banyak metode yang bisa digunakan untuk melakukan penyelesaian permasalahan deteksi tepi, diantaranya adalah *Operator Prewitt*, *Operator Sobel*, *Operator Kompas*, *Operator Roberts*, *Operator turunan kedua* (*laplacian*) dan *Laplacian of Gaussian*. *Laplacian of Gaussian* adalah operator deteksi tepi orde kedua yang boleh dikatakan sukses untuk mengurangi tingkat sensitif pada terhadap derau. *Laplacian of Gaussian* terbentuk dari proses gaussian yang diikuti operasi *Laplace*. Fungsi *Gaussian* akan mengurangi derau sedangkan *Laplacian mask* meminimalisasi kemungkinan kesalahan deteksi tepi (Susanto, 2013). Operator *Laplace* mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam. Hal ini dikarenakan *zero-crossing* sendiri yang mendefinisikan lokasi tepi. Pada tepi yang curam, turunan keduanya mempunyai *zero-crossing*, yaitu titik dimana terdapat pergantian tanda nilai turunan kedua sedangkan pada tepi yang landai tidak terdapat *zero crossing*.

Berdasarkan penelitian terdahulu yakni “Deteksi tepi pada citra daun kopi menggunakan metode *Laplacian Of Gaussian* oleh Tri Septia Prihartini menghasilkan kesimpulan bahwa proses pendeteksian tepi dengan metode LOG dilakukan melalui beberapa langkah yaitu merubah citra RGB menjadi citra *grayscale* dan kemudian melakukan operasi metode LOG pada citra *grayscale*” (Prihartini & Andono, n.d.) dan “Pendeteksian tepi citra CT Scan Menggunakan *Laplacian Of Gaussian* oleh Nurhasanah menghasilkan kesimpulan bahwa dengan metode ini cukup efektif digunakan karena menghasilkan citra bagian-bagian otak yang dapat terbedakan dengan jelas” (Tomography & Tomography, 2012) maka dari itu penulis Mengusulkan Penelitian Untuk Menerapkan Metode *Laplacian Of Gaussian* Pada Citra Bola Futsal.



## 2. METODOLOGI PENELITIAN

### 2.1 Deteksi Tepi

Deteksi Tepi adalah suatu himpunan piksel yang terhubung dan terletak pada batas dua area. Deteksi tepi berfungsi untuk memperoleh tepi objek. Deteksi tepi memanfaatkan perubahan nilai intensitas yang drastic pada batas dua area (Susanto, 2013). Suatu garis atau edge didefinisikan sebagai sederetan piksel yang mempunyai intensitas antara piksel permulaan dan piksel akhir. Definisi *edge* disini adalah kumpulan dari titik-titik atau *plane point set*. Karena jarak antara titik-titik yang sangat berdekatan membentuk *edge* dalam suatu objek.

### 2.2 Bola Futsal

Futsal adalah permainan bola yang dimainkan oleh dua tim, yang masing-masing beranggotakan lima orang. Tujuannya adalah memasukkan bola ke gawang lawan, dengan memanipulasi bola menggunakan kaki ("FUTSAL," n.d.). Selain lima pemain utama, setiap regu juga diizinkan memiliki pemain cadangan. Tidak seperti permainan sepakbola dalam ruangan lainnya, lapangan futsal dibatasi garis, bukan net atau papan. Spesifikasi Bola Futsal

- Ukuran: 4
- Keliling: 62-64 cm
- Berat: 390-430 gram
- Lambungan: 55-65 cm pada pantulan pertama
- Bahan: kulit atau bahan yang cocok lainnya (yaitu bahan tak berbahaya)

### 2.3 Metode Laplacian of Gaussian

*Laplacian of Gaussian* adalah salah satu operator deteksi tepi yang dikembangkan dari turunan kedua. *Laplacian of Gaussian* terbentuk dari proses *Gaussian* yang diikuti operasi *laplace*. Fungsi *Gaussian* akan mengurangi derau sedangkan *Laplacian mask* meminimalisasi kemungkinan kesalahan deteksi tepi. Operator Laplace mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam. Hal ini dikarenakan *zero-crossing* sendiri yang mendefinisikan lokasi tepi. Pada tepi yang curam, turunan keduanya mempunyai *zerocrossing*, yaitu titik dimana terdapat pergantian tanda nilai turunan kedua sedangkan pada tepi yang landai tidak terdapat *zero-crossing*. Tepi dari suatu objek pada image dimodelkan dengan menentukan spesifikasi dari unsure posisi, orientasi dan nilai intensitas yang konstan. Operator ini bekerja dengan mencari nilai nol pada turunan kedua dari citra, karena ketika turunan pertama terdapat pada nilai maksimum maka turunan kedua akan menghasilkan nilai nol (Susanto, 2013).

*Laplacian of Gaussian* (LoG) adalah operator deteksi tepi orde kedua yang boleh dikatakan sukses untuk mengurangi tingkat sensitive terhadap derau. Hal ini disebabkan penggunaan fungsi Gaussian yang memuluskan citra dan berdampak pada pengurangan derau pada citra. Operator turunan kedua disebut juga operator Laplace. Operator Laplace mendeteksi lokasi tepi lebih akurat khususnya pada tepi yang curam. Metode *Laplacian of Gaussian* dapat mendeteksi tepi lebih akurat khususnya pada tepi yang curam. Selain itu, dapat dikatakan lebih akurat karena dapat mengurangi kemunculan tepi palsu, karena citra disaring terlebih dahulu dengan fungsi Gaussian. Turunan kedua fungsi dengan dua pengubah adalah:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (1)$$

Dengan menggunakan definisi hampiran selisih-mundur (*backward difference approximation*):

$$G_3(x) = \frac{\partial f(x,y)}{\partial x} = \frac{f(x,y) - f(x-\Delta x, y)}{\Delta x} \quad (2)$$

$$G_3(y) = \frac{\partial f(x,y)}{\partial y} = \frac{f(x,y) - f(x, y-\Delta y)}{\Delta y} \quad (3)$$

Maka :

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \quad (4)$$

$$\begin{aligned} G_1(G_3(x)) &= G_1(G_3(y)) \\ &= \frac{1}{\Delta x} G_1(f(x, y)) - G_1(f(x - \Delta x, y)) + \frac{1}{\Delta y} G_1(f(x, y)) - G_1(f(x, y - \Delta y)) \\ &= \frac{1}{\Delta x} \left\{ \frac{f(x+\Delta x, y) - f(x, y) - f(x, y) + f(x-\Delta x, y)}{\Delta x} \right\} \\ &\quad + \frac{1}{\Delta y} \left\{ \frac{f(x, y+\Delta y) - f(x, y) - f(x, y) + f(x, y-\Delta y)}{\Delta y} \right\} \\ &= \frac{f(x+\Delta x, y) - 2f(x, y) + f(x-\Delta x, y)}{(\Delta x^2)} \\ &\quad + \frac{f(x, y+\Delta y) - 2f(x, y) + f(x, y-\Delta y)}{(\Delta y^2)} \end{aligned}$$

Dengan mengasumsikan  $\Delta x = \Delta y = 1$ , maka diperoleh:



$$\begin{aligned}\Delta^2 f(x, y) &= f(x+1, y) - 2f(x, y) + f(x-1, y) + f(x, y+1) - 2f(x, y) + f(x, y-1) \\ &= f(x, y-1) + f(x-1, y) - 4f(x, y) + f(x+1, y) + f(x, y+1)\end{aligned}$$

Atau dapat dinyatakan sebagai *mask* :

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Selain *mask* di atas, masih ada dua hampiran operator Laplace yang lain, yaitu

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \text{ dan } \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

Kadang-kadang diinginkan memberi bobot yang lebih pada *pixel* tengah di antara *pixel* tetangganya. Operator Laplace yang digunakan untuk tujuan ini adalah

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

Operator Laplace termasuk ke dalam penapis lolos-tinggi sebab jumlah seluruh koefisiennya nol dan koefisiennya mengandung nilai negatif maupun positif. Kadangkala pendeteksian tepi dengan operator Laplace menghasilkan tepi-tepi palsu yang disebabkan oleh gangguan.

Fungsi  $\nabla^2 G(x, y)$  merupakan turunan kedua dari fungsi Gauss, kadang-kadang disebut juga fungsi *Laplacian of Gaussian (LoG)* atau fungsi tepi orang Mexico (*Mexican Hat*), karena bentuk kurvanya seperti topi Meksiko. Jadi, untuk mendeteksi tepi dari citra yang mengalami gangguan, kita dapat melakukan salah satu dari dua operasi ekuivalen di bawah ini:

1. Konvolusi citra dengan fungsi Gauss  $G(x, y)$ , kemudian lakukan operasi Laplacian terhadap hasilnya, atau
2. Konvolusi citra dengan penapis *LoG*.

Contoh penapis *LoG* yang berukuran 5 x 5:

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & -16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

### 3. HASIL DAN PEMBAHASAN

Dalam merancang sebuah sistem khususnya sistem yang berbasis aplikasi perlu dilakukan analisa. Analisa berguna untuk meminimalisir terjadinya kesalahan pada saat akan mendeteksi tepi citra. Analisa merupakan upaya untuk melakukan pemahaman tertentu terhadap sesuatu masalah yang dilakukan dalam pengkajian. Dalam mendeteksi suatu tepi citra mutlak dilakukan penelitian dan penganalisaan tentang tepi citra yang akan dideteksi, berikut beberapa analisa yang dilakukan untuk mendeteksi tepi menggunakan metode Laplacian of Gaussian.

Dalam proses pendeteksian tepi pada citra, langkah pertama yang kita lakukan adalah melakukan input citra, pada sub bab ini penulis menggunakan foto berukuran 300x300 piksel. Gambar yang diinput berupa jpeg. Setelah citra kita inputkan maka sistem akan melakukan proses pengambilan nilai *pixel* masing-masing dari Red, Green, dan Blue (RGB).



**Gambar 1.** Citra input ukuran 300x300

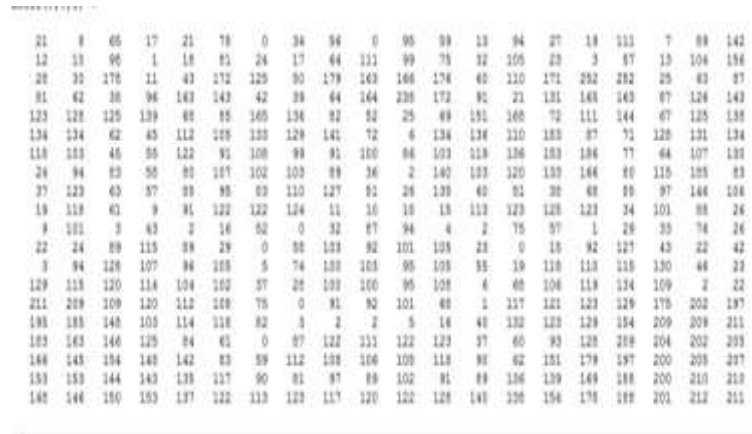
Data yang digunakan dalam penelitian ini adalah beberapa citra untuk pengujian dimana citra yang digunakan merupakan citra *grayscale* dengan *image size* 20x20*pixel*, tepi yang dideteksi adalah tepi luar, selanjutnya citra tersebut akan mengalami proses untuk mendapatkan tepi dan menghasilkan citra baru dengan ukuran 20x20*pixel* yang merupakan citra hasil (*Citra Output*). Adapun *flow process diagram* pengolahan data citra yang akan digunakan.



**Gambar 2.** Citra input ukuran 20x20

Citra *input* merupakan citra yang memiliki intensitas warna berkisar antara 0 sebagai nilai minimum sampai 255 yang merupakan nilai maksimum. Citra input yang memiliki ukuran  $20 \times 20 \text{ pixel}$  kemudian dikonversi ke dalam bentuk matriks  $20 \times 20 = 400$ , untuk masing-masing citra.

Contoh nilai dari citra *grayscale* dengan *image size*  $20 \times 20 \text{ pixel}$  yang dikonversi kedalam bentuk matriks sebagai berikut:



**Gambar 3.** Matriks citra  $20 \times 20$

### 3.1 Penerapan Metode *Laplacian of Gaussian*

*Filter Gaussian* tergolong sebagai *filter* lolos rendah yang didasarkan pada fungsi *gaussian*. Model dua dimensinya berupa:

$$G(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Dimana  $x$  dan  $y$  adalah posisi kordinat pada sumbu  $x$  dan  $y$ . Persamaan inilah yang dipakaisebagai dasar untuk menentukan nilai-nilai setiapelemen dalam *filter gaussian* yang akan dibuat. Misalkan dibuat kernel *filter* ukuran  $5 \times 5$  dan mengisi elemen/bobot  $g(x, y)$  pada matriks kernel *gaussian*.

Keterangan:

$e$  (*euler*) = 2,718281828 (konstanta)

$\sigma$  (*Standart deviasi*) = 2

Maka :

$$\begin{aligned} G(0,0) &= e^{-\frac{0^2 + 0^2}{2 \cdot 2^2}} \\ &= e^0 \\ &= 1 \end{aligned}$$

$$G(1,0) = G(0,1) = G(-1,0) = G(0,-1) = e^{(-1/4)} = 0,7788$$

$$G(1,1) = G(-1,1) = G(1,-1) = G(-1,-1) = e^{(-2/4)} = 0,6065$$

$$G(1,2) = G(2,1) = G(-1,-2) = G(-2,-1) = e^{(-5/4)} = 0,2865$$

$$G(2,0) = G(0,2) = G(-2,0) = G(0,-2) = e^{(-4/4)} = 0,3678$$

$$G(2,2) = G(2,-2) = G(-2,2) = G(-2,-2) = e^{(-8/4)} = 0,1353$$

Maka diperoleh nilai elemen atau bobot matrik kernel *gaussian* sebagai berikut:

**Tabel 1.** Elemen atau Bobot Matrik Kernel

x/y	-2	-1	0	1	2
-2	0,1353	0,2865	0,3678	0,2865	0,1353
-1	0,2865	0,6065	0,7788	0,6065	0,2865
0	0,3678	0,7788	1	0,7788	0,3678
1	0,2865	0,6065	0,7788	0,6065	0,2865
2	0,1353	0,2865	0,3678	0,2865	0,1353

Selanjutnya normalisasi nilai pembobotan setiap bobot dengan nilai terkecil dari nilai bobot. Nilai terkecil 0,1353 dan kemudian hasilnya dibulatkan ke atas). Dengan demikian, diperoleh hasil seperti berikut:

**Tabel 2.** Normalisasi Nilai Pembobotan

Nilai bobot	Pembagian bobot
1	$1/0,1353 = 7$
0,7788	$0,7788/0,1353 = 6$
0,6065	$0,6065/0,1353 = 4$
0,3678	$0,3678/0,1353 = 3$
0,2865	$0,2865/0,1353 = 2$
0,1353	$0,1353/0,1353 = 1$

Maka diperoleh matriks sebagai berikut:

**Tabel 3.** Hasil Elemen atau bobot matrik

x/y	-2	-1	0	1	2
-2	1	2	3	2	1
-1	2	4	6	4	2
0	3	6	7	6	3
1	2	4	6	4	2
2	1	2	3	2	1

Jumlah semua elemen nilai pembobot pada *filter* agar selang nilai intensitas tetap seperti semula. Berdasarkan matriks pada tabel 4.2 jumlah semua elemen nilai pembobot pada *filter* = 79. Berikut adalah *filter gaussian* hasil rancangannya.

**Tabel 4.** Filter Gaussian

1	2	3	2	1
2	4	6	4	2
3	6	7	6	3
2	4	6	4	2
1	2	3	2	1

$$g(x, y) = \frac{1}{79}$$

Untuk melakukan operasi perbaikan *noise* citra dengan metode *filter gaussian*. Operasi ini dilakukan dengan cara konvolusi, konvolusi seringkali melibatkan dalam operasi ketetanggaan *pixel*. Konvolusi pada citra sering disebut konvolusi 2 dimensi. Konvolusi 2 dimensi didefinisikan sebagai proses untuk memperoleh suatu *pixel* berdasarkan nilai *pixel* itu sendiri dan tetangganya, dengan melibatkan suatu matriks yaitu kernel yang mempresentasikan pembobotan. Penjelasan rumus yang digunakan dalam konvolusi filter Gaussian adalah sebagai berikut:

$$h(x, y) = f(x, y) * g(x, y) = \sum_{k=1}^M \sum_{l=1}^N f(k, l) \cdot g(x - k, y - l)$$

Keterangan:

$h(x, y)$ : gambar output

$f(x, y)$ : adalah gambar input

$g(x, y)$ : adalah filter gaussian

Jadi secara umum rumus diatas adalah jumlah dari perkalian antara *pixel* citra dengan filter gaussian dan hasilnya dibagi dengan jumlah dari matriks filter agar selang nilai intensitas tetap seperti semula. Untuk penjelasan proses konvolusi penulis membuat sebuah perumpamaan matriks citra *grayscale* yang terdapat pada gambar 4.3, dengan resolusi matriks 20x20 *pixel* yang akan dikonvolusikan dengan filter gaussian dengan ukuran matriks 5x5. Langkah selanjutnya adalah melakukan operasi konvolusi dengan cara menempatkan atau menumpangkan suatu filter atau kernel pada setiap *pixel* yang ditimpali, kemudian nilai rata-rata diambil dari hasil-hasil tersebut.

Pada proses pelaksanaan konvolusi kernel digeser sepanjang baris dan kolom dalam citra sehingga diperoleh nilai baru pada citra keluaran. Langkah-langkah konvolusi digambarkan sebagai berikut:

**Langkah 1:** Tempatkan *kernel* pada sudut kiri atas, kemudian hitung nilai *pixel* pada posisi (0,0) dari *kernel*



**Gambar 4.** Matriks citra 20x20 dengan 5 kernel



1	2	3	2	1
2	4	6	4	2
3	6	7	6	3
2	4	6	4	2
1	2	3	2	1

$$G(x,y) = (1 \times 21) + (2 \times 8) + (3 \times 65) + (2 \times 17) + (1 \times 21) + (2 \times 12) + (4 \times 10) + (6 \times 98) + (4 \times 1) + (2 \times 18) + (3 \times 28) + (6 \times 30) + (7 \times 178) + (6 \times 11) + (3 \times 43) + (2 \times 81) + (4 \times 62) + (6 \times 38) + (4 \times 96) + (2 \times 163) + (1 \times 123) + (2 \times 128) + (3 \times 125) + (2 \times 139) + (1 \times 68) = (21 + 16 + 195 + 34 + 21 + 24 + 40 + 408 + 584 + 4 + 180 + 1246 + 66 + 129 + 291 + 162 + 248 + 228 + 384 + 326 + 123 + 256 + 378 + 278 + 68) = 5710 : 79 = 72.278$$

Nilai *pixel* yang diubah adalah 178 menjadi 72

**Langkah 2:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (1,0) dari *kernel*.

8	65	17	21	78	1	2	3	2	1
10	98	1	18	81	2	4	6	4	2
30	178	11	43	172	3	6	7	6	3
62	38	96	163	143	2	4	6	4	2
128	125	139	68	85	1	2	3	2	1

$$G(x,y) = (1 \times 8) + (2 \times 65) + (3 \times 17) + (2 \times 21) + (1 \times 78) + (2 \times 10) + (4 \times 98) + (6 \times 1) + (4 \times 18) + (2 \times 81) + (3 \times 30) + (6 \times 178) + (7 \times 11) + (6 \times 43) + (3 \times 172) + (2 \times 62) + (4 \times 38) + (6 \times 96) + (4 \times 163) + (2 \times 143) + (1 \times 128) + (2 \times 125) + (3 \times 139) + (2 \times 68) + (1 \times 85) = (8 + 130 + 51 + 42 + 78 + 20 + 392 + 6 + 72 + 162 + 90 + 1068 + 77 + 258 + 516 + 124 + 152 + 576 + 652 + 286 + 128 + 250 + 417 + 136 + 85) = 5776 : 79 = 73.113$$

Nilai *pixel* yang diubah adalah 11 menjadi 73.

**Langkah 3:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (2,0) dari *kernel*.

65	17	21	78	0	1	2	3	2	1
98	1	18	81	24	2	4	6	4	2
178	11	43	172	125	3	6	7	6	3
38	96	163	143	42	2	4	6	4	2
125	139	68	85	165	1	2	3	2	1

$$G(x,y) = (1 \times 65) + (2 \times 17) + (3 \times 21) + (2 \times 78) + (1 \times 0) + (2 \times 98) + (4 \times 1) + (6 \times 18) + (4 \times 81) + (2 \times 24) + (3 \times 178) + (6 \times 11) + (7 \times 43) + (6 \times 172) + (3 \times 125) + (2 \times 38) + (4 \times 96) + (6 \times 163) + (4 \times 143) + (2 \times 42) + (1 \times 125) + (2 \times 139) + (3 \times 68) + (2 \times 85) + (1 \times 165) = (65 + 34 + 63 + 156 + 0 + 196 + 4 + 108 + 324 + 48 + 534 + 66 + 301 + 1032 + 375 + 76 + 356 + 978 + 572 + 84 + 125 + 278 + 204 + 170 + 165) = 6314 : 79 = 79.924$$

Nilai *pixel* yang diubah adalah 43 menjadi 80.

**Langkah 4:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (3,0) dari *kernel*.

17	21	78	0	34	1	2	3	2	1
1	18	81	24	17	2	4	6	4	2
11	43	172	125	50	3	6	7	6	3
96	163	143	42	39	2	4	6	4	2
139	68	85	165	136	1	2	3	2	1

$$G(x,y) = (1 \times 17) + (2 \times 21) + (3 \times 78) + (2 \times 0) + (1 \times 34) + (2 \times 1) + (4 \times 18) + (6 \times 81) + (4 \times 24) + (2 \times 17) + (3 \times 11) + (6 \times 43) + (7 \times 172) + (6 \times 125) + (3 \times 50) + (2 \times 96) + (4 \times 163) + (6 \times 143) + (4 \times 42) + (2 \times 39) + (1 \times 139) + (2 \times 68) + (3 \times 85) + (2 \times 165) + (1 \times 136) = (17 + 42 + 234 + 0 + 34 + 2 + 72 + 486 + 96 + 34 + 33 + 258 + 1204 + 750 + 150 + 192 + 652 + 858 + 84 + 78 + 139 + 136 + 255 + 330 + 136) = 6272 : 79 = 79.392$$

Nilai *pixel* yang diubah adalah 172 menjadi 79

**Langkah 5:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (4,0) dari *kernel*.

21	78	0	34	56	1	2	3	2	1
18	81	24	17	64	2	4	6	4	2
43	172	125	50	179	3	6	7	6	3
163	143	42	39	64	2	4	6	4	2
68	85	165	136	82	1	2	3	2	1

$$G(x,y) = (1 \times 21) + (2 \times 78) + (3 \times 0) + (2 \times 34) + (1 \times 56) + (2 \times 18) + (4 \times 81) + (6 \times 24) + (4 \times 17) + (2 \times 64) + (3 \times 43) + (6 \times 172) + (7 \times 125) + (6 \times 50) + (3 \times 179) + (2 \times 163) + (4 \times 143) + (6 \times 42) + (4 \times 39) + (2 \times 64) + (1 \times 68) + (2 \times 85) + (3 \times 165) + (2 \times 136) + (1 \times 82) = (21 + 156 + 0 + 68 + 56 + 36 + 504 + 144 + 68 + 128 + 129 + 1092 + 1575 + 150 + 358 + 326 + 286 + 168 + 170 + 128 + 136 + 170 + 170 + 82) = 7992 : 79 = 101.164$$





$$(1 \times 82) = (21 + 156 + 0 + 68 + 56 + 36 + 324 + 144 + 68 + 128 + 129 + 1032 + 875 + 300 + 537 + 326 + 572 + 252 + 156 + 128 + 68 + 170 + 495 + 272 + 82) = 6395 : 79 = 80.949$$

Nilai *pixel* yang diubah adalah 125 menjadi 81

**Langkah 6:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (5,0) dari *kernel*.

78	0	34	56	0	1	2	3	2	1
81	24	17	64	111	2	4	6	4	2
172	125	50	179	163	3	6	7	6	3
143	42	39	64	164	2	4	6	4	2
85	165	136	82	52	1	2	3	2	1

$$G(x,y) = (1 \times 78) + (2 \times 0) + (3 \times 34) + (2 \times 56) + (1 \times 0) + (2 \times 81) + (4 \times 24) + (6 \times 17) + (4 \times 64) + (2 \times 111) + (3 \times 172) + (6 \times 125) + (7 \times 50) + (6 \times 179) + (3 \times 163) + (2 \times 143) + (4 \times 42) + (6 \times 39) + (4 \times 64) + (2 \times 164) + (1 \times 85) + (2 \times 165) + (3 \times 136) + (2 \times 82) + (1 \times 52) = (78 + 0 + 102 + 112 + 0 + 162 + 96 + 102 + 256 + 222 + 516 + 750 + 350 + 1074 + 489 + 286 + 168 + 234 + 256 + 328 + 85 + 330 + 408 + 164 + 52) = 6620 : 79 = 83.797$$

Nilai *pixel* yang diubah adalah 50 menjadi 84

**Langkah 7:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (6,0) dari *kernel*.

0	34	56	0	95	1	2	3	2	1
24	17	64	111	99	2	4	6	4	2
125	50	179	163	166	3	6	7	6	3
42	39	64	164	238	2	4	6	4	2
165	136	82	52	25	1	2	3	2	1

$$G(x,y) = (1 \times 0) + (2 \times 34) + (3 \times 56) + (2 \times 0) + (1 \times 95) + (2 \times 24) + (4 \times 17) + (6 \times 64) + (4 \times 111) + (2 \times 99) + (3 \times 125) + (6 \times 50) + (7 \times 179) + (6 \times 163) + (3 \times 166) + (2 \times 42) + (4 \times 39) + (6 \times 64) + (4 \times 164) + (2 \times 238) + (1 \times 165) + (2 \times 136) + (3 \times 82) + (2 \times 52) + (1 \times 25) = (0 + 68 + 168 + 0 + 95 + 48 + 68 + 384 + 444 + 198 + 375 + 300 + 1253 + 978 + 498 + 84 + 156 + 384 + 584 + 476 + 165 + 272 + 246 + 104 + 25) = 7373 : 79 = 93.329$$

Nilai *pixel* yang diubah adalah 179 menjadi 93

**Langkah 8:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (7,0) dari *kernel*.

34	56	0	95	59	1	2	3	2	1
17	64	111	99	75	2	4	6	4	2
50	179	163	166	176	3	6	7	6	3
39	64	164	238	172	2	4	6	4	2
136	82	52	25	69	1	2	3	2	1

$$G(x,y) = (1 \times 34) + (2 \times 56) + (3 \times 0) + (2 \times 95) + (1 \times 59) + (2 \times 17) + (4 \times 64) + (6 \times 111) + (4 \times 99) + (2 \times 75) + (3 \times 50) + (6 \times 179) + (7 \times 163) + (6 \times 166) + (3 \times 176) + (2 \times 39) + (4 \times 64) + (6 \times 164) + (4 \times 238) + (2 \times 172) + (1 \times 136) + (2 \times 82) + (3 \times 52) + (2 \times 25) + (1 \times 69) = (34 + 112 + 0 + 190 + 59 + 34 + 256 + 666 + 396 + 150 + 150 + 1074 + 1141 + 996 + 528 + 78 + 256 + 984 + 952 + 344 + 136 + 164 + 156 + 50 + 169) = 9075 : 79 = 114.873$$

Nilai *pixel* yang diubah adalah 163 menjadi 115

**Langkah 9:** Geser *kernel* satu *pixel* ke kanan, kemudian hitung nilai *pixel* pada posisi (8,0) dari *kernel*.

56	0	95	59	13	1	2	3	2	1
64	111	99	75	32	2	4	6	4	2
179	163	166	176	60	3	6	7	6	3
64	164	238	172	91	2	4	6	4	2
82	52	25	69	151	1	2	3	2	1

$$G(x,y) = (1 \times 56) + (2 \times 0) + (3 \times 95) + (2 \times 59) + (1 \times 13) + (2 \times 64) + (4 \times 111) + (6 \times 99) + (4 \times 75) + (2 \times 32) + (3 \times 179) + (6 \times 163) + (7 \times 166) + (6 \times 176) + (3 \times 60) + (2 \times 64) + (4 \times 164) + (6 \times 238) + (4 \times 172) + (2 \times 91) + (1 \times 82) + (2 \times 52) + (3 \times 25) + (2 \times 69) + (1 \times 151) = (56 + 0 + 285 + 118 + 13 + 128 + 444 + 594 + 300 + 64 + 537 + 978 + 1162 + 180 + 128 + 656 + 1428 + 688 + 182 + 82 + 104 + 75 + 138 + 151) = 8491 : 79 = 107.481$$

Nilai *pixel* yang diubah adalah 166 menjadi 107

**Langkah 10** sampai dengan **Langkah 400** sama dengan seperti **Langkah 9**, dimana konvolusi diabaikan sehingga *pixel-pixel* sudut bawah, nilainya tetap sama seperti citra asal.



**Gambar 5.** Citra bola futsal dengan metode LOG

#### 4. KESIMPULAN

Berdasarkan penelitian yang berjudul Deteksi Tepi Menggunakan Metode Laplacian Of Gaussian pada Citra Bola Futsal (Studi Kasus: AFK Medan), maka kesimpulan yang diperoleh sebagai berikut:

1. Proses dalam mendeteksi tepi citra bola futsal harus menggunakan metode laplacian of Gaussian untuk mengetahui tepi pada citra bola futsal tersebut.
2. Perancangan Deteksi tepi Citra bola futsal Menggunakan Matlab dan menerapkan metode dari Laplacian Of Gaussian.
3. Metode Laplacian Of Gaussian harus diterapkan dalam program Matlab Agar program yang berjalan sesuai dengan perancangan aplikasi deteksi tepi ini.

#### REFERENCES

- Prihartini, T. S., & Andono, P. N. (n.d.). *DETEKSI TEPI DENGAN METODE LAPLACIAN OF GAUSSIAN PADA CITRA DAUN TANAMAN KOPI*. 1–5.
- Susanto, A. K. & A. (2013). *Teori dan Aplikasi Pengolahan Citra* (2012th ed.; D. Hardjono, Ed.). Yogyakarta: CV. ANDI OFFSET.
- Tomography, C., & Tomography, C. (2012). *Pendeteksian Tepi Citra CT Scan dengan Menggunakan Laplacian of Gaussian ( LOG )*. *II*(1), 17–22.
- Darma, P. (2010). *Pengolahan Citra Digital*. Yogyakarta: Andi.
- Prasetyo, E. (2011). *Pengolahan Citra Digital dan Aplikasinya menggunakan Matlab*. Yogyakarta: Andi.
- Prihartini, T. S., & Andono, P. N. (n.d.). *DETEKSI TEPI DENGAN METODE LAPLACIAN OF GAUSSIAN PADA CITRA DAUN TANAMAN KOPI*. 1–5.
- Susanto, A. K. & A. (2013). *Teori dan Aplikasi Pengolahan Citra* (2012th ed.; D. Hardjono, Ed.). Yogyakarta: CV. ANDI OFFSET.
- T Sutoyo. (2009). *Teori Pengolahan Citra Digital*. Yogyakarta: ANDI.