

***FISH MOVEMENT TRACKING MENGGUNAKAN
METODE GAUSSIAN MIXTURE MODELS (GMM) DAN
KALMAN FILTER***

Skripsi

**Disusun untuk memenuhi salah satu syarat
memperoleh gelar Sarjana Komputer**



**Hafizhun Alim
1313617032**

**PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA
2023**

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

Skripsi yang berjudul

Fish Movement Tracking Menggunakan Metode Gaussian Mixture Models (GMM) dan Kalman Filter

disusun oleh

Nama :Hafizhun Alim

NIM :1313617032

telah dipertahankan di hadapan sidang Panitia Ujian Skripsi FMIPA UNNES pada 9 Februari 2023

Panitia:

Ketua

Sekretaris

Dr Sugianto M.Si

NIP. 1961 0219 1993 03 1 001

Endang Sugiharti S.Si.,M.Kom

NIP. 1974 0107 1999 03 2 001

Penguji 1

Riza Arifudin S.Pd., M.Cs.

NIP. 1980 0525 2005 01 1 001

Penguji 2

Pembimbing

Endang Sugiharti S.Si.,M.Kom

NIP. 1974 0107 1999 03 2 001

Drs. Mulyono, M.Kom.

NIP. 196605171994031003

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul "***Fish Movement Tracking Menggunakan Metode Gaussian Mixture Models (GMM) dan Kalman Filter***" yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Sumber informasi yang diperoleh dari penulis lain yang telah dipublikasikan dan disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

Jika dikemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian-bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang berlaku.

Jakarta, 31 Januari 2023

Hafizhun Alim
1313617032

HALAMAN PERSEMBAHAN

*Untuk Umi, Abi,
dan Adik-adikku tercinta.*

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur penulis panjatkan ke hadirat Allah SWT karena hanya dengan rahmat dan hidayah-Nya, Tugas Akhir ini dapat terselesaikan. Keberhasilan dalam menyusun laporan Tugas Akhir ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan masukan guna sempurnanya Tugas Akhir ini. Oleh karena itu dalam kesempatan ini, dengan kerendahan hati penulis mengucapkan terima kasih kepada:

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Ir. Fariani Hermin Indiyah, M.T selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Drs. Mulyono, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
4. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
5. Yth. Seluruh Dosen Pengajar dan Staff di Prodi Ilmu Komputer Universitas Negeri Jakarta.
6. Umi dan Abi yang selama ini telah sabar membimbing, mengarahkan, dan mendoakan penulis tanpa kenal lelah untuk selama-lamanya.
7. Pradyanti, Bagus Nugraha, dan Farid W.M. sebagai rekan belajar yang telah membantu dan mendukung penulis dalam menyusun tugas akhir ini.
8. Teman-teman Program Studi Ilmu Komputer 2017 yang telah membantu dan mendukung, tugas akhir ini dapat diselesaikan.
9. Dan semua pihak yang juga telah membantu dengan tidak mengurangi rasa hormat penulis yang tidak dapat disebutkan satu persatu

Penulis menyadari bahwa penyusunan tugas akhir ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karenanya, kritik dan saran yang bersifat membangun akan penulis terima dengan senang hati. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri. Semoga Allah SWT senantiasa membela kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan tugas akhir ini.

Wassalamu'alaikum Wr. Wb.

Jakarta, 31 Januari 2023

Hafizhun Alim
1313617032

ABSTRAK

HAFIZHUN ALIM. *Fish Movement Tracking Menggunakan Metode Gaussian Mixture Models dan Kalman Filter.* Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta. 2023. Dibawah bimbingan Drs. Mulyono, M.Kom dan Muhammad Eka Suryana, M.Kom.

Potensi industri perikanan Indonesia merupakan yang terbesar di dunia, baik perikanan tangkap maupun perikanan budidaya. Salah satu masalah yang sering dihadapi oleh pelaku usaha budidaya ikan adalah pada saat proses penghitungan ikan yang masih menggunakan cara-cara manual. Penelitian ini bertujuan untuk melakukan proses penghitungan ikan dengan lebih mudah dan efisien. Dalam penelitian ini dilakukan proses *tracking* objek ikan menggunakan *Kalman Filter* yang dibantu oleh GMM sebagai metode deteksinya. Keluaran yang didapat menunjukan bahwa sistem mampu melakukan pelacakan dengan *error* yang kecil, serta menghasilkan rata-rata jumlah objek yang mendekati rata-rata jumlah objek sebenarnya pada video kategori latar belakang sederhana.

Kata kunci : *Object Detection Fish Movement Tracking, Ikan, Object Tracking, Peternakan Ikan.*

ABSTRACT

HAFIZHUN ALIM. *Fish Movement Tracking Using Gaussian Mixture Models and Kalman Filter.* Thesis. Computer Science Study Program, Faculty of Mathematics and Natural Sciences, State University of Jakarta. January 2023. Under the supervision of Drs. Mulyono, M.Kom dan Muhammad Eka Suryana, M.Kom.

The potential of Indonesia's fisheries industry is the largest in the world, both capture fisheries and aquaculture. One of the problems often faced by fish farmers is during the fish counting process which still uses manual methods. This research aims to make the fish counting process easier and more efficient. In this research, the process of tracking fish objects using Kalman Filter assisted by GMM as a detection method is carried out. The output obtained shows that the system is able to track with a small error, as well as produce an average number of objects that is close to the average number of actual objects in the simple background video category.

Keywords : *Object Detection Fish Movement Tracking, Fish, Object Tracking, Fishery.*

DAFTAR ISI

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI	ii
LEMBAR PERNYATAAN	iii
HALAMAN PERSEMBAHAN	iv
KATA PENGANTAR	v
ABSTRAK	vii
ABSTRACT	viii
DAFTAR ISI	xi
DAFTAR TABEL	xiii
DAFTAR GAMBAR	xv
DAFTAR LAMPIRAN	xv
I PENDAHULUAN	1
1.1 Latar Belakang Masalah	1
1.2 Rumusan Masalah	5
1.3 Batasan Masalah	5
1.4 Tujuan Penelitian	6
1.5 Manfaat Penelitian	6
II KAJIAN PUSTAKA	7
2.1 Pengertian Citra Digital	7
2.2 Pengolahan Citra Digital	8
2.3 Pengertian Citra Bergerak (Video)	9
2.4 Deteksi Objek Bergerak	10
2.5 <i>Background Subtraction</i>	11
2.6 <i>Background Subtraction</i> Menggunakan <i>Gaussian Mixture Models</i>	12
2.6.1 <i>Gaussian Mixture Models</i>	13
2.6.2 <i>Modeling Process</i>	14

2.6.3 <i>Background Model Estimation</i>	15
2.7 Operasi Morfologi	16
2.8 Jenis - Jenis Operasi Morfologi	17
2.8.1 Erosi	18
2.8.2 Dilasi	18
2.8.3 <i>Opening</i>	19
2.8.4 <i>Closing</i>	20
2.9 <i>Downsampling</i>	21
2.10 Contour Tracing	22
2.11 Pelacakan Objek	29
2.12 Kalman Filter	29
2.12.1 Definisi	29
2.12.2 Algoritme Kalman Filter	31
2.13 <i>Decision of Occlusion</i>	33
2.14 Asosiasi Data	34
III METODE PENELITIAN	36
3.1 Deskripsi Sistem	36
3.2 Perancangan Sistem	36
3.2.1 Proses Input Video	38
3.2.2 Keluaran	38
3.2.3 Deteksi Objek dengan GMM	38
3.2.4 Proses Menghilangkan Noise Melalui Operasi Morfologi	40
3.2.5 <i>Downsampling</i> dan <i>Contour Tracing</i>	41
3.2.6 <i>Decision of Occlusion</i> , Asosiasi Data, dan Kalman Filter	43
3.3 Perancangan Eksperimen	46
3.3.1 Sumber Data	46
3.3.2 Evaluasi Hasil	49
3.3.3 Parameter Sistem	53
IV HASIL DAN PEMBAHASAN	54
4.1 Input Video	54
4.2 GMM	54
4.3 Operasi Morfologi	61
4.4 <i>Contour Tracing</i> dan <i>Downsampling</i>	66
4.5 Kalman Filter	81

V PENUTUP	93
5.1 Kesimpulan	93
5.2 Saran	94
DAFTAR PUSTAKA	97
LAMPIRAN	98
DAFTAR RIWAYAT HIDUP	99

DAFTAR TABEL

Tabel 2.1	Aturan penetapan <i>parent border</i>	25
Tabel 4.1	Hasil ujicoba proses <i>background subtraction</i> menggunakan GMM terhadap video indeks 9908	55
Tabel 4.2	Hasil ujicoba proses <i>background subtraction</i> menggunakan GMM terhadap video indeks 9986	56
Tabel 4.3	Hasil uji coba proses <i>background subtraction</i> menggunakan GMM terhadap video indeks gt_124	57
Tabel 4.4	Hasil uji coba proses <i>background subtraction</i> menggunakan GMM terhadap video indeks gt_116	58
Tabel 4.5	Skor <i>Accuracy</i> , <i>Recall</i> , <i>Precision</i> , dan <i>F1</i> pada metode <i>Background Subtraction</i> menggunakan GMM	59
Tabel 4.6	Hasil uji coba proses <i>background subtraction</i> menggunakan GMM yang disempurnakan oleh Operasi Morfologi	61
Tabel 4.7	Skor <i>Accuracy</i> , <i>Recall</i> , <i>Precision</i> , dan <i>F1</i> Operasi Morfologi .	63
Tabel 4.8	Hasil uji coba metode CT yang ditingkatkan oleh <i>Downsampling</i> pada video indeks 9908 dengan ukuran <i>kernel</i> Operasi Morfologi 7x13	66
Tabel 4.9	Hasil uji coba metode CT yang ditingkatkan oleh <i>Downsampling</i> pada video indeks 9866 dengan ukuran <i>Kernel</i> Operasi Morfologi 7x13	68
Tabel 4.10	Hasil uji coba metode CT yang ditingkatkan oleh <i>Downsampling</i> pada video indeks gt_124 dengan ukuran <i>kernel</i> Operasi Morfologi 7x13	70
Tabel 4.11	Hasil uji coba performa metode CT yang ditingkatkan oleh <i>Downsampling</i> pada video indeks gt_116 dengan ukuran <i>kernel</i> Operasi Morfologi 7x13	72
Tabel 4.12	Evaluasi hasil dan performa metode CT video kategori objek tunggal terhadap ukuran <i>kernel</i> Operasi Morofologi serta skala <i>Downsampling</i> yang berbeda-beda	74
Tabel 4.13	Evaluasi hasil dan performa metode CT video kategori objek lebih dari satu terhadap ukuran <i>kernel</i> Operasi Morofologi serta skala <i>Downsampling</i> yang berbeda-beda	77

Tabel 4.14	Evaluasi hasil dan performa <i>counting</i> metode KF terhadap ukuran <i>kernel</i> Operasi Morofologi serta skala <i>Downsampling</i> yang berbeda-beda	85
Tabel 4.15	Evaluasi hasil dan performa <i>tracking</i> metode KF terhadap ukuran <i>kernel</i> Operasi Morofologi serta skala <i>Downsampling</i> yang berbeda-beda	88

DAFTAR GAMBAR

Gambar 2.1	Pendekatan Tradisional Deteksi Objek Bergerak	10
Gambar 2.2	Skema sederhana metode <i>Background Subtraction</i>	12
Gambar 2.3	Representasi <i>Structuring Element</i>	17
Gambar 2.4	Operasi Erosi	18
Gambar 2.5	Operasi Dilasi	19
Gambar 2.6	Operasi <i>Opening</i>	20
Gambar 2.7	Operasi <i>Closing</i>	20
Gambar 2.8	<i>Image Downsampling dan Upsampling</i>	21
Gambar 2.9	Piramid Citra	21
Gambar 2.10	Hubungan antar komponen dan tepi	23
Gambar 2.11	Kondisi <i>starting point</i> dari algoritme <i>border following</i> untuk tepi luar (a) dan tepi lubang (b)	24
Gambar 2.12	Struktur topologi antara tepi satu dengan tepi yang lain jika menggunakan Algoritme 1. Hasil citra (a) dan Struktur yang diekstrak (b)	27
Gambar 2.13	Hasil dari algoritme 2. Tanda "#" merepresentasikan <i>starting point</i> dan ":" merepresentasikan nilai -2	28
Gambar 2.14	Ilustrasi dari proses <i>merge</i> dan <i>split</i>	33
Gambar 3.1	Diagram alir keseluruhan proses sistem	37
Gambar 3.2	Diagram alir proses <i>Background Subtraction</i> dengan GMM .	39
Gambar 3.3	Diagram alir Operasi Morfologi	40
Gambar 3.4	Diagram alir proses <i>Contour Tracing</i>	42
Gambar 3.5	Diagram alir proses pelacakan dengan Kalman Filter	45
Gambar 3.6	Contoh keluaran sistem	46
Gambar 3.7	Sampel <i>frame</i> video dataset <i>DeepFish</i> indeks 9908 . . .	47
Gambar 3.8	Sampel <i>frame</i> video dataset <i>DeepFish</i> indeks 9866 . . .	48
Gambar 3.9	Sampel <i>frame</i> video dataset <i>PeRCeiVeLab</i> indeks gt_124 . .	48
Gambar 3.10	Sampel <i>frame</i> video dataset <i>PeRCeiVeLab</i> indeks gt_116 . .	49
Gambar 3.11	Contoh <i>groundtruth</i> GMM	50
Gambar 3.12	Contoh <i>groundtruth</i> <i>Contour Tracing</i>	52
Gambar 4.1	Potongan sampel kode input sistem	54

Gambar 4.2 Sampel <i>groundtruth</i> data uji video indeks (a) 9908 (b) 9866 (c) gt_124 (d) gt_116	55
Gambar 4.3 Potongan sampel kode Background Subtraction yang dibantu oleh GMM	58
Gambar 4.4 Potongan sampel kode <i>Background Subtraction</i> dan Operasi Morfologi	65
Gambar 4.5 Potongan sampel kode <i>Contour Tracing</i>	81
Gambar 4.6 Sample <i>frame</i> hasil tracking <i>Kalman Filter</i> video indeks 9908	82
Gambar 4.7 Sample <i>frame</i> hasil tracking <i>Kalman Filter</i> video indeks 9866	82
Gambar 4.8 Sample <i>frame</i> hasil tracking <i>Kalman Filter</i> video indeks gt_124	83
Gambar 4.9 Sample <i>frame</i> hasil tracking <i>Kalman Filter</i> video indeks gt_116	84
Gambar 4.10 Potongan sampel kode <i>Kalman Filter</i>	91
Gambar 4.11 Potongan sampel kode <i>Kalman Filter</i>	92

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Potensi industri perikanan Indonesia merupakan yang terbesar di dunia, baik perikanan tangkap maupun perikanan budidaya. Berdasarkan taktik atau metode produksi, perikanan terbagi menjadi dua yaitu perikanan tangkap (*capture fisheries*) dan perikanan budidaya (*aquaculture*), dengan potensi produksi lestari sekitar 67 juta ton per tahun. Berdasarkan angka tersebut, maka potensi produksi tangkapan lepas pantai adalah 9,3 juta ton per tahun, dengan potensi tangkapan darat (danau, sungai, waduk, rawa) sebesar 0,9 ton per tahun. Sisanya 56,8 juta ton per tahun merupakan potensi perikanan budidaya, baik budidaya laut, budidaya air payau, maupun budidaya air tawar. Budidaya ikan adalah salah satu jenis akuakultur yang melibatkan ikan, yang biasanya dilakukan dalam sebuah tangki tertutup ataupun kolam ikan buatan. Budidaya ikan memiliki berbagai jenis manfaat antara lain adalah, untuk memproduksi atau menghasilkan bahan pangan, sebagai sarana rekreasi dalam bentuk tempat pemancingan ikan, dan juga ikan hias. Di Indonesia sendiri, Badan Pusat Statistik (BPS) mencatat ada lebih dari 1,5 juta usaha budidaya ikan dengan total produksi mencapai 16 juta ton di tahun 2016. Dengan proporsi konsumsi protein yang berasal dari ikan/udang/cumi/kerang mencapai 13,27 persen pada tahun 2019, masih di atas konsumsi daging sebesar 6,54 persen serta konsumsi telur/susu sebesar 5,50 persen.

Seiring dengan semakin tingginya angka produksi, dibutuhkan segera integrasi penuh Teknologi Informasi terhadap perikanan budidaya, guna meningkatkan efisiensi dan efektivitas dari level produksi hingga penjualan. Salah satu masalah yang sering dihadapi oleh pelaku usaha budidaya ikan adalah pada saat proses penghitungan ikan. Menghitung jumlah ikan diperlukan ketika peternak ingin

menjual ikan yang telah mereka budidaya. Pada praktiknya (Al-Amri, 2020), banyak peternak ikan yang masih menggunakan cara manual dalam proses penghitungan ikan, yaitu dengan cara memindahkan ikan satu persatu dari satu wadah ke wadah lainnya. Cara lainnya adalah dengan memasukkan ikan ke dalam suatu wadah yang kemudian ditimbang beratnya. Kedua cara tersebut tentu sangat tidak efisien dan efektif. Penghitungan ikan secara manual dapat merusak fisik ikan dan sangat tidak hemat waktu, sementara penghitungan ikan dengan menggunakan berat tidaklah selalu akurat.

Dari permasalahan di atas, diperlukan sebuah alat yang dapat menghitung jumlah ikan secara otomatis. Affandy (2016) melakukan penelitian terhadap proses penghitungan ikan, yang membuat sebuah *prototype* alat penghitung ikan otomatis dengan menggunakan sensor *infra red*, yaitu dengan cara mengalirkan ikan secara satu persatu melalui sensor tersebut, yang keluarannya kemudian ditampilkan melalui layar LED. Dari penelitian tersebut, didapatkan hasil akhir yang dapat diterima, dengan tingkat kepercayaan sebesar 95%. Efisiensi waktu juga menjadi lebih baik, untuk setiap 100 ekor bibit ikan mas, dapat dihitung dalam waktu 100 detik yang semula 190 detik. Al-Amri (2020) juga melakukan penelitian yang sama. Perbedaannya hanya di sensor yang digunakan, yaitu sensor *proximity*. Dari hasil uji coba didapatkan hasil yang baik, dengan persentase error sebesar 4,07%. Pengukuran waktu juga tercatat mengalami perubahan yang cukup signifikan dengan perbandingan waktu, 20 menit per 1000 bibit ikan dengan cara manual, dan 228 detik per 1000 bibit ikan dengan menggunakan alat tersebut.

Penghitungan ikan menggunakan alat fisik seperti di atas bukan tanpa kekurangan. Walaupun hasil yang dicapai baik, terdapat beberapa poin yang perlu diperhatikan. Hal yang paling utama adalah penelitian tersebut dilakukan dalam kondisi *closed and controlled environment*. Misalnya pada penelitian Al-Amri (2020), modifikasi terhadap lubang keluaran ikan perlu dilakukan jika ingin menghitung ikan dengan ukuran yang berbeda. Dan juga soal sensor *proximity* yang

kurang responsif dalam membaca pergerakan ikan jika ikan bergerak terlalu cepat serta saling berdempetan. Lalu ada sensor *infra red* pada penelitian Affandy (2016) yang sangat sensitif terhadap perubahan cahaya dan juga getaran dari luar yang dapat membuat *reading* pada sensor menjadi tidak akurat. Belum lagi biaya yang harus dikeluarkan untuk membeli alat dan juga biaya perawatan jika terdapat sensor yang rusak. Maka dari itu, dibutuhkan sebuah sistem yang lebih fleksibel, mudah digunakan, *robust* terhadap perubahan lingkungan dan *occlusion*, serta membutuhkan biaya yang sangat minim ataupun tidak sama sekali (*zero-cost*).

Pelacakan objek (*object tracking*) adalah salah satu bidang penting dalam visi komputer. Salah satu objektifnya adalah untuk menentukan jumlah objek. Dijelaskan dalam (Challa et al., 2011), pelacakan objek mengacu kepada penggunaan sensor (radar, kamera, dan lain-lain) untuk menentukan lokasi, lintasan, atau bahkan karakteristik sebuah objek. Survey yang dilakukan oleh (Yilmaz et al., 2006) mendefinisikan pelacakan objek sebagai tindakan melakukan estimasi atau prediksi terhadap lintasan sebuah objek bergerak pada bidang gambar dalam sebuah adegan video (kamera sebagai sensor). Dengan kata lain, sebuah pelacak (*tracker*) dapat secara konsisten melacak informasi (lokasi) sebuah objek di dalam *frame* selama video berlangsung. Yilmaz juga mengemukakan bahwa terdapat tiga *tasks* penting dalam sistem pelacakan objek: deteksi objek bergerak, pelacakan objek, dan representasi objek. Salah satu tantangan dalam membangun sebuah sistem pelacakan objek adalah menentukan metode deteksi objek yang efisien. Metode yang sering digunakan untuk mengekstrak objek bergerak dari sebuah video adalah *Background Subtraction*. Metode ini melibatkan operasi pengurangan antara *frame* berisi objek yang ingin diekstrak, dengan *background model* (Saravanakumar et al., 2010). *Background model* berisi segala sesuatu yang dapat dianggap sebagai *background*. Selisih nilai piksel antara keduanya kemudian diekstrak sebagai objek dengan menggunakan teknik *thresholding*. Walaupun teknik *Background Subtraction* sangat populer karena kemudahan implementasinya, teknik ini menjadi sangat buruk performanya ketika *background* yang dimodelkan tidak statik, seperti adanya

perubahan iluminasi, pergerakan yang berulang dari suatu objek (dahan pohon yang tertutup angin), atau perubahan bentuk geometri pada *background*.

Penggunaan *Gaussian Mixture Model* (GMM) untuk memodelkan *background* pertama kali diperkenalkan oleh Friedman dan Russell (1997) untuk mendeteksi kendaraan. Metode tersebut mengklasifikasikan seluruh nilai piksel ke dalam tiga distribusi *Gaussian* yang telah ditentukan sebelumnya. Distribusi pertama sebagai warna jalan, distribusi kedua sebagai warna bayangan, dan distribusi ketiga sebagai warna kendaraan. Hasil deteksi objek dari percobaan yang mereka lakukan tampak lebih jernih dan lebih baik dari hasil *Background Subtraction*, akan tetapi metode yang mereka ajukan tidak menjelaskan bagaimana perilaku piksel yang tidak masuk ke dalam tiga distribusi tersebut, karena sebuah piksel bisa saja merepresentasikan lebih dari satu warna selama video berlangsung. Stauffer dan Grimson (1999) menyajikan metode yang lebih sempurna. Tidak seperti metode Friedman yang memodelkan seluruh nilai piksel ke dalam tiga distribusi yang kelasnya sudah ditentukan, Stauffer memodelkan nilai sebuah piksel sebagai *mixture of Gaussians*. Berdasarkan *evidence* dan varians dari masing-masing distribusi Gaussian, dapat ditentukan distribusi mana yang masuk ke dalam kelas *background* dan distribusi mana yang masuk ke dalam kelas *foreground*.

Objektif dari sebuah metode pelacakan adalah untuk menghasilkan lintasan dari sebuah objek dengan cara menentukan lokasi objek pada setiap *frame* dalam video. Pada penelitian yang dilakukan oleh (Jeong et al., 2014), setelah proses deteksi objek, Kalman Filter diinisialisasi sebanyak jumlah objek yang terdeteksi (pemberian identitas untuk masing-masing objek). Kalman Filter pertama kali diperkenalkan oleh R. E. Kalman pada tahun 1960 (Kalman, 1960). Secara garis besar, Kalman Filter digunakan untuk memprediksi lokasi objek pada *frame* selanjutnya. Hasil prediksi tersebut kemudian diasosiasikan kembali dengan deteksi objek pada *frame* berikutnya, dengan begitu identitas objek dapat dipertahankan di setiap *frame* selama video berlangsung.

Dari permasalahan di atas, penulis mengusulkan untuk melakukan pelacakan pergerakan objek ikan (*fish movement tracking*) menggunakan metode Kalman Filter dengan GMM sebagai metode deteksi objeknya. GMM berguna untuk memodelkan latar belakang yang dinamis, adaptif terhadap perubahan cahaya, pergerakan repetitif (dahan pohon yang tertutup angin), objek yang bergerak lambat, serta dapat melakukan deteksi objek pada *scene* yang berantakan / ramai. Penelitian ini berfokus pada pelacakan lebih dari satu objek ikan menggunakan metode GMM dan Kalman Filter. Hasil yang diharapkan adalah sistem mampu melakukan pelacakan lebih dari satu objek sekaligus secara akurat.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan, maka perumusan masalah pada penelitian ini adalah "Bagaimana merancang dan membangun sebuah sistem pelacakan pergerakan pada objek ikan menggunakan metode GMM dan Kalman Filter?"

1.3 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. *Fish movement tracking* menggunakan metode Kalman Filter dengan GMM sebagai metode deteksi objek bergeraknya.
2. Antarmuka sistem berbasis *command-line*.
3. Masukan sistem berupa video berisikan ikan.
4. Objek selain ikan masih dapat dideteksi.
5. Jenis video yang digunakan adalah video dengan format mp4, flv, avi.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk merancang dan membangun sebuah sistem yang dapat melakukan pelacakan pergerakan ikan menggunakan metode GMM dan Kalman Filter, serta menghasilkan keluaran rata-rata objek selama video berlangsung.

1.5 Manfaat Penelitian

Beberapa manfaat yang ingin diperoleh dari penelitian ini adalah sebagai berikut:

1. Bagi Peneliti

Menambah pengetahuan penulis tentang penggunaan metode GMM dan Kalman Filter untuk melakukan *tracking* pergerakan ikan.

2. Bagi Peneliti Selanjutnya

Diharapkan metode yang diusulkan pada penelitian ini dapat membantu penelitian selanjutnya dalam mengembangkan sistem yang lebih kompleks dan bermanfaat, yaitu sistem penghitungan ikan.

3. Bagi Program Studi Ilmu Komputer

Penelitian "*Fish Movement Tracking*" menggunakan metode GMM dan Kalman Filter" ini dapat dijadikan sebagai referensi serta menambah wawasan masyarakat Program Studi Ilmu Komputer Universitas Negeri Jakarta.

BAB II

KAJIAN PUSTAKA

2.1 Pengertian Citra Digital

Citra digital adalah citra analog yang diubah ke dalam bentuk digital melewati proses, akuisisi, pengambilan sampel (*sampling*), dan kuantisasi. Proses akuisisi yaitu proses pemetaan suatu fenomena fisik menjadi citra kontinu dengan menggunakan sensor. Keluaran dari proses akuisisi adalah sebuah gelombang tegangan bersifat kontinu yang amplitudo dan perilaku spasialnya berkaitan dengan fenomena fisik yang ditangkap oleh sensor tersebut. Untuk mendigitalisasi citra tersebut, perlu didefinisikan sebuah fungsi yang mengambil koordinat dan amplitudo sebagai parameternya. Proses *sampling* dan kuantisasi diperlukan untuk mengubah data kontinu tersebut ke dalam bentuk digital. Proses *sampling* adalah proses mendigitalisasi nilai koordinat, sementara kuantisasi adalah proses mendigitalisasi nilai amplitudonya (Gonzalez dan Woods, 2018).

Citra digital tersusun dari banyak *picture element* atau biasa disebut piksel, yang masing-masing bernilai diskrit. Nilai diskrit tersebut merepresentasikan intensitas atau tingkat keabuan (*grayscale*) dari masing-masing piksel. Citra digital dapat didefinisikan sebagai fungsi dua dimensi $f(x, y)$, dimana x dan y melambangkan koordinat spasial dan amplitudo f sebagai nilai intensitas atau tingkat keabuan citra pada koordinat tersebut (Gonzalez dan Woods, 2018).

$$f(x, y) = \begin{bmatrix} f(0, 0) & f(0, 1) & \cdots & f(0, N - 1) \\ f(1, 0) & f(1, 1) & \cdots & f(1, N - 1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M - 1, 0) & f(M - 1, 1) & \cdots & f(M - 1, N - 1) \end{bmatrix} \quad (2.1)$$

Persamaan 2.2 merepresentasikan citra digital $f(x, y)$ dalam bentuk *array* numerik berukuran $M \times N$, dimana M adalah jumlah baris dan N adalah jumlah kolom. Variabel x dan y pada (x, y) adalah koordinat diskrit yang ditulis menggunakan bilangan integer: $x = 0, 1, 2, \dots, M - 1$ dan $y = 0, 1, 2, \dots, N - 1$. Setiap elemen dalam *array* di atas dapat disebut sebagai *image element*, *picture element*, *pel*, atau *pixel*. Citra juga dapat direpresentasikan dalam bentuk matriks tradisional. Bentuk representasi ini adalah yang digunakan untuk memproses citra pada komputer (Gonzalez dan Woods, 2018).

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,N-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,N-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{M-1,0} & a_{M-1,1} & \cdots & a_{M-1,N-1} \end{bmatrix} \quad (2.2)$$

2.2 Pengolahan Citra Digital

Pengolahan citra digital mengacu pada proses manipulasi citra digital menggunakan komputer. Proses ini mengambil citra digital sebagai masukan (*input*) dan menghasilkan keluaran (*output*) berupa citra yang sudah diolah atau dimanipulasi. Dalam bukunya, Gonzalez dan Woods (2018) membagi pengolahan citra ke dalam tiga tingkatan:

1. *Low-level Process*

Low-level Process melibatkan beberapa operasi tradisional (*image enhancement* dan *image restoration*) seperti pengurangan derau *noise reduction*, peningkatan kontras dan cahaya, penajaman citra (*image sharpening*), dan lain-lain. Karakteristik dari proses ini adalah masukan dan keluarannya berupa sebuah citra.

2. *Mid-level Process*

Proses ini meliputi beberapa teknik seperti segmentasi citra (*image segmentation*) untuk membagi citra ke dalam beberapa bagian atau objek, lalu ada pendeskripsi citra (*image descriptor*) yang dapat mendeskripsikan sebuah bagian atau objek pada citra dari segi bentuk, warna, tekstur, atau gerakan, dan selanjutnya adalah klasifikasi (*object classification / recognition*) terhadap masing-masing objek pada citra. Karakteristik dari *Mid-level Processing* adalah, walaupun masukannya merupakan sebuah citra, akan tetapi keluarannya berupa atribut-atribut yang diekstrak dari citra tersebut (contoh: garis tepi, kontur, dan identitas dari objek).

3. *High-level Process*

Terakhir, *High-level Process* adalah proses "memahami" sebuah citra (*image analysis*). Dari yang paling sederhana seperti membaca *barcode* sebuah produk, sampai ke teknik yang lebih rumit dan canggih seperti mengidentifikasi seseorang dari wajahnya.

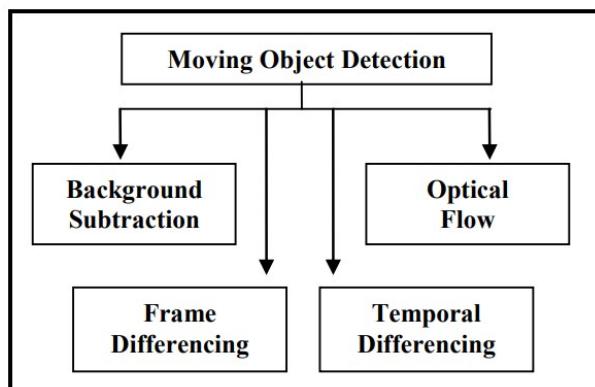
2.3 Pengertian Citra Bergerak (Video)

Munir (2004) menjelaskan dalam bukunya bahwa citra dapat dibagi menjadi dua, yaitu citra diam (*still images*) dan citra bergerak (*moving images*). Seperti namanya, citra diam adalah citra tunggal yang tidak bergerak. Sementara itu, citra bergerak adalah sekumpulan atau serangkaian citra diam yang ditampilkan secara beruntun sehingga memberikan kesan bergerak. Contoh dari pengaplikasian citra bergerak dapat dilihat pada televisi atau video. Dalam sebuah video, sekumpulan citra diam tersebut biasa didefinisikan sebagai *frame*.

2.4 Deteksi Objek Bergerak

Deteksi objek bergerak adalah sebuah tindakan segmentasi objek non-stasioner dengan latarnya dari seluruh *frame* dalam video. Dengan kata lain deteksi objek bergerak adalah sebuah teknik untuk mendeteksi pergerakan fisik dari sebuah objek di suatu wilayah atau area tertentu. Penentuan objek bergerak ini dapat dijadikan sebagai langkah dasar untuk proses klasifikasi dan pelacakan objek bergerak. Objektif utama dari deteksi objek bergerak adalah untuk menghasilkan latar depan (*foreground*) dari objek bergerak tersebut dalam setiap *frame* video (Kulchandani dan Dangarwala, 2015).

Deteksi objek bergerak telah menjadi topik diskusi utama di bidang visi komputer karena jangkauan pengaplikasiannya yang sangat luas, seperti *video surveillance*, pemantauan keamanan di bandara, navigasi robot, dan analisis lalu lintas. Deteksi objek bergerak merupakan tugas yang sangat menantang dikarenakan beberapa faktor, seperti latar belakang yang dinamis, variasi perubahan iluminasi, bayangan, dan *object occlusion*. Pendekatan tradisional untuk deteksi objek bergerak dapat dikategorikan ke dalam empat teknik, yaitu *Background Subtraction*, *Frame Differencing*, *Temporal Differencing*, dan *Optical Flow* (Kulchandani dan Dangarwala, 2015).



Gambar 2.1: Pendekatan Tradisional Deteksi Objek Bergerak
Sumber: Kulchandani dan Dangarwala (2015)

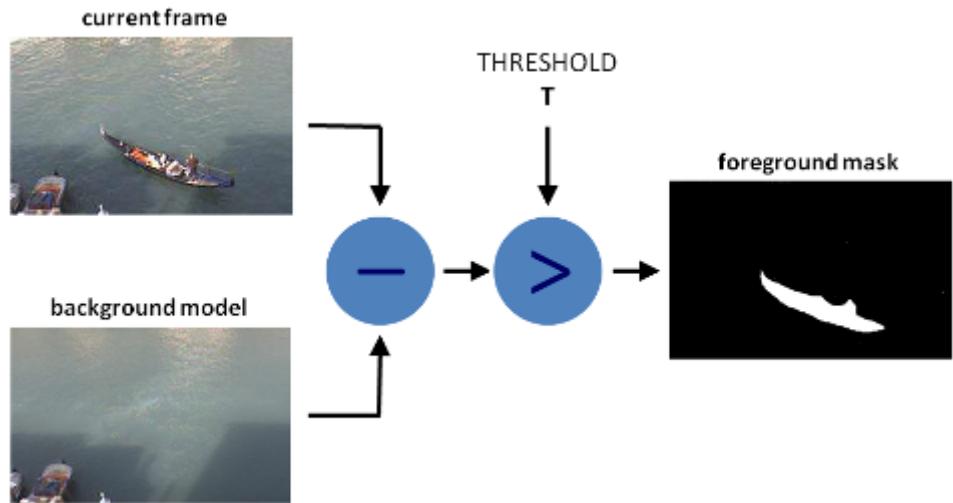
2.5 *Background Subtraction*

Background Subtraction adalah metode yang digunakan untuk mendeteksi objek bergerak dalam video dengan cara membandingkan setiap *frame* dengan sebuah (*reference frame*) atau *background model*, yang kemudian dihitung perbedaan nilai piksel diantara keduanya. Perbedaan inilah yang nantinya akan dikomparasi dengan sebuah *threshold* yang nilainya sudah ditentukan. Jika perbedaan nilai piksel melebihi nilai *threshold*, maka piksel tersebut dapat dikatakan bagian dari objek bergerak (*foreground*) dan jika tidak, piksel akan dikatakan sebagai *background* (Desai dan Gandhi, 2016). Dalam banyak literasi, metode BS dapat dirumuskan sebagai berikut:

$$X_t(s) = \begin{cases} 1 & \text{if } d(I_{s,t}, B_s) > T, \\ 0 & \text{otherwise} \end{cases} \quad (2.3)$$

Dimana T adalah nilai *threshold* yang sudah ditentukan, $X_t(s)$ adalah *foreground mask* pada waktu t , d adalah selisih antara $I_{s,t}$ yang merupakan nilai piksel s pada waktu t , dengan B_s yang merupakan *background model* pada piksel s . Yang membedakan metode BS satu dengan yang lain biasanya terletak pada bagaimana metode tersebut memodelkan B (Benezeth et al., 2010).

Background modeling adalah tahap yang penting dari *Background Subtraction*. Objektif utama dari proses *background modeling* adalah untuk mendapatkan model latar belakang yang berisikan informasi berupa bagian statis dari sebuah *scene*, atau segala sesuatu yang dapat dikatakan sebagai latar belakang. Cara paling sederhana dalam memodelkan latar belakang B adalah dengan sebuah citra yang berisikan *scene* statis tanpa objek bergerak. Citra tersebut dapat diambil dari sebuah video ketika tidak ada objek yang bergerak di dalamnya.



Gambar 2.2: Skema sederhana metode *Background Subtraction*

Sumber: OpenCV - Background Subtraction

Dibalik kemudahan implementasinya, kelemahan dari metode tradisional ini adalah sistem tidak dapat mengatasi perubahan yang terjadi pada latar belakang selama video berlangsung. Karena dalam praktiknya, latar belakang sebuah video tidak akan selalu statis. Perubahan iluminasi, pergerakan berulang suatu objek (dahan pohon yang tertutup angin), atau perubahan bentuk geometri objek latar belakang tentu dapat memengaruhi akurasi metode ini.

2.6 *Background Subtraction Menggunakan Gaussian Mixture Models*

Gaussian Mixture Models (GMM) dapat digunakan untuk memodelkan latar belakang pada teknik *Background Subtraction*. Stauffer dan Grimson (1999), dalam penelitiannya memodelkan sebuah piksel sebagai *mixture of Gaussians*. Berdasarkan *evidence* dan varians dari masing-masing distribusi Gaussian, dapat ditentukan distribusi mana yang masuk ke dalam kelas warna latar belakang (*background color*). Nilai piksel yang tidak masuk ke dalam distribusi warna latar belakang akan dianggap sebagai latar depan (*foreground*) sampai ada distribusi Gaussian yang nilai piksel tersebut termasuk di dalamnya.

Dikatakan oleh Stauffer bahwa sistem ini *robust* terhadap latar belakang yang dinamis, seperti perubahan iluminasi, pergerakan berulang sebuah benda (dahan pohon yang tertutup angin), objek yang bergerak lambat, dan objek yang masuk serta keluar dari *scene*. Objek yang bergerak lambat membutuhkan waktu yang lama untuk dapat dikatakan sebagai latar depan. Hal ini dikarenakan warna piksel mereka mempunyai varians yang lebih besar dibandingkan latar belakang. Sistem ini memiliki dua parameter, yaitu *learning rate* α dan *threshold* T (proporsi data yang harus diperhitungkan/digunakan oleh latar belakang).

2.6.1 Gaussian Mixture Models

Didefinisikan nilai sebuah piksel berturut-turut sebagai "*pixel process*". *Pixel process* adalah serangkaian nilai piksel, nilai piksel tersebut dapat berupa skalar untuk citra *grayscale* dan vektor untuk citra berwarna (r, g, b). Suatu hal yang dapat diketahui dari sebuah piksel x_0, y_0 pada waktu t adalah riwayat dari piksel itu sendiri.

$$\{X_1, \dots, X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\} \quad (2.4)$$

Dimana I adalah serangkaian citra. Jika perubahan cahaya terjadi di dalam *scene*, perubahan itu harus tetap dilacak oleh distribusi-distribusi Gaussian tersebut. Jika sebuah objek statis masuk ke dalam *scene* dan tidak langsung masuk ke dalam distribusi latar belakang sampai objek tersebut sudah lebih lama ada di area tersebut dari objek sebelumnya, maka objek tersebut akan dikategorikan sebagai latar depan dalam waktu yang cukup lama. Hal ini akan berujung pada estimasi latar depan yang buruk. Misalnya dahan pohon yang bergerak karena tertutup angin, tentu tidak diinginkan objek tersebut dideteksi sebagai objek bergerak oleh sistem.

Hal di atas adalah faktor yang menjadi landasan utama Stauffer dalam menentukan model dan prosedur updatenya. Riwayat dari setiap piksel $\{X_1, \dots, X_t\}$ dimodelkan oleh K distribusi Gaussian yang probabilitas nilai pikselnya didefinisikan oleh fungsi berikut

$$P(X_t) = \sum_{i=1}^K \omega_{i,t} \cdot \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \quad (2.5)$$

Dimana K adalah jumlah distribusi Gaussian, $\omega_{i,t}$ adalah estimasi dari *evidence / weight / bobot* (proporsi data yang direpresentasikan oleh Gaussian yang bersangkutan), $\mu_{i,t}$ adalah nilai rata-rata dari Gaussian ke- i pada waktu t , $\Sigma_{i,t}$ adalah kovarians matriks Gaussian ke- i pada waktu t , dan η adalah fungsi probabilitas Gaussian

$$\eta(X_t, \mu_t, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma^{-1} (X_t - \mu_t)} \quad (2.6)$$

Dengan kovarians matriks

$$\Sigma_{i,t} = \sigma_k^2 I \quad (2.7)$$

K ditentukan oleh kemampuan komputasi komputer. Untuk nilai bawaan $K = 5$ diset dan dapat diatur sesuai dengan kebutuhan pengguna untuk mendapatkan hasil terbaik.

2.6.2 Modeling Process

Distribusi intensitas untuk setiap nilai piksel dimodelkan oleh *mixture of Gaussian*. Secara garis besar, jika ada nilai piksel baru yang masuk ke dalam model, piksel tersebut akan direpresentasikan oleh salah satu komponen atau distribusi Gaussian yang ada dan digunakan untuk memperbarui model tersebut. Teknik *on-line K-means approximation* digunakan untuk memodelkan distribusi nilai masing-masing piksel.

Langkah pertama yang harus dilakukan adalah, inisialisasi Gaussian η untuk memodelkan nilai piksel pada observasi X_1 dengan $\omega = 1$, $\mu = X_1$, dan varians σ

yang besar. Observasi X_1 adalah observasi nilai sebuah piksel pada waktu t . Selanjutnya, setiap piksel baru X_t diperiksa terhadap K distribusi Gaussian yang sudah ada, sampai terdapat "*match*". Sebuah nilai piksel dapat dikatakan "*match*" dengan K distribusi Gaussian jika nilai piksel terletak dalam jangkauan 2,5 dari standar deviasi distribusi tersebut. Jika terdapat "*match*", parameter (ω, μ, σ) dari distribusi Gaussian diperbarui sebagai berikut

$$\omega_{k,t} = (1 - \alpha)\omega_{k,t-1} + \omega(M_{k,t}) \quad (2.8)$$

$$\mu_t = (1 - \rho)\mu_t - 1 + \rho X_t \quad (2.9)$$

$$\sigma_t^2 = (1 - \rho)\sigma_t^2 - 1 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \quad (2.10)$$

Dimana α adalah *learning rate*, $M_{k,t}$ bernilai 1 untuk distribusi yang "*matched*", dan 0 untuk distribusi yang lainnya. Lalu

$$\rho = \alpha \eta(X_t | \mu_t, \sigma_t) \quad (2.11)$$

Jika tidak ada distribusi K yang "*match*" dengan nilai piksel baru tersebut, maka buat distribusi baru yang menggunakan nilai piksel baru sebagai rata-ratanya, nilai varians awal yang besar, dan juga bobot yang rendah. Distribusi yang paling kecil kemungkinannya akan digantikan oleh distribusi baru jika $KGaussians > K$.

2.6.3 Background Model Estimation

Dalam tahap ini, ditentukan distribusi Gaussian mana yang merupakan bagian dari latar belakang. Secara heuristik, distribusi Gaussian yang memiliki bobot yang paling tinggi dan juga varians yang paling kecil adalah distribusi yang dapat diklasifikasikan sebagai latar belakang. Untuk memahami hal tersebut, diberikan sebuah objek statis yang mempunyai distribusi nilai piksel dengan bobot

yang tinggi dan juga varians yang kecil. Objek statis ini dapat dikatakan sebagai *background* dalam suatu *scene* video. Ketika terdapat objek baru yang melewati objek statis tersebut, nilai piksel objek baru tidak akan "*match*" dengan salah satu distribusi Gaussian yang sudah ada sebelumnya. Hal ini akan berujung pada pembuatan distribusi baru atau peningkatan nilai varians dari distribusi yang sudah ada. Selain itu, varians dari objek bergerak diasumsikan bernilai lebih besar daripada varians latar belakang. Untuk memodelkan hal tersebut, dibutuhkan sebuah metode yang dapat menentukan bagian mana dari model distribusi Gaussian yang secara akurat merepresentasikan latar belakang.

Pertama, distribusi Gaussian diurutkan berdasarkan nilai ω/σ dari masing-masing distribusi. Kemudian, *background model* dipilih berdasarkan fungsi berikut, dimana T adalah sebuah ukuran untuk menentukan seberapa besar bagian dari data yang dapat diklasifikasikan sebagai latar belakang.

$$B = \operatorname{argmin}_b \left(\sum_{k=1,b} \omega_k > T \right) \quad (2.12)$$

2.7 Operasi Morfologi

Morfologi dalam konteks pengolahan citra adalah deskripsi bentuk dan struktur dari sebuah objek di dalam citra. Berdasarkan hal tersebut, operasi morfologi adalah sebuah operasi yang diaplikasikan terhadap bentuk dan struktur sebuah objek di dalam citra. Teknik ini bekerja berdasarkan teori himpunan. Operasi Morfologi mengambil input berupa citra biner ataupun citra *grayscale* (Srisha dan Khan, 2013). Operasi Morfologi dapat digunakan untuk memperbesar atau memperkecil ukuran objek. Operasi morfologi juga dapat digunakan untuk menutup ataupun membuka lubang pada sebuah objek dalam citra.

Operasi morfologi melakukan "pemindaian" terhadap sebuah citra dengan menggunakan *Structuring element*. *Structuring element* adalah sebuah matriks biner yang biasanya direpresentasikan sebagai matriks persegi berdimensi ganjil.

Mekanisme pemindaian citra menggunakan *structuring element* sangat menyerupai *kernel / mask* pada operasi konvolusi. Akan tetapi, alih-alih melakukan operasi konvolusi, *structuring element* hanya melakukan operasi logika sederhana. Interaksi antara *structuring element* dengan citra yang ingin diperiksa akan menghasilkan citra baru berdasarkan jenis operasi morfologi yang diterapkan pada citra tersebut.

1	1	1		
1	1	1		
1	1	1		

0	1	0		
1	1	1		
0	1	0		

0	0	1	0	0
0	1	1	1	0
1	1	1	1	1
0	1	1	1	0
0	0	1	0	0

		1		
	1	1	1	
1	1	1	1	1
	1	1	1	
		1		

Gambar 2.3: Representasi *Structuring Element*
Sumber: Srisha dan Khan (2013)

2.8 Jenis - Jenis Operasi Morfologi

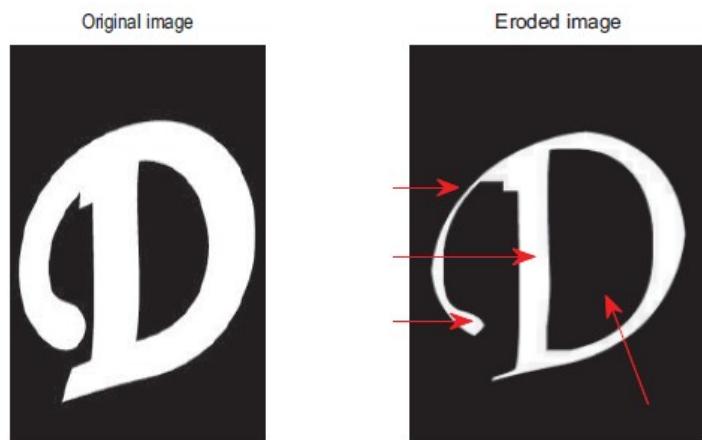
Walaupun operasi morfologi berdasar pada teori himpunan, banyak dari jenis operasi ini yang secara garis besar hanyalah operasi logika sederhana dan sangat mudah untuk digunakan. Terdapat dua jenis operasi fundamental dari operasi morfologi, yaitu erosi dan dilasi. Jenis operasi yang lain sangat bergantung kepada dua operasi fundamental tersebut.

2.8.1 Erosi

Operasi erosi membuat objek pada citra menjadi lebih kecil. Sederhananya, piksel di sekitar garis tepi sebuah objek akan dihilangkan. Erosi dapat digunakan untuk menghilangkan *noise* pada citra dan juga memisahkan objek satu dengan yang lainnya. Erosi dari sebuah citra A oleh *structuring element* B dapat didefinisikan oleh fungsi berikut

$$A \ominus B = \{z | (B)_z \subseteq A\} \quad (2.13)$$

Structuring element yang sudah dibuat akan memindai seluruh piksel di dalam citra dari mulai kiri ke kanan, dan dari atas ke bawah. Sebuah piksel tengah dari *structuring element* akan dibiarkan bernilai 1 jika seluruh piksel di dalam *structuring element* bernilai 1. Jika tidak, piksel akan dihilangkan atau diset nilainya menjadi 0 (*background*).



Gambar 2.4: Operasi Erosi
Sumber: Srisha dan Khan (2013)

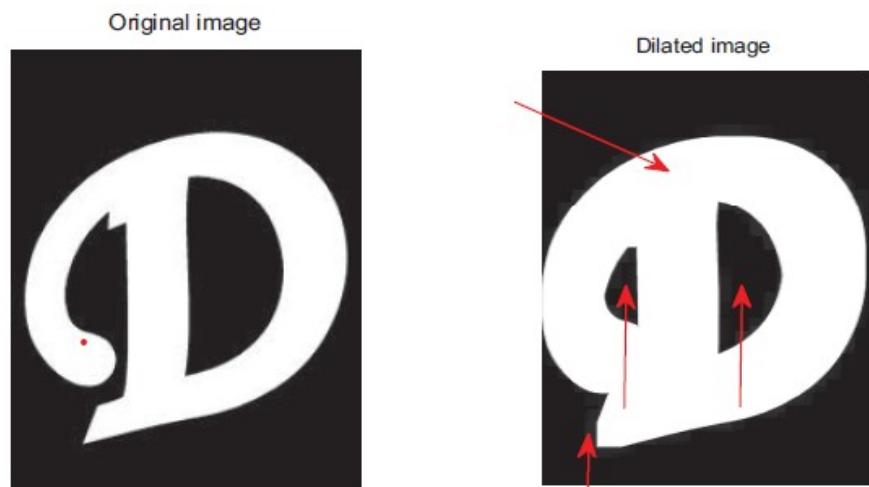
2.8.2 Dilasi

Operasi dilasi membuat objek pada citra bertambah ukurannya menjadi lebih besar. Seberapa jauh operasi ini dapat membuat objek menjadi lebih besar bergantung pada bentuk dan ukuran *structuring element*-nya. Dilasi sangat berguna

untuk menutup lubang pada objek atau menggabungkan bagian-bagian objek yang terpisah. Operasi ini dapat didefinisikan sebagai

$$A \oplus B = \{z | (\widehat{B})_z \cap A \neq \emptyset\} \quad (2.14)$$

Sama seperti erosi, dilasi juga menggunakan *structuring element* dalam proses pemindaian citra. Sebuah piksel tengah dari *structuring element* diset nilaiinya menjadi 1 jika *structuring element* memiliki paling tidak satu piksel yang bernilai 1.



Gambar 2.5: Operasi Dilasi
Sumber: Srisha dan Khan (2013)

2.8.3 *Opening*

Opening adalah erosi yang diikuti oleh dilasi. Jenis operasi ini sangat berguna untuk menghilangkan objek yang sangat kecil atau *noise* pada citra. *Opening* citra A oleh *structuring element* B dapat didefinisikan sebagai

$$A \circ B = (A \ominus B) \oplus B \quad (2.15)$$



Gambar 2.6: Operasi *Opening*
Sumber: OpenCV - Opening

2.8.4 *Closing*

Closing adalah kebalikan dari operasi *opening*. *Closing* adalah dilasi yang diikuti oleh erosi. Seperti namanya, operasi *closing* digunakan untuk menutup lubang di dalam sebuah objek atau menyambungkan bagian-bagian objek yang terpisah. *Closing* citra A oleh *structuring element* B dapat didefinisikan sebagai

$$A \circ B = (A \oplus B) \ominus B \quad (2.16)$$

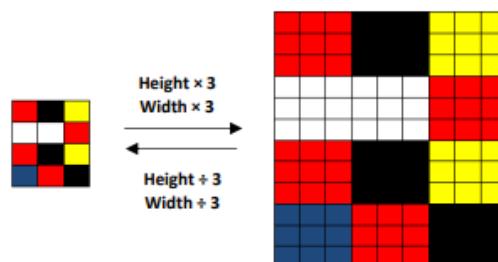


Gambar 2.7: Operasi *Closing*
Sumber: OpenCV - Closing

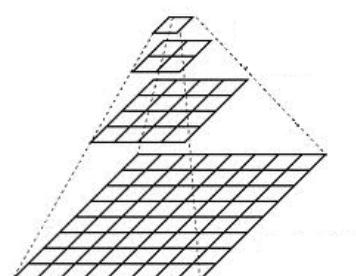
2.9 *Downsampling*

Downsampling merupakan salah satu operasi fundamental yang sering digunakan dalam pengolahan citra. *Downsampling* adalah proses mereduksi resolusi spasial dari sebuah citra dengan tetap mempertahankan representasi 2D dari citra tersebut. Operasi ini biasanya digunakan untuk mengurangi ukuran penyimpanan dari sebuah citra. Beberapa metode standar untuk melakukan operasi ini diantaranya adalah *decimation* dan *averaging*.

Decimation adalah proses mengeliminasi piksel dari sebuah citra. Terdapat beberapa *pattern* yang biasa digunakan, seperti menghapus setiap baris dan kolom piksel yang bermotor genap. Metode lainnya adalah *averaging*, yang menggabungkan beberapa piksel menjadi satu buah piksel dengan cara *averaging* (dicari rata-ratanya).



Gambar 2.8: *Image Downsampling dan Upsampling*
Sumber: Paul dan Godambe (2021)



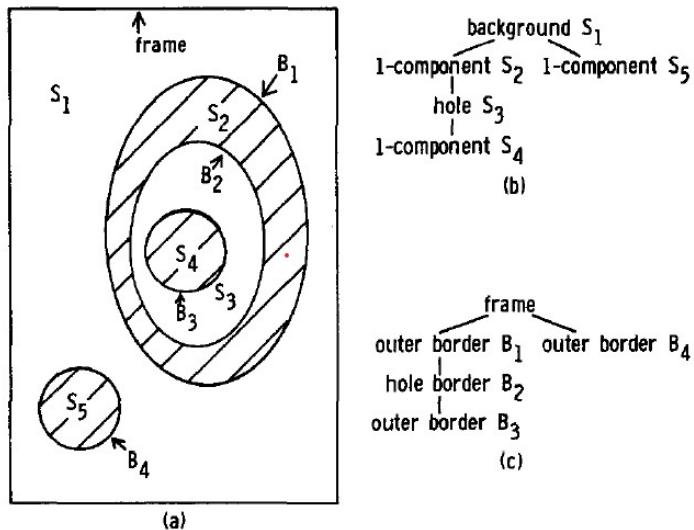
Gambar 2.9: Piramid Citra
Sumber: OpenCV - Image Pyramid

2.10 Contour Tracing

Contour Tracing atau *Border Following* adalah salah satu teknik yang sangat penting dalam bidang pengolahan citra digital, terutama pada citra biner. *Border following* telah dipelajari secara mendalam karena metode ini memiliki jangkauan pengaplikasian yang sangat luas, seperti pengenalan objek, analisis citra, deteksi objek, dan kompresi data citra (Suzuki, 1985).

Suzuki (1985) dalam penelitiannya menawarkan dua buah algoritme *border following* yang sampai saat ini masih banyak digunakan oleh berbagai macam *library* pengolahan citra modern seperti OpenCV. Algoritme ini adalah salah satu algoritme yang pertama kali mendefinisikan hubungan hierarki antar tepi (*border*). Selain itu, algoritme ini juga dapat membedakan tepi luar (*outer border*) dan tepi lubang (*hole border*) di sebuah objek. Beberapa definisi yang terdapat pada algoritme ini antara lain:

- **Definisi 1 (*border point*).** Sebuah piksel bernilai 1 yang dikelilingi oleh piksel bernilai 0 dapat disebut sebagai *border point*.
- **Definisi 2 (komponen yang mengelilingi komponen lain).** Dari Gambar 2.10 dapat dikatakan bahwa komponen S_2 mengelilingi komponen S_4 .
- **Definisi 3 (*outer border* dan *hole border*).** Sebuah rangkaian *border point* antara komponen yang mempunyai nilai piksel 1 (objek) dengan komponen yang mempunyai nilai piksel 0 (latar belakang).
- **Definisi 4 (*parent border*).** Dari Gambar 2.10 dapat dikatakan bahwa tepi B_2 adalah *parent border* dari tepi B_3 .
- **Definisi 5 (tepi yang mengelilingi tepian lain).** Dari Gambar 2.10 dapat dikatakan bahwa tepi B_2 mengelilingi tepi B_3 .



Gambar 2.10: Hubungan antar komponen dan tepi

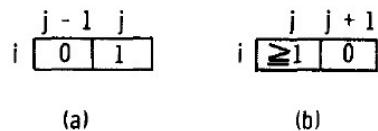
Sumber: (Suzuki, 1985)

Diasumsikan citra masukan algoritme ini berupa citra biner. Piksel dengan nilai 0 dan 1 disebut sebagai 0-piksel dan 1-piksel. Piksel bernilai 0 merepresentasikan latar belakang, sementara piksel bernilai 1 merepresentasikan latar depan (objek). Didefinisikan $f_{i,j}$ sebagai nilai sebuah piksel pada koordinat (i,j) yang masing-masing melambangkan baris dan kolom sebuah citra digital. Baris paling atas, baris paling bawah, kolom paling kiri, dan kolom paling kanan adalah *frame* dari citra tersebut. Dalam hal ini, diberikan angka unik untuk setiap tepi baru yang ditemukan dan bisa disebut sebagai **NBD**. Asumsikan NBD dari *frame* adalah 1, sementara tepi lain diberikan angka NBD secara berurutan. Nilai NBD dari *parent* setiap tepi di dalam variabel **LNBD** kemudian disimpan. Berikut adalah langkah-langkah *border following* algoritme 1:

Mulai memindai piksel citra dari kiri ke kanan hingga *scanner* menemukan piksel objek $f_{i,j} \neq 0$. Lalu tentukan apakah piksel tersebut merupakan tepi luar atau tepi lubang. Untuk setiap baris baru yang dipindai, *reset LNBD* ke 1.

1. Langkah 1

- (a) Jika $f_{i,j} = 1$ dan $f_{i,j-1} = 0$, maka piksel (i, j) adalah *starting point* dari operasi *border following* untuk tepi luar (*outer border*). Inkremen *NBD* dan set $(i_2, j_2) \leftarrow (i, j - 1)$.
 - (b) Jika $f_{i,j} \geq 1$ dan $f_{i,j+1} = 0$, maka piksel (i, j) adalah *starting point* dari operasi *border following* untuk tepi lubang (*hole border*). Inkremen *NBD*, set $(i_2, j_2) \leftarrow (i, j + 1)$, dan $LNBD \leftarrow f_{i,j}$ jika $f_{i,j} > 1$.
 - (c) Jika dua kondisi di atas tidak terpenuhi, maka lanjutkan ke langkah 4.



Gambar 2.11: Kondisi *starting point* dari algoritme *border following* untuk tepi luar (a) dan tepi lubang (b)
 Sumber: (Suzuki, 1985)

Sumber: (Suzuki, 1985)

2. Langkah 2

Berdasarkan jenis tepi dari piksel (i, j) dan jenis tepi dari *LNB*D (tepi terakhir yang ditemukan), dapat ditentukan *parent border* untuk tepi piksel (i, j) seperti yang terlihat pada Tabel 2.1.

Table 2.1: Aturan penetapan *parent border*

<i>Type of the border B' with the sequential number LNBD</i>		
<i>Type of B</i>	<i>Outer border</i>	<i>Hole border</i>
<i>Outer border</i>	<i>The parent border of the border B'</i>	<i>The border B'</i>
<i>Hole border</i>	<i>The border B'</i>	<i>The parent border of the border B'</i>

3. Langkah 3

Pada langkah ini, dimulai dari *starting point* piksel (i, j) lakukan proses *border following*:

- (a) Dimulai dari piksel (i_2, j_2) , searah jarum jam periksa piksel tetangga dari piksel (i, j) sampai menemukan piksel *non-zero*. Didefinisikan piksel *non-zero* yang pertama ditemukan sebagai (i_1, j_1) . Jika piksel *non-zero* tidak ditemukan, set $f_{i,j} = -NBD$ dan lanjutkan ke langkah 4.
- (b) $(i_2, j_2) \leftarrow (i_1, j_1)$ dan $(i_3, j_3) \leftarrow (i, j)$.
- (c) Mulai dari elemen selanjutnya dari piksel (i_2, j_2) , berlawanan arah jarum jam carilah piksel *non-zero* di sekitar piksel (i_3, j_3) dan set piksel *non-zero* pertama yang ditemukan tersebut ke dalam variabel (i_4, j_4) .
- (d) Ubah nilai f_{i_3,j_3} dari piksel (i_3, j_3) seperti berikut *marking policy*:
 - i. Jika piksel $(i_3, j_3 + 1)$ dari langkah (3.a) adalah 0-piksel (piksel bernilai 0), set $(i_3, j_3 + 1) \leftarrow -NBD$
 - ii. Jika piksel $(i_3, j_3 + 1)$ dari langkah (3.a) bukan 0-piksel (piksel bernilai 0), set $(i_3, j_3 + 1) \leftarrow NBD$
 - iii. Jika kedua kondisi di atas tidak terpenuhi, jangan ubah nilai f_{i_3,j_3}

- (e) Jika $(i_4, j_4) = (i, j)$ dan $(i_3, j_3) = (i_1, j_1)$ (kembali ke *starting point*), maka lanjut ke langkah 4, Jika tidak set $(i_2, j_2) \leftarrow (i_3, j_3)$, $(i_3, j_3) \leftarrow (i_4, j_4)$ dan kembali ke langkah (3.a).

4. Langkah 4

Jika $f_{i,j} \neq 1$, maka $LNBD \leftarrow |f_{i,j}|$ dan lanjutkan pemindaian dari mulai piksel $(i, j + 1)$ untuk kembali mencari nilai piksel objek $f_{i,j} \neq 0$. Algoritme dinyatakan selesai jika *scanner* sudah sampai sudut kanan bawah dari citra tersebut (piksel terakhir).

	10	20	30	40
1				
2	22: 333- 444444444444444+	555= 66#		
3	21: 31- 417*	771+ 51= 61#		
4	2: 3- 447*	774+ 5= 6#		
5	: 3- 41* 888888888?	71+ 5= #		
6	: 31- 41* 8819999999918?	71+ 51= #		
7	31- 41* 8198 991?	71+ 51=		
8	3- 4* 81& 91?	7+ 5=		
9	31- 4* 81& AAAAAAAe 91?	7+ 51=		
10	3- 41* 81& A1BBBBBBB1e 91?	71+ 5=		
11	3- 4* 81& AAB BBAe 91?	7+ 5=		
12	3- 4* 8& A* Be 9?	7+ 5=		
13	3- 4* 81& A* CCCCC* Be 91?	7+ 5=		
14	3- 4* 8& A1* C1DDDD1* BlE 9?	7+ 5=		
15	3- 4* 8& A* C1 D1 Be 9?	7+ 5=		
16	3- 4* 8& A* C* D Be 9?	7+ 5=		
17	3- 4* 8& A* C* EE* D* Be 9?	7+ 5=		
18	3- 4* 8& A* C* E1* D* Be 9?	7+ 5=		
19	3- 4* 8& A* C* EE* D* Be 9?	7+ 5=		
20	3- 4* 8& A* C* D* Be 9?	7+ 5=		
21	3- 4* 8& A* C1* D1* Be 9?	7+ 5=		
22	3- 4* 8& A1* C1DDDD1* BlE 9?	7+ 5=		
23	3- 4* 81& A* CCCCC* Be 91?	7+ 5=		
24	3- 4* 8& A* Be 9?	7+ 5=		
25	3- 4* 81& AAB* BBAe 91?	7+ 5=		
26	3- 41* 81& A1BBBBBBB1e 91?	71+ 5=		
27	31- 4* 81& AAAAAAAe 91?	7+ 51=		
28	3- 4* 81& 91?	7+ 5=		
29	31- 41* 8198 991?	71+ 51=		
30	* 31- 41* 8819999999918?	71+ 51= *		
31	* 3- 41* 888888888?	71+ 5= *		
32	F* 3- 447*	774+ 5= G*		
33	F1* 31- 417*	771+ 51= G1*		
34	FF* 333- 444444444444444+	555= GG*		
35				

(b)

Structured borders

Border no.	Outer border (1) /Hole border (0)	Coordinates of the border following starting point	Parent border no
1	0	(1, 1)	0
2	1	(2, 3)	1
3	1	(2, 8)	1
4	1	(2, 15)	1
5	1	(2, 35)	1
6	1	(2, 41)	1
7	0	(3, 17)	4
8	1	(5, 18)	7
9	0	(7, 18)	8
10	1	(9, 19)	9
11	0	(11, 19)	10
12	1	(13, 20)	11
13	0	(15, 20)	12
14	1	(17, 22)	13
15	1	(30, 2)	1
16	1	(30, 44)	1

Gambar 2.12: Struktur topologi antara tepi satu dengan tepi yang lain jika menggunakan Algoritme 1. Hasil citra (a) dan Struktur yang diekstrak (b)

Sumber: (Suzuki, 1985)

Selain algoritme di atas, Suzuki juga mengusulkan algoritme *border following* yang hanya akan menghasilkan tepi terluarnya saja. Beberapa perbedaan algoritme 1 dan algoritme 2 antara lain adalah:

1. Kondisi *starting point* untuk melakukan operasi *border following* hanya berlaku untuk tepi terluar (Gambar 2.11) dan jika $LNBD \leq 0$.
 2. *Marking policy* tetap sama dengan algoritme 1 (Langkah 3.d), tetapi nilai dari NBD dan $-NBD$ akan digantikan masing-masing oleh nilai 2 dan -2.
 3. Nilai $LNBD$ juga tetap disimpan dari piksel *non-zero* yang ditemukan sebelumnya. Akan tetapi, $LNBD$ akan di reset ke nilai 0 untuk setiap baris baru yang dipindai.

Gambar 2.13: Hasil dari algoritme 2. Tanda "#" merepresentasikan *starting point* dan ":" merepresentasikan nilai -?

Sumber: (Suzuki 1985)

2.11 Pelacakan Objek

Pelacakan objek (*object tracking*) mengacu kepada penggunaan sensor untuk menentukan lokasi, lintasan, atau bahkan karakteristik dari sebuah objek. Dalam kasus video, pelacakan objek didefinisikan sebagai sebuah tindakan melakukan estimasi atau prediksi terhadap lintasan sebuah objek bergerak di setiap *frame* dalam video. Dengan kata lain, sebuah *tracker* dapat memberikan label / identitas yang konsisten terhadap objek yang dilacak selama video berlangsung. Selain itu, sebuah *tracker* juga dapat memberikan informasi lain yang berkaitan dengan objek tersebut, seperti orientasi, ukuran, atau bentuk geometrinya (Yilmaz et al., 2006).

Permintaan terhadap teknologi analisis video yang lebih *advanced* membuat metode pelacakan objek menjadi sangat menarik untuk dikembangkan. Penggunaan metode pelacakan objek sangat berguna dalam berbagai bidang, antara lain seperti *motion recognition*, pengawasan, pemantauan lalu lintas, navigasi kendaraan, dan masih banyak lagi. Pelacakan objek merupakan dasar teori dari metode penghitungan objek (*objek counting*).

2.12 Kalman Filter

2.12.1 Definisi

Secara matematis, Kalman Filter adalah sebuah algoritme *estimator* yang dapat melakukan prediksi dan koreksi terhadap keadaan sebuah proses linear (Chavan dan Gengaje, 2017). Model matematis yang akurat sangat diperlukan untuk mendapatkan hasil yang optimal. Kalman Filter memiliki bentuk yang relatif sederhana dan tidak membutuhkan kekuatan komputasi yang besar.

Kalman filter digunakan untuk melakukan estimasi suatu keadaan x dalam sistem linear. Model dari proses (*process model*) yang mendefinisikan evolusi suatu

keadaan dari waktu $k - 1$ sampai waktu k adalah:

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1} \quad (2.17)$$

dimana F adalah matriks transisi dari keadaan x yang diterapkan pada vektor keadaan sebelumnya x_{k-1} , B adalah matriks kontrol yang diterapkan pada vektor kontrol u_{k-1} , dan w_{k-1} adalah vektor proses *noise* yang diasumsikan sebagai Gaussian dengan rata-rata nol dan kovarians Q , i.e., $w_{k-1} \sim N(0, Q)$.

Model proses di atas dipasangkan dengan model pengukuran / observasi (*measurement / observation model*) yang menggambarkan hubungan antara keadaan dan observasi pada waktu k sebagai:

$$z_k = Hx_k + v_k \quad (2.18)$$

dimana z_k adalah vektor observasinya, H adalah matriks transisi dari observasi z , dan v_k adalah vektor *noise* dari observasi yang diasumsikan sebagai Gaussian dengan rata-rata nol dan kovarians R , i.e., $v_k \sim N(0, R)$.

Peran dari Kalman Filter adalah untuk memberikan estimasi x pada waktu k , mengingat diberikannya estimasi awal x_0 , rangkaian observasi z_1, z_2, \dots, z_k , dan informasi sistem yang digambarkan oleh F , B , H , Q , dan R . Walaupun matriks kovarians (Q dan R) seharusnya mencerminkan statistik dari *noise* masing-masing model, dalam banyak pengaplikasiannya di dunia nyata, model statistika dari *noise* tersebut tidak benar-benar dapat dimodelkan. Karena hal itu, Q dan R biasanya digunakan sebagai *tuning parameter* yang bisa digunakan untuk mendapatkan hasil dan performa yang diinginkan.

2.12.2 Algoritme Kalman Filter

Kalman Filter terdiri dari dua tahap, yaitu *predict* dan *update*. Dalam literasi lain, tahapan ini juga biasa disebut sebagai *propagation* dan *correction*. Tahapan *predict* dan *update* dapat diringkas ke dalam persamaan berikut:

Predict

$$\hat{x}_k^- = F\hat{x}_{k-1}^+ + Bu_{k-1} \quad (2.19)$$

$$P_k^- = FP_{k-1}^+ F^T + Q \quad (2.20)$$

Update

$$\tilde{y}_k = z_k - H\hat{x}_k^- \quad (2.21)$$

$$K_k = P_k^- H^T (R + HP_k^- H^T)^{-1} \quad (2.22)$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k \tilde{y}_k \quad (2.23)$$

$$P_k^+ = (I - K_k H)P_k^- \quad (2.24)$$

Dari persamaan di atas, operator hat (^) menandakan sebuah estimasi dari sebuah variabel. Dengan kata lain, \hat{x} adalah estimasi dari x . Superskrip – dan + masing-masing menyatakan estimasi yang diprediksi (*prior*) dan diperbarui (*posterior*).

Untuk memprediksi estimasi dari sebuah keadaan \hat{x}_k^- dibutuhkan estimasi keadaan \hat{x}_{k-1}^+ dari iterasi Kalman Filter sebelumnya. Variabel P adalah matriks kovarians dari keadaan x . Matriks kovarians ini digunakan untuk membantu Kalman Gain K_k dalam menentukan bobot terhadap nilai yang diprediksi atau nilai yang diukur.

Dalam tahap *update*, K_k pada Persamaan 2.22 adalah Kalman Gain. Nilai dari Kalman Gain akan berada diantara 0 dan 1. Kemudian nilai tersebut digunakan

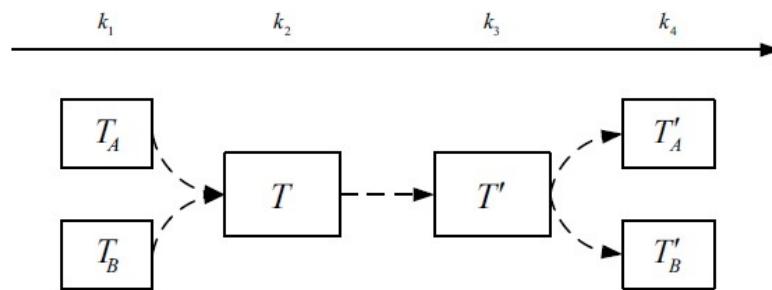
untuk memperbarui estimasi keadaan \hat{x}_k^+ . Jika nilai dari Kalman Gain mendekati 1, maka observasi dikatakan akurat dan estimasi dikatakan tidak akurat (*error* besar). Begitu juga sebaliknya, jika K_k mendekati 0, maka observasi dikatakan tidak akurat dan estimasi dapat dikatakan akurat (*error* kecil). Akibatnya, nilai K_k yang kecil akan membuat selisih antara keadaan yang diobservasi dan keadaan yang diestimasi mempunyai efek yang kecil terhadap proses pembaruan keadaan (Persamaan 2.23). Residual pengukuran \tilde{y}_k adalah selisih antara nilai yang diobservasi z_k , dengan nilai yang diestimasi $H\hat{x}_k^-$. Residual pengukuran \tilde{y}_k kemudian dikalikan dengan Kalman Gain K_k untuk memberikan koreksi $K_k\tilde{y}_k$ terhadap estimasi keadaan \hat{x}_k^- (Persamaan 2.23). Setelah mendapatkan estimasi keadaan yang baru, Kalman Filter menghitung kovarians P_k^+ untuk kemudian digunakan pada tahap atau iterasi Kalman Filter selanjutnya. Dapat dilihat bahwa kovarians P_k^+ mempunyai nilai yang lebih kecil dibandingkan kovarians P_k^- , yang berarti Kalman Filter menjadi lebih yakin dengan estimasi keadaan setelah tahap *update*.

Dibutuhkan tahap inisialisasi untuk dapat mengimplementasi Kalman Filter. Dalam tahap tersebut, dibutuhkan tebakan awal dari \hat{x}_0^+ dan juga matriks kovarians P_0^+ . Bersama dengan Q dan R , \hat{x}_0^+ dan P_0^+ memainkan peran yang sangat penting untuk mendapatkan hasil dan performa yang diinginkan. Sebagai "*rule of thumb*", kovarians P_0^+ dapat diset ke nilai yang besar untuk konvergensi yang lebih cepat. Secara keseluruhan, Kalman Filter dapat diimplementasi dengan menerapkan tahap *predict* dan tahap *update* untuk setiap iterasi waktu $k = 1, 2, 3, \dots$ setelah tahap inisialisasi dilakukan.

Perlu dicatat bahwa Kalman Filter bekerja berdasarkan asumsi bahwa model proses (*process model*) dan model pengukuran (*measurement model*) merupakan model yang linear, yang berarti model-model tersebut dapat diekspresikan dengan matriks F, B, dan H. Oleh karena itu, Kalman Filter dapat memberikan hasil estimasi yang optimal jika dan hanya jika asumsi di atas terpenuhi.

2.13 Decision of Occlusion

Occlusion adalah sebuah peristiwa dimana dua atau lebih objek saling bertumpuk satu sama lain. Hal ini tentu dapat berpengaruh terhadap performa sistem jika tidak ditangani dengan baik. *Occlusion* pada dasarnya mencakup dua masalah, yaitu masalah *merge* dan *split*. Menyatunya dua atau lebih objek disebut sebagai proses *merge*. Sebaliknya, terpisahnya objek yang sebelumnya menyatu disebut sebagai proses *split*. Proses *merge* dan *split* diilustrasikan pada Gambar 2.14.



Gambar 2.14: Ilustrasi dari proses *merge* dan *split*

Sumber: (Li et al., 2010)

Ketika dua atau lebih keadaan objek yang diprediksi menggunakan Kalman Filter mempunyai pasangan observasi (deteksi) yang sama, maka akan terjadi proses *merge* sebagai bagian dari salah satu proses *decision of occlusion*. Dua buah objek T_A dan T_B mengalami proses *merge* pada waktu k_1 . Dimulai dari waktu k_2 , objek T akan dilacak sebagai objek baru, Kalman Filter diinisialisasi untuk objek tersebut. Selama waktu k_2 sampai k_3 , objek T akan terus diupdate bersamaan dengan objek T_A dan objek T_B . Objek T kini menjadi objek T' . Pada waktu k_4 , terjadi proses *split* terhadap objek T' menjadi objek T'_A dan T'_B . Kedua objek tersebut kemudian dipasangkan kembali dengan objek T_A dan T_B , membuat objek T' hasil dari proses *merge* sebelumnya kehilangan pasangan observasinya. Jika selama k_n objek T' terus menerus kehilangan pasangan observasinya, maka objek T' akan dihapus.

2.14 Asosiasi Data

Pada proses asosiasi data, prediksi estimasi keadaan dari sebuah objek yang dihasilkan oleh Kalman Filter kemudian diasosiasikan dengan observasi keadaan objek pada *frame* selanjutnya, agar identitas dari objek tersebut dapat dipertahankan selama video berlangsung. Penelitian ini berfokus kepada pelacakan lebih dari satu objek. Dalam lingkungan yang mempunyai banyak objek, bisa didapatkan hasil pelacakan lebih dari satu objek. Setiap objek ikan dalam video dideskripsikan oleh *bounding box* dan titik pusatnya masing-masing. Untuk mendapatkan hasil pelacakan objek yang akurat, proses asosiasi data perlu dilakukan dengan menggunakan jarak antara keadaan objek yang diestimasi dengan hasil observasi serta luas dari objek tersebut sebagai *cost function*. Secara garis besar, *cost function* ini digunakan untuk mengklasifikasikan objek hasil dari prediksi Kalman Filter dengan objek hasil dari observasi (deteksi). Pertama, jarak antara dua titik pusat didefinisikan sebagai:

$$D_k(i, j) = \frac{|\sqrt{(p_{k-1}^j - p_k^i)^2 + (q_{k-1}^j - q_k^i)^2}|}{\max|(p_{k-1}^j - p_k^i)^2 + (q_{k-1}^j - q_k^i)^2|} \quad (2.25)$$

dimana p_{k-1}^j dan q_{k-1}^j adalah nilai titik pusat dari koordinat x dan y yang didapatkan dari keadaan \hat{x}_{k-1}^+ dalam iterasi ke j^{th} . Lalu p_k^i dan q_k^i adalah nilai titik pusat dari koordinat x dan y yang didapatkan dari observasi z_k dalam iterasi ke i^{th} . Variabel $i = 1, \dots, m$ dan $j = 1, \dots, n$ dimana m adalah jumlah objek yang terdeteksi dari hasil observasi pada *frame* k dan n adalah jumlah objek yang dihasilkan oleh estimasi Kalman Filter pada *frame* $k - 1$. Kedua, luas antara dua objek didefinisikan sebagai:

$$A_k(i, j) = \frac{|A_k^i - A_{k-1}^j|}{\max|A_k^i - A_{k-1}^j|} \quad (2.26)$$

Dimana $A_k^i = 4l_k^i h_k^i$ dan $A_{k-1}^j = 4l_{k-1}^j h_{k-1}^j$ yang merepresentasikan *bounding box*. Semakin kecil nilai fungsi di atas, semakin mirip deskripsi bentuk

dari dua objek tersebut. Dengan mengombinasikan persamaan 4.2 dan persamaan 3.8, didefinisikan *cost function* sebagai berikut.

$$C_k(i, j) = \alpha D_k(i, j) + \beta A_k(i, j) \quad (2.27)$$

Dimana $\alpha + \beta = 1$, parameter ini dapat diatur secara eksperimental, dalam hal ini digunakan $\alpha = 0.8$ dan $\beta = 0.2$. Variabel α dan β menitik beratkan *cost function* mana yang akan lebih besar digunakan nilainya. Tidak ada aturan khusus dalam menentukan α dan β . Semakin besar nilai α maka sistem akan semakin bias terhadap *cost function* jarak objek. Sebaliknya, semakin besar nilai β , maka sistem akan semakin bias terhadap *cost function* luas objek. Jika hasil dari *cost function* di atas bernilai kecil, dapat diasumsikan bahwa dua objek tersebut memiliki korespondensi.

BAB III

METODE PENELITIAN

3.1 Deskripsi Sistem

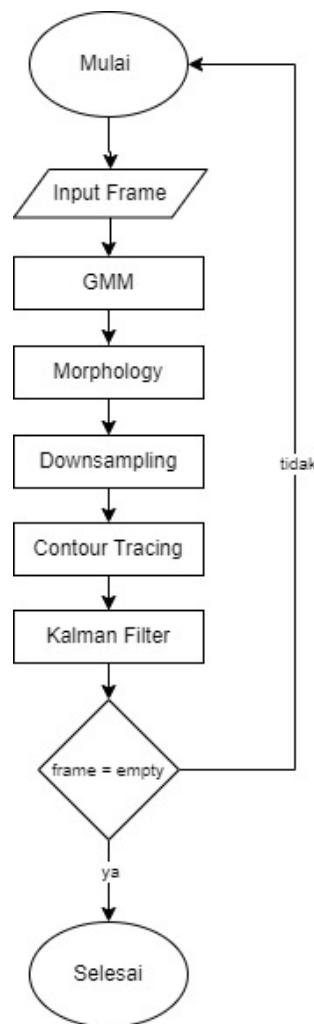
Sistem yang akan dibuat pada penelitian ini adalah sistem yang dapat melakukan pelacakan pergerakan objek ikan dengan menggunakan metode GMM dan Kalman Filter. Fokus dari penelitian ini adalah untuk menghasilkan *output* berupa video yang di dalamnya terdapat objek yang sudah dilacak lengkap dengan *bounding box* serta label uniknya. Dataset video yang akan digunakan pada sistem ini berasal dari dua sumber. Sementara bahasa pemrograman yang digunakan untuk membuat sistem adalah Python versi 3.

Tahapan proses pelacakan pergerakan objek ikan dengan menggunakan metode GMM dan Kalman Filter adalah memasukan *input* video, deteksi objek bergerak dengan GMM, *noise removal* dengan *Morphology Operation*, *Contour Tracing*, *tracking* masing-masing objek menggunakan Kalman Filter, *decision of occlusion*, dan terakhir adalah tahap asosiasi data. Tahapan-tahapan tersebut akan terus berulang selama video berlangsung.

3.2 Perancangan Sistem

Bagian ini membahas tentang desain proses yang digunakan untuk mengetahui tahapan-tahapan yang dilakukan untuk membangun sistem pelacakan pergerakan objek ikan. Tahapan tersebut adalah sebagai berikut: tahap *input* video yang kemudian dibaca oleh sistem dalam bentuk citra (*frame per frame*). Lalu dilakukan proses deteksi objek ikan yang bergerak menggunakan GMM, setelah itu dilakukan proses *Morphology Operation* untuk menghilangkan *noise* dari proses deteksi objek menggunakan GMM. Kemudian teknik *downsampling* dan *contour tracing* dijalankan untuk menghasilkan *border* dari piksel yang dapat dikatakan

sebagai objek. *Bounding box* kemudian dipasangkan kepada masing-masing objek tersebut dan *decision of occlusion* dilakukan. Terakhir Kalman Filter diinisiasi atau diperbarui untuk setiap objek yang ada. Hal ini dilakukan untuk memasangkan atau mempertahankan label unik masing-masing objek sehingga objek dapat terus dilacak atau diasosiasikan dengan tepat selama video berlangsung. Keluaran yang dihasilkan adalah sebuah *running video* yang di dalamnya terdapat objek-objek yang sudah dilacak lengkap dengan label unik serta *bounding box*-nya.



Gambar 3.1: Diagram alir keseluruhan proses sistem

3.2.1 Proses Input Video

Data yang digunakan sebagai masukan dari sistem adalah video dengan format .mp4 atau .flv. Masukan video dibaca ke dalam bentuk citra sehingga menghasilkan *frame* video. Setiap *frame* video juga melalui proses *pre-processing* dahulu sebelum masuk ke tahap deteksi objek dengan metode GMM yaitu, mengubah ruang warna citra dari RGB ke *grayscale*.

3.2.2 Keluaran

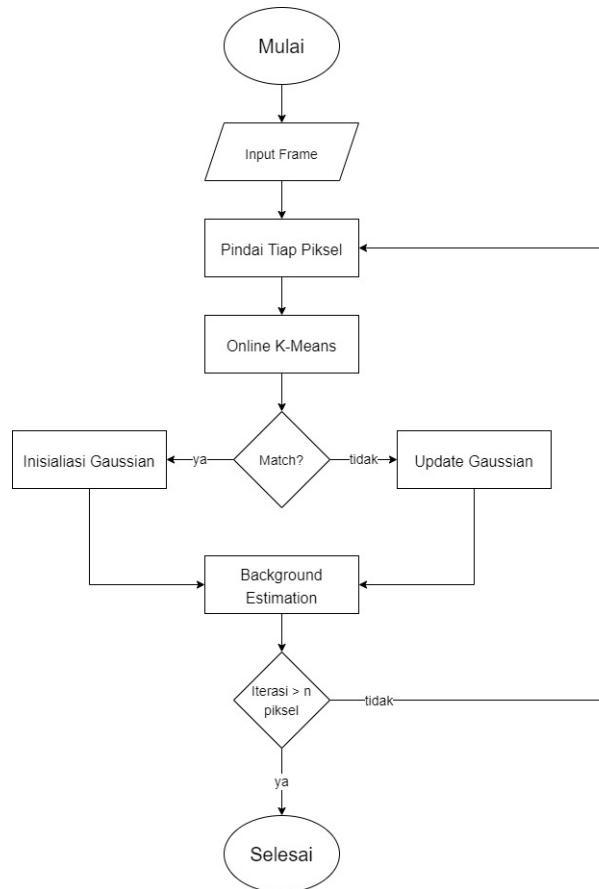
Keluaran yang diharapkan dari sistem adalah *running* video yang di dalamnya sudah terdapat objek yang dilacak menggunakan metode GMM dan Kalman Filter lengkap dengan *bounding box* serta label unik yang mengikutinya.

3.2.3 Deteksi Objek dengan GMM

Metode yang digunakan untuk mendeteksi objek ikan bergerak pada video adalah *Background Subtraction* dengan bantuan GMM untuk memodelkan latar belakangnya. Langkah pertama pada GMM adalah inisialisasi Gaussian η untuk memodelkan nilai piksel pada observasi X_1 dengan $\omega = 1$, $\mu = X_1$, dan varians σ . Observasi X_1 adalah observasi nilai sebuah piksel pada waktu t .

Selanjutnya, setiap piksel baru X_t akan diperiksa terhadap K distribusi Gaussian yang sudah ada sampai terdapat "*match*". Sebuah piksel dapat dikatakan "*match*" jika nilai piksel tersebut terletak dalam jangkauan 2,5 dari nilai standar deviasi sebuah distribusi. Jika terdapat "*match*" maka parameter (ω, μ, σ) dari distribusi Gaussian tersebut diperbaharui. *Learning rate* α merupakan parameter dari sistem. Jika tidak terdapat "*match*", maka distribusi lama yang mempunyai nilai kemungkinan paling kecil akan digantikan oleh distribusi baru. Distribusi baru ini menggunakan nilai piksel baru sebagai rata-ratanya, memiliki nilai varians awal yang besar, dan juga bobot yang rendah.

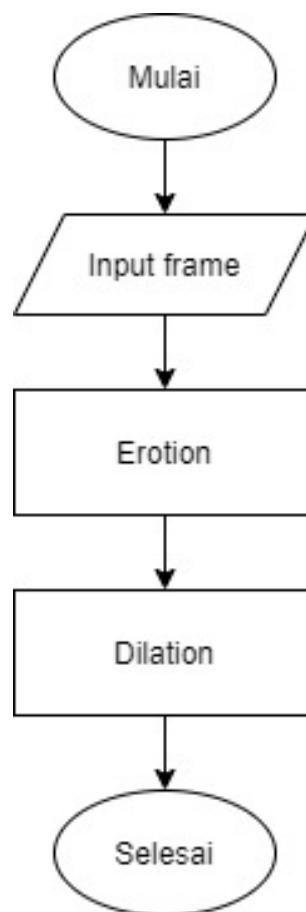
Distribusi K kemudian diurutkan berdasarkan nilai ω/σ dari masing-masing distribusi dan distribusi B pertama dijadikan sebagai model dari latar belakang. Penentuan distribusi B pertama sebagai model latar belakang ditentukan oleh parameter T . *Threshold T* adalah sebuah nilai minimum yang digunakan untuk menentukan distribusi B mana yang dapat dikatakan sebagai latar belakang. Untuk setiap distribusi B yang tidak masuk ke dalam klasifikasi latar belakang dapat dikatakan sebagai latar depan (objek bergerak). Keluaran dari proses ini adalah citra biner dengan nilai 0 sebagai latar belakang dan nilai 1 sebagai latar depannya. Alur diagram proses *Background Subtraction* dengan bantuan GMM dapat dilihat pada Gambar 3.2.



Gambar 3.2: Diagram alir proses *Background Subtraction* dengan GMM

3.2.4 Proses Menghilangkan Noise Melalui Operasi Morfologi

Keluaran dari proses deteksi objek menggunakan metode *Background Subtraction* dengan bantuan GMM adalah sebuah citra biner dengan nilai piksel 0 sebagai latar belakang dan nilai piksel 1 sebagai latar depan. Citra hasil keluaran proses tersebut masih berisikan banyak *noise* (piksel latar depan yang bukan objek). Maka dari itu, operasi morfologi diperlukan untuk menghilangkan *noise* tersebut.



Gambar 3.3: Diagram alir Operasi Morfologi

Citra biner akan melalui dua tahap operasi morfologi, yaitu erosi yang disusul oleh dilasi (*opening*). Erosi sesuai dengan namanya adalah proses pengikisan piksel. Dalam hal ini, erosi dapat menghilangkan *noise* pada citra biner terutama *salt and pepper noise* (derau kecil pada citra yang terlihat seperti garam bertaburan).

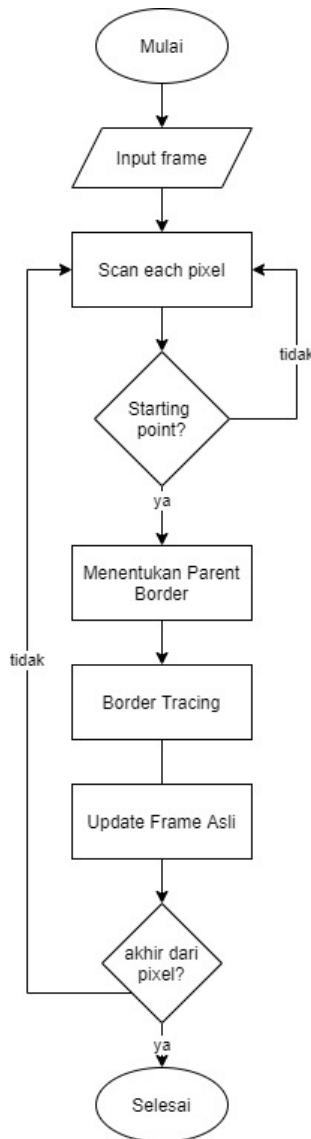
Structuring element yang akan digunakan oleh proses erosi berukuran 5×5 dengan bentuk kotak. Proses dilasi selanjutnya dilakukan untuk mengembalikan objek latar depan ke bentuk semula yang sebelumnya terkikis oleh proses erosi. Dilasi dapat memperbesar objek latar depan dan juga menyambungkan kembali bagian-bagian objek yang terpisah akibat proses erosi. *Structuring element* yang akan digunakan oleh proses dilasi berukuran 11×11 dengan bentuk kotak. Keluaran yang diharapkan adalah citra biner yang bersih dari *noise* serta objek latar depan yang utuh (tidak terpotong ataupun berlubang). Gambar 3.3 memperlihatkan diagram alir dari alur operasi morfologi secara keseluruhan.

3.2.5 *Downsampling* dan *Contour Tracing*

Downsampling terhadap sebuah *frame* dijalankan sebelum algoritme CT dieksekusi. Hal ini dilakukan untuk meningkatkan perfoma CT. Algoritme dua dari metode *contour tracing / border following* oleh Suzuki digunakan pada sistem ini. Algoritme dua hanya berfokus pada deteksi tepi bagian terluar (*outer border*), karena pada sistem kali ini tidak diperlukan deteksi tepi bagian dalam (*hole border*). Algoritme ini dijalankan dengan asumsi bahwa citra masukan merupakan citra biner dengan piksel bernilai 0 sebagai latar belakang dan piksel bernilai 1 sebagai latar depan (objek).

Masukan berupa citra biner dari proses operasi morfologi yang kemudian dilakukan proses pemindaian (*scan*) mulai dari piksel paling kiri atas. Ketika pemindai sampai pada sebuah piksel (i, j) bernilai $f_{i,j} \neq 0$, ditentukan apakah piksel tersebut merupakan *starting point* dari operasi *border following* untuk tepi luar atau tepi dalam (Gambar 2.11). Lalu *parent border* untuk piksel (i, j) ditentukan berdasarkan Gambar 2.13. Langkah selanjutnya adalah melakukan operasi *border following* dimulai dari *starting point* piksel (i, j) sampai pointer kembali ke posisi *starting point* (3.e). Dikarenakan algoritme yang digunakan adalah algoritme dua, nilai *NBD* dan $-NBD$ akan diset masing-masing 2 dan -2 . Kemudian *scanner*

melanjutkan pemindaian untuk kembali mencari nilai piksel $f_{i,j} \neq 0$. Algoritme dinyatakan selesai apabila *scanner* sudah sampai pada sudut kanan bawah (piksel terakhir).



Gambar 3.4: Diagram alir proses *Contour Tracing*

Proses *contour tracing* menghasilkan citra biner yang objek-objeknya sudah mempunyai tepi/konturnya masing-masing. Kemudian dari kontur tersebut dapat diekstraksi lebar dan tinggi sebuah objek. Dari nilai panjang dan lebar itulah penulis

dapat menggambarkan *bounding box* untuk masing-masing objek tersebut.

3.2.6 Decision of Occlusion, Asosiasi Data, dan Kalman Filter

Kalman Filter digunakan sebagai pelacak (*tracker*) untuk masing-masing objek yang terdeteksi. Langkah pertama pada proses pelacakan objek adalah untuk menginisialisasi Kalman Filter sebanyak jumlah objek yang terdeteksi pada proses sebelumnya. Setiap Kalman Filter dikonfigurasikan sebagai berikut:

$$x_k = \begin{bmatrix} p_k & q_k & l_k & h_k & v_{p,k} & v_{q,k} \end{bmatrix}^T \quad (3.1)$$

Dimana p_k dan q_k merepresentasikan nilai horizontal dan vertikal koordinat titik tengah, l_k dan h_k merepresentasikan setengah dari lebar dan tinggi *bounding box*, serta $v_{p,k}$ dan $v_{q,k}$ merepresentasikan kecepatan dari p_k dan q_k .

Kemudian untuk model pengukuran (*measurement model*) yang digunakan oleh sistem ini didefinisikan sebagai berikut:

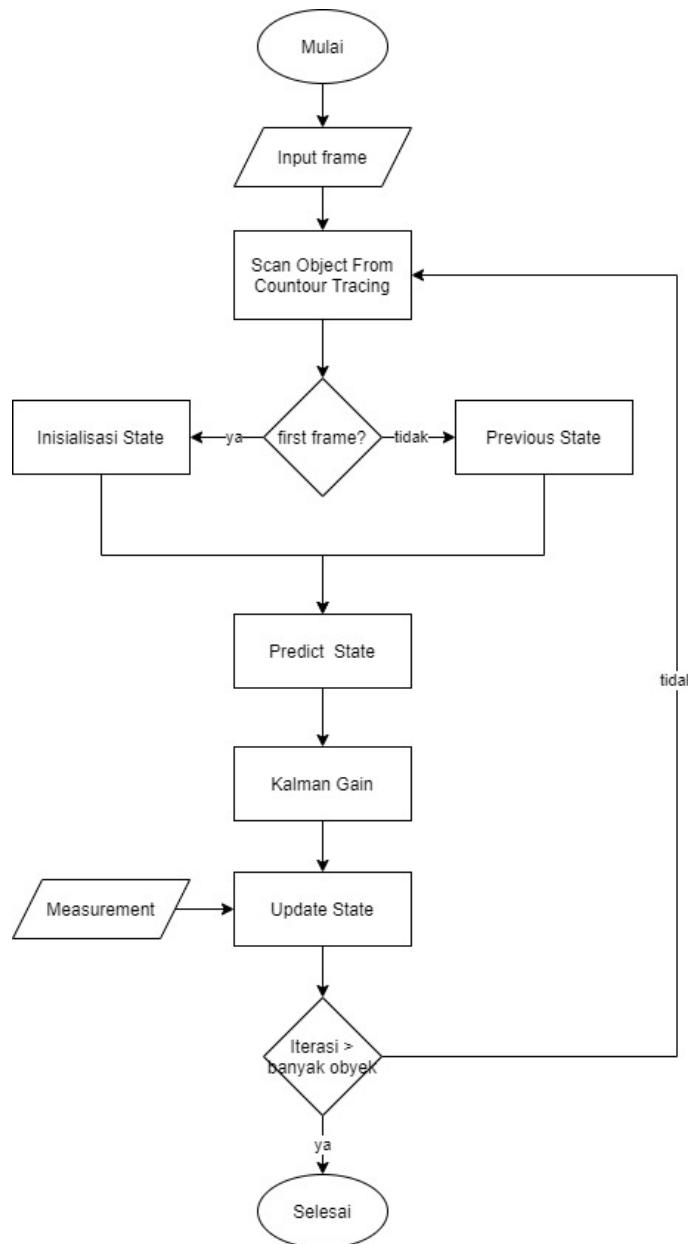
$$z_k = \begin{bmatrix} p_k & q_k & l_k & h_k \end{bmatrix}^T \quad (3.2)$$

Di bawah ini dimodelkan matriks transisi F dan matriks transisi H untuk masing-masing model proses dan model pengukuran:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad (3.4)$$

Setelah melalui proses pendefinisian model proses dan model pengukuran di atas, Kalman Filter kemudian dapat digunakan untuk memperkirakan lokasi serta ukuran sebuah objek pada *frame* selanjutnya. Diagram alir untuk proses Kalman Filter dapat dilihat pada Gambar 3.5.



Gambar 3.5: Diagram alir proses pelacakan dengan Kalman Filter

Pada proses ini, Kalman Filter membutuhkan variabel z_k yang didapatkan dari hasil deteksi objek. Untuk menentukan objek hasil deteksi mana yang sesuai dengan prediksi Kalman Filter, dilakukan proses asosiasi data terhadap jarak serta variasi luas antara objek yang diestimasi dengan objek yang dideteksi. Selain itu,

proses *decision of occlusion* juga dilakukan untuk menentukan apakah terdapat objek yang mengalami proses *merging* ataupun *splitting*. Setiap objek yang mengalami proses *merging* dengan objek lainnya akan menghasilkan objek baru yang kemudian dipasangkan dengan Kalman Filternya sendiri. Objek baru tersebut akan terus dilacak sampai proses *splitting* terjadi.



Gambar 3.6: Contoh keluaran sistem
Sumber: ResearchGate

3.3 Perancangan Eksperimen

Pada subbab ini akan dibahas mengenai rancangan eksperimen yang akan dilakukan pada penelitian ini. Eksperimen diawali dengan pengambilan data, kemudian dilanjutkan dengan pelacakan oleh sistem, dan terakhir evaluasi hasil.

3.3.1 Sumber Data

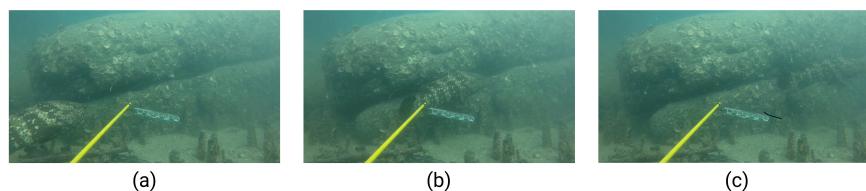
Berdasarkan sumber datanya, dataset pada penelitian ini akan dibagi menjadi dua. Secara keseluruhan, pengujian dalam penelitian ini dilakukan terhadap video ikan yang sudah direkam sebelumnya (*offline tracking*). Video tersebut kemudian dimasukkan sebagai input sistem dengan pengaturan bawaan (*default*). Pengaturan parameter dapat disesuaikan dengan kebutuhan pengguna untuk mendapatkan hasil pelacakan yang optimal.

Dataset didapatkan dari sumber internet. Diunduh dari situs *DeepFish* (Saleh et al., 2020) dan *PeRCeiVeLab* (Kavasidis et al., 2012). Dataset yang diunduh dari situs *DeepFish* berupa kumpulan gambar *frame* video yang kemudian diubah ke dalam bentuk video dengan format mp4 dengan resolusi 720 x 480 dan *frame rate* 24 fps. Sementara itu, dataset yang diunduh dari situs *PeRCeiVeLab* merupakan video dengan format *flv* berdurasi 9 menit dan dengan resolusi 320x240 serta *frame rate* 8 fps. Berdasarkan jumlah ikan dan keadaan latar belakang, terdapat empat skenario pengujian yang akan dilakukan pada penelitian ini. Keempat skenario tersebut adalah:

1. Pengujian terhadap video berisikan satu ikan dengan latar belakang sederhana
2. Pengujian terhadap video berisikan satu ikan dengan latar belakang kompleks
3. Pengujian terhadap video berisikan dua atau lebih ikan (*multiple object*) dengan latar belakang sederhana
4. Pengujian terhadap video berisikan dua atau lebih ikan (*multiple object*) dengan latar belakang kompleks

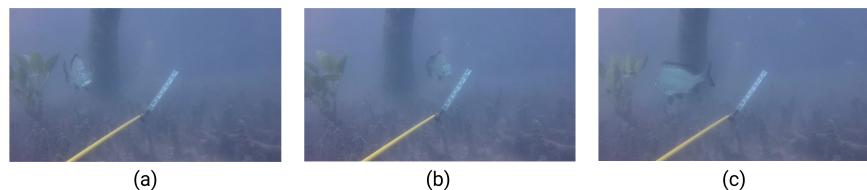
Secara keseluruhan, terdapat empat video yang akan digunakan dalam penelitian ini. Dataset video yang diambil sebagai masukan sistem adalah sebagai berikut:

1. Video skenario pengujian satu diambil dari situs *DeepFish* indeks 9908 berdurasi 12 detik (24fps). Video ini digunakan karena latar belakang video termasuk ke dalam kategori sederhana.



Gambar 3.7: Sampel *frame* video dataset *DeepFish* indeks 9908
Sumber: *DeepFish Video Dataset*

2. Video skenario pengujian dua diambil dari situs *DeepFish* indeks 9866 berdurasi 35 detik (24fps). Video ini digunakan karena latar belakang video termasuk ke dalam kategori kompleks (dinamis).



Gambar 3.8: Sampel *frame* video dataset *DeepFish* indeks 9866
Sumber: *DeepFish Video Dataset*

3. Video skenario pengujian tiga diambil dari situs *PeRCeiVeLab* indeks gt_124 berdurasi 9 menit 22 detik (8fps) yang telah dipotong menjadi 2 menit 30 detik. Video ini digunakan karena sudah cukup memenuhi kriteria skenario pengujian ke-tiga, yaitu jumlah objek lebih dari satu dengan latar belakang sederhana.



Gambar 3.9: Sampel *frame* video dataset *PeRCeiVeLab* indeks gt_124
Sumber: *DeepFish Video Dataset*

4. Video skenario pengujian empat diambil dari situs *PeRCeiVeLab* indeks gt_116 berdurasi 9 menit 35 detik (8fps) yang telah dipotong menjadi 2 menit 30 detik. Video ini digunakan karena sudah cukup memenuhi kriteria skenario pengujian ke-empat, yaitu jumlah objek lebih dari satu dengan latar belakang kompleks.



(a)

(b)

Gambar 3.10: Sampel *frame* video dataset *PeRCeVeLab* indeks gt_116
Sumber: *DeepFish Video Dataset*

3.3.2 Evaluasi Hasil

Pada subbab ini akan dijelaskan terkait validasi / pembuktian kebenaran dari metode-metode utama pada penelitian ini. Metode-metode tersebut adalah GMM, *Contour Tracing*, dan Kalman Filter. Adapun pembuktian metode-metode tersebut dapat dijabarkan sebagai berikut.

1. GMM dan Morfologi

Masukan dari metode ini dapat berupa citra RGB ataupun citra biner. Keluaran yang dihasilkan dari metode GMM adalah citra biner dengan piksel bernilai 0 sebagai latar belakang dan piksel bernilai 1 sebagai latar depan (objek). Hasil dari metode ini dapat dikatakan baik apabila ekstraksi latar depan sudah sesuai dengan objek bergerak yang ingin dilacak, dalam kasus ini yaitu objek ikan. Sebaliknya, jika objek selain ikan ikut terekstrak sebagai bagian dari latar depan (latar belakang dinamis, gelembung air), maka sistem dianggap kurang baik.



Gambar 3.11: Contoh *groundtruth* GMM

Performa metode GMM dihitung dengan cara membandingkan hasil deteksi objek dengan *groundtruth* (Gambar 3.12). Sebanyak 3 (tiga) sampel *groundtruth* diambil dari masing-masing video. Empat buah variabel, A (*Accuracy*), P (*Precision*), R (*Recall*), $F1$ (*F1 Score*) dan digunakan untuk mengukur performa metode sebagai berikut:

$$A = \frac{TN + TP}{TN + FP + TP + FN} \times 100\% \quad (3.5)$$

$$P = \frac{TP}{TP + FP} \times 100\% \quad (3.6)$$

$$R = \frac{TP}{TP + FN} \times 100\% \quad (3.7)$$

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \times 100\% \quad (3.8)$$

dimana A merepresentasikan jumlah piksel yang dideteksi dari seluruh deteksi adalah benar latar depan, P (*Precision*) merepresentasikan seberapa besar persentase piksel yang diprediksi adalah benar latar depan, R (*Recall*) merepresentasikan seberapa besar persentase piksel latar depan yang berhasil dideteksi, $F1$ merupakan gabungan dari *Recall* dan *Precision*, TP (*True Positive*) adalah hasil deteksi piksel latar depan yang benar, TN (*True Negative*) adalah hasil deteksi piksel latar belakang yang benar, FP (*False*

Positive) adalah piksel latar belakang yang terdeteksi sebagai piksel latar depan, dan *FN* (*False Negative*) adalah piksel latar depan yang tidak terdeteksi oleh sistem (Chavan dan Gengaje, 2017). Performa metode GMM dapat dikatakan baik apabila variabel *P* dan *R* menghasilkan nilai persentase yang besar.

2. *Downsampling*

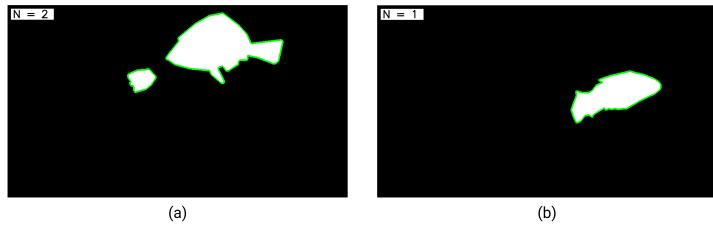
Metode *Downsampling* diperlukan untuk meningkatkan performa *tracing* pada metode CT dengan memperhitungkan juga akurasi sistem serta kualitas gambar. Tiga (3) buah sampel diambil dari masing-masing video. Evaluasi dihitung dengan cara membandingkan waktu eksekusi metode CT sebelum dan sesudah dilakukan *downsampling* pada *frame* terkait.

3. *Contour Tracing*

Metode CT mengambil masukan citra biner dari proses deteksi objek bergerak sebelumnya. Metode ini menghasilkan citra biner yang setiap objek bergerak sudah mempunyai tepi / konturnya masing-masing. Dari kontur tersebut dapat dihasilkan *bounding box*. Pembuktian dapat dilakukan dengan cara membandingkan jumlah objek / jumlah kontur hasil metode CT dengan *groundtruth*. Sebanyak 3 (tiga) buah sampel *groundtruth* diambil dari masing-masing video. Dua atau lebih objek yang mengalami proses *occlusion* dapat diklasifikasikan sebagai satu objek. *Absolute Error* digunakan dalam proses evaluasi metode CT.

$$AE = \left| \frac{Jumlah objek sebenarnya - Jumlah objek hasil CT}{Jumlah objek hasil CT} \right| \times 100\% \quad (3.9)$$

Nilai *Absolute Error* = 0% menyatakan bahwa jumlah objek sudah sesuai dengan *groundtruth*. Performa metode CT sangat bergantung pada hasil deteksi objek dari metode sebelumnya. Maka dari itu, jika nilai *error* cukup besar, perlu dilakukan penyesuaian parameter sistem untuk kemudian dilakukan pengujian ulang agar mendapatkan hasil yang baik.



Gambar 3.12: Contoh *groundtruth Contour Tracing*

4. Pelacakan Objek dengan Kalman Filter

Kalman Filter bertugas sebagai pemberi label/identitas untuk setiap objek yang mempunyai *bounding box*. Proses asosiasi data membuat Kalman Filter dapat mempertahankan label masing-masing objek tersebut selama video berlangsung. Inilah yang disebut sebagai proses "pelacakan objek". Keluaran dari proses ini adalah *running* video berisikan objek yang dilacak lengkap dengan *bounding box* serta label uniknya masing-masing.

Performa metode Kalman Filter dihitung dengan cara membandingkan hasil pelacakan oleh sistem dengan *groundtruth* berupa koordinat nilai tengah masing-masing objek. Sampel *groundtruth* diambil dari masing-masing video sebanyak 3 sampel. *Root Mean Squared Error (RMSE)* digunakan untuk mengukur performa nilai yang diprediksi oleh model (*predicted*) dengan nilai sebenarnya (*observed*). Formula *RMSE* dapat didefinisikan sebagai berikut:

$$TOTAL\ RMSE = \sqrt{(RMSE_X)^2 + (RMSE_Y)^2} \quad (3.10)$$

dimana

$$RMSE_X = \sqrt{\sum_{i=1}^n \frac{(\hat{x}_i - x_i)^2}{n}} \quad (3.11)$$

$$RMSE_Y = \sqrt{\sum_{i=1}^n \frac{(\hat{y}_i - y_i)^2}{n}} \quad (3.12)$$

Variabel \hat{x}_i , \hat{y}_i adalah nilai koordinat titik tengah (x, y) yang diprediksi,

sementara x_i, y_i adalah nilai koordinat titik tengah (x, y) yang diobservasi, dan n adalah jumlah observasi. Hasil pelacakan dapat dikatakan baik apabila nilai *TOTAL RMSE* mendekati 0.

3.3.3 Parameter Sistem

Dalam sistem pelacakan ini, terdapat beberapa parameter yang perlu diatur demi mendapatkan hasil pelacakan yang optimal. Pengujian *tuning* parameter dilakukan terhadap seluruh skenario pengujian masing-masing metode pada subbab sebelumnya.

1. *Downsampling Scale*

Besaran skala dalam proses *Downsampling* berfungsi sebagai penentu seberapa besar sistem akan melakukan *Downsampling* terhadap sebuah *frame* sebelum dilakukannya proses *Contour Tracing*. Besaran skala yang digunakan adalah $S = 2, 4, 8$. Dengan mempertimbangkan kualitas gambar yang dihasilkan dan juga akurasi metode, penentuan besaran skala ini diharapkan dapat meningkatkan performa sistem secara signifikan.

2. Ukuran *Structuring Element*

Terdapat dua *structuring element* yang digunakan pada operasi morfologi. Ukuran bawaan *structuring element* proses erosi adalah 5×5 . Semakin besar ukurannya maka semakin banyak *salt-pepper noise* yang hilang, akan tetapi semakin banyak pula piksel *foreground* yang terkikis. Sementara itu, ukuran bawaan *structuring element* proses dilasi adalah 7×7 . Semakin besar ukurannya, maka akan semakin baik sistem dalam menyambungkan bagian objek yang terputus ataupun berlubang, akan tetapi semakin besar juga kemungkinan dua buah objek yang seharusnya terpisah menjadi satu.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Input Video

Pada tahap pertama, sumber data berupa video dimasukkan kedalam sistem yang kemudian diubah menjadi sekumpulan citra (*frame*). Kemudian citra melalui proses *pre-processing*, yaitu konversi ruang warna RGB ke ruang warna *grayscale*.

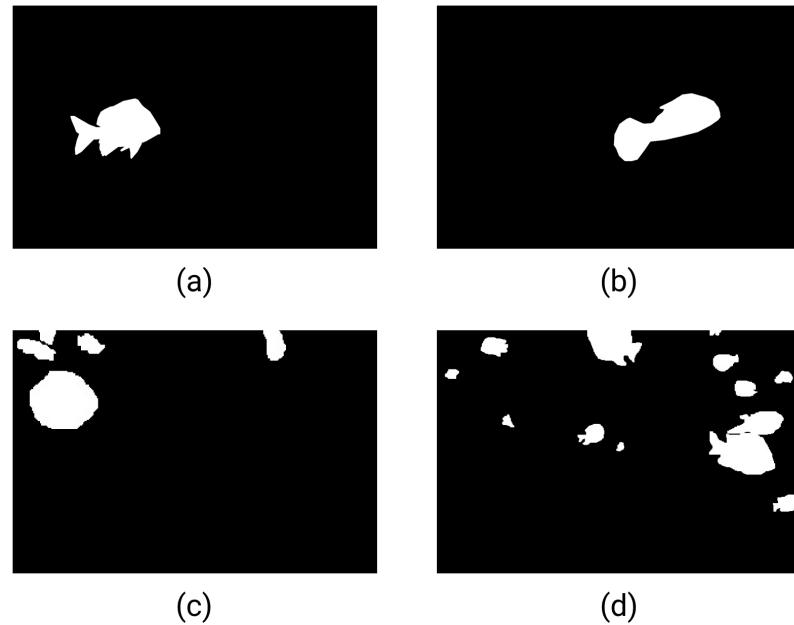


```
1 capture = cv2.VideoCapture(video_path)
2 while True:
3     ret, frame = capture.read()
4     if frame is None:
5         break
6     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
7     .
8     .
9     .
10
11 capture.release()
12 cv2.destroyAllWindows()
```

Gambar 4.1: Potongan sampel kode input sistem

4.2 GMM

Citra yang telah melalui proses *pre-processing* selanjutnya digunakan sebagai data masukan proses *background subtraction* menggunakan GMM. Keluaran dari metode ini kemudian dievaluasi terhadap *groundtruth* yang sudah dipersiapkan sebelumnya. Diperoleh skor *accuracy*, *recall*, *precision*, dan *F1* untuk masing-masing data uji.



Gambar 4.2: Sampel *groundtruth* data uji video indeks (a) 9908 (b) 9866 (c) gt_124 (d) gt_116

Tabel 4.1 memperlihatkan hasil dari dataset video kategori objek tunggal latar belakang sederhana. Untuk kategori video tersebut, dataset yang digunakan dapat dikatakan kurang optimal. Skor performa cenderung fluktuatif dikarenakan terlalu banyak *noise* yang dihasilkan, serta jumlah frame yang sedikit.

Table 4.1: Hasil ujicoba proses *background subtraction* menggunakan GMM terhadap video indeks 9908

Frame	Citra Asli	Hasil GMM	Groundtruth	<i>F1</i>
120				58

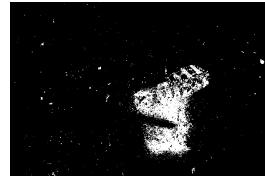
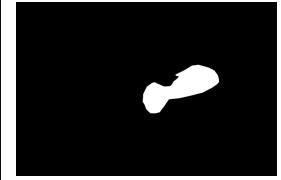
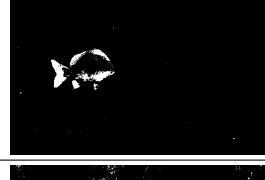
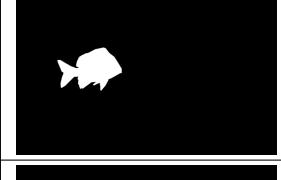
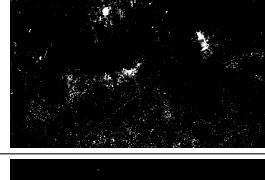
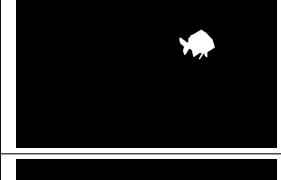
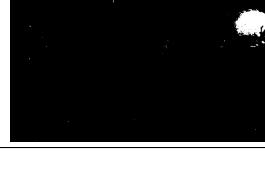
Frame	Citra Asli	Hasil GMM	Groundtruth	<i>F1</i>
230				31,8
290				58,4

Table 4.2: Hasil ujicoba proses *background subtraction* menggunakan GMM terhadap video indeks 9986

Frame	Citra Asli	Hasil GMM	Groundtruth	<i>F1</i>
509				41,7
789				8,4
849				44,8

Tabel 4.2 memperlihatkan hasil GMM pada video kategori kedua yaitu, objek tunggal latar belakang kompleks. Dalam pengujian, ketika terdapat *burst of dynamic background* dalam waktu yang singkat (*frame* 789), GMM tampak kesulitan

menghasilkan keluaran yang diharapkan, beberapa objek latar belakang yang bergerak *sewaktu-waktu* juga ikut terdeteksi, sehingga menghasilkan skor yang cukup buruk pada frame tersebut.

Table 4.3: Hasil uji coba proses *background subtraction* menggunakan GMM terhadap video indeks gt_124

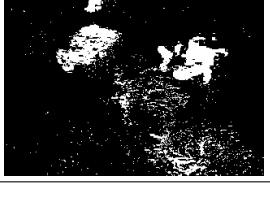
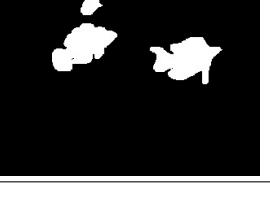
Frame	Citra Asli	Hasil GMM	Groundtruth	F1
705				68,5
1173				80
1191				67,1

Selanjutnya, GMM dapat menghasilkan keluaran yang cukup baik jika latar belakang video termasuk ke dalam kategori sederhana. Hal ini dapat terlihat pada Tabel 4.3 dimana skor tertinggi berada di angka 80%, dengan menghasilkan sedikit *noise* yang nantinya akan dieliminasi lebih baik lagi oleh Operasi Morfologi.

Berbeda dengan kasus video indeks gt_124, latar belakang pada video indeks gt_116 (Tabel 4.4) merupakan latar belakang yang *sangat* dinamis dan kompleks. Dari hasil pengujian, walaupun rata-rata skor yang dihasilkan adalah 58%, GMM tampak gagal dalam mengeskat objek ikan secara keseluruhan. Dikarenakan masih terdapat banyak *noise* latar belakang yang ikut terdeteksi sebagai objek bergerak. Hal

ini akan berpengaruh terhadap jumlah objek serta jumlah *track* pada metode-metode selanjutnya, yaitu *Contour Tracing* dan *Kalman Filter*.

Table 4.4: Hasil uji coba proses *background subtraction* menggunakan GMM terhadap video indeks gt_116

Frame	Citra Asli	Hasil GMM	Groundtruth	F1
803				43,5
859				55,2
1167				50,7



```

1 def detect(self, frame):
2     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
3
4     # Subtract bg using GMM
5     fg_mask = self.bg_subtractor.apply(gray)

```

Gambar 4.3: Potongan sampel kode Background Subtraction yang dibantu oleh GMM

Table 4.5: Skor *Accuracy*, *Recall*, *Precision*, dan *F1* pada metode *Background Subtraction* menggunakan GMM

Indeks	Frame	Accuracy	Recall	Precision	<i>F1 Score</i>
9908	120	96	50,3	68,6	58
9908	230	94,8	31,9	31,7	31,8
9908	290	97,2	46,9	77,4	58,4
9866	509	97,5	27	91,4	41,7
9866	789	97,6	9	7	8
9866	849	96,9	28,9	99,4	44,8
gt_124	705	96,5	60,3	79,2	68,5
gt_124	1173	97,4	79,9	80,1	80
gt_124	1191	95,4	59,1	77,7	67,1
gt_116	803	94,9	49,7	38,6	43,5
gt_116	859	94,1	55,8	54,9	55,2
gt_116	1167	93,6	49,3	52,3	50,7

Secara teori, GMM seharusnya dapat menghasilkan keluaran yang lebih baik seiring dengan bertambahnya *frame* yang diolah oleh sistem. Dikarenakan proses *training* model latar belakang oleh GMM dapat memanfaatkan jumlah dataset yang lebih banyak. Akan tetapi pada praktiknya, banyak juga faktor yang dapat membuat keluaran metode ini kurang baik. Hasil yang diharapkan adalah skor *Precision* dan *Recall* yang mendekati satu, dimana *Precision* merepresentasikan seberapa besar kasus positif (pixsel latar depan) yang benar dari seluruh prediksi positif, dan *Recall* merepresentasikan seberapa besar kasus positif yang benar dari seluruh kasus positif di dataset.

Jika dilihat pada Tabel 4.5, secara garis besar skor GMM cenderung fluktuatif, bahkan dalam beberapa kasus GMM tampak gagal dalam mengekstraksi objek ikan. Contohnya ada pada video indeks 9866 *frame* 789, *F1 Score* menyentuh angka yang sangat rendah, yaitu 8%. Hal ini dikarenakan metode mengalami lonjakan input latar belakang dinamis yang sangat besar pada *frame* tersebut. Artinya, model latar belakang yang *di-generate* oleh GMM gagal dalam mengklasifikasikan piksel *frame* ke dalam kelasnya masing-masing.

Hal lain yang dapat disimpulkan adalah pada video indeks 9908 dan gt_124, dimana kedua dataset tersebut merupakan dataset dengan kategori latar belakang sederhana, menghasilkan rata-rata skor yang lebih tinggi dari pada video indeks 9866 dan gt_116, yang merupakan video dengan kategori latar belakang kompleks. Hasil ini seperti yang diharapkan penulis, GMM akan menghasilkan keluaran yang lebih baik jika sedari awal *noise* yang ada pada masukkan sistem sudah sangat *minim*. Walaupun salah satu fungsi dari GMM itu sendiri adalah mengeliminasi *noise* yang berulang (seperti dahan pohon yang berayun dan iluminasi), akan tetapi jika *noise* itu sendiri sudah tidak ada pada video masukkan tentu hal ini akan berpengaruh terhadap meningkatnya skor yang dihasilkan.

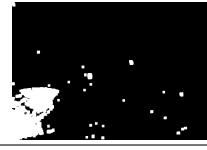
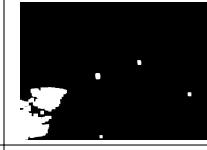
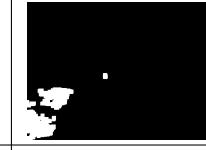
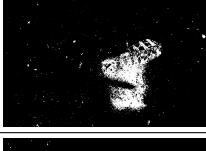
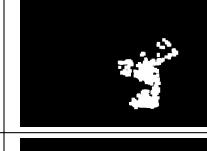
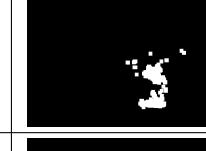
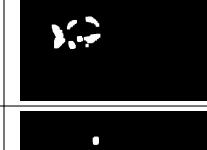
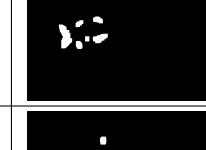
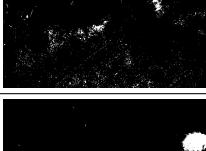
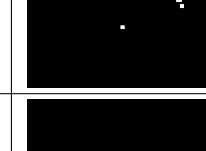
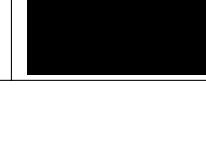
$$Precision = \frac{TP}{TP + FP} = \frac{N. \text{ of } Correctly \text{ Predicted Positive Instances}}{N. \text{ of } Total \text{ Positive Predictions We Made}} \quad (4.1)$$

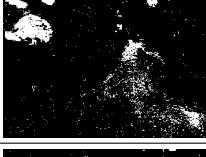
$$Recall = \frac{TP}{TP + FN} = \frac{N. \text{ of } Correctly \text{ Predicted Positive Instances}}{N. \text{ of } Total \text{ Positive Instances in the Dataset}} \quad (4.2)$$

4.3 Operasi Morfologi

Selanjutnya, citra melewati proses eliminasi *noise* menggunakan Operasi Morfologi. Dalam pengujian tahap ini, percobaan dilakukan dengan beberapa nilai *structuring element / kernel* Operasi Morfologi yang berbeda-beda.

Table 4.6: Hasil uji coba proses *background subtraction* menggunakan GMM yang disempurnakan oleh Operasi Morfologi

Id / Frame	Hasil GMM	Kernel		
		3×9	5×11	7×13
9908 / 120				
9908 / 230				
9908 / 290				
9866 / 509				
9866 / 789				
9866 / 849				

Id / Frame	Hasil GMM	Kernel		
		3×9	5×11	7×13
gt_124 / 705				
gt_124 / 1173				
gt_124 / 1191				
gt_116 / 803				
gt_116 / 859				
gt_116 / 1167				

Digunakan 3 parameter *structuring element / kernel* (Operasi Morfologi) untuk masing-masing sampel *frame*. Proses ini menyempurnakan keluaran metode GMM dengan menghapus *noise* (*noise removal*) secara lebih **agresif** dan menghasilkan citra biner yang diharapkan bersih dari *noise*. Fokus dari Operasi Morfologi adalah menghapus *noise* sebanyak-banyaknya. Jika dilihat pada Tabel 4.7, skor *metric* cenderung mengalami penurunan seiring dengan semakin besarnya ukuran *kernel* yang digunakan. Hal ini dikarenakan ukuran *kernel* yang lebih besar akan mengikis piksel lebih banyak dari pada ukuran *kernel* yang lebih kecil.

Penentuan besaran ukuran *kernel* perlu mempertimbangkan rata-rata ukuran objek yang ingin dilacak untuk meminimalisir piksel objek tereliminasi, atau bahkan dalam kasus terburuk objek tersebut hilang dari *frame*.

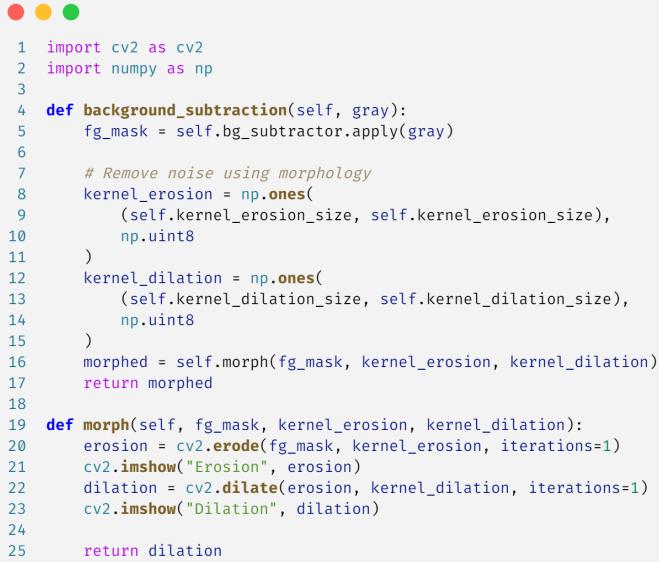
Table 4.7: Skor *Accuracy*, *Recall*, *Precision*, dan *F1* Operasi Morfologi

Indeks	Frame	<i>Kernel</i>	<i>Accuracy</i>	<i>Recall</i>	<i>Precision</i>	<i>F1</i>
9908	120	3 × 9	96,8	78,6	68,3	73,1
		5 × 11	97,1	68	76,1	71,8
		7 × 13	96,8	58	77,1	66,2
9908	230	3 × 9	94,5	64,1	36,9	46,9
		5 × 11	94,6	38,2	32,2	35
		7 × 13	94,4	12,2	16,6	14,1
9908	290	3 × 9	97,8	70,2	76,2	73,1
		5 × 11	97,7	57,7	88,4	69,8
		7 × 13	97,8	51,6	92,9	66,4
9866	509	3 × 9	97,7	44	76,8	55,9
		5 × 11	97,5	37,4	78,1	50,5
		7 × 13	97,4	32,7	78,5	46,2
9866	789	3 × 9	98,1	17,5	19	18,2
		5 × 11	98,6	13,7	32,9	19,4
		7 × 13	98,7	10,8	34,9	16,5
9866	849	3 × 9	97,3	39,8	99	56,8
		5 × 11	97,2	36,6	99,5	53,5
		7 × 13	97,1	34,1	99,5	50,9

Indeks	Frame	<i>Kernel</i>	<i>Accuracy</i>	<i>Recall</i>	<i>Precision</i>	<i>F1</i>
gt_124	705	3×9	96,8	86,3	69,8	77,2
		5×11	96,6	80,5	70,5	75,2
		7×13	96,5	74,4	71,5	72,9
gt_124	1173	3×9	96,5	94,6	66	77,8
		5×11	97,1	93	71	80
		7×13	97,3	89,2	74,8	81,4
gt_124	1191	3×9	96,3	90,9	70,5	79,4
		5×11	96,6	81,7	76,1	78,8
		7×13	96,2	71,4	77,74	74,3
gt_116	803	3×9	95	85,1	43,3	57,4
		5×11	96,1	76,8	50,7	61,1
		7×13	96,8	65,3	59,6	62,3
gt_116	859	3×9	95,2	89,7	58,1	70,5
		5×11	96,	72,6	69,3	70,9
		7×13	96,2	58,5	77,2	66,6
gt_116	1167	3×9	94,8	85,	57,6	68,7
		5×11	96,2	72,9	71,1	72
		7×13	96	58,5	76	66

Penentuan 2 besaran ukuran *kernel* yang digunakan, juga mempertimbangkan *trade-off* yang dihasilkan dari masing-masing parameter tersebut. Semakin besar ukuran *kernel*, semakin sedikit *noise* yang tampak, akan tetapi semakin banyak juga informasi yang hilang (piksel objek). Hal ini berdampak pada ukuran serta bentuk objek yang juga ikut menyusut. Sebaliknya, semakin kecil

ukuran *kernel* yang digunakan, maka semakin banyak informasi (piksel objek) yang dapat dipertahankan, namun semakin banyak juga *noise* yang tampak. Hasil skor keluaran Operasi Morfologi dapat dilihat pada Tabel 4.7.



```

1 import cv2 as cv2
2 import numpy as np
3
4 def background_subtraction(self, gray):
5     fg_mask = self.bg_subtractor.apply(gray)
6
7     # Remove noise using morphology
8     kernel_erosion = np.ones(
9         (self.kernel_erosion_size, self.kernel_erosion_size),
10        np.uint8
11    )
12    kernel_dilation = np.ones(
13        (self.kernel_dilation_size, self.kernel_dilation_size),
14        np.uint8
15    )
16    morphed = self.morph(fg_mask, kernel_erosion, kernel_dilation)
17    return morphed
18
19 def morph(self, fg_mask, kernel_erosion, kernel_dilation):
20     erosion = cv2.erode(fg_mask, kernel_erosion, iterations=1)
21     cv2.imshow("Erosion", erosion)
22     dilation = cv2.dilate(erosion, kernel_dilation, iterations=1)
23     cv2.imshow("Dilation", dilation)
24
25     return dilation

```

Gambar 4.4: Potongan sampel kode *Background Subtraction* dan Operasi Morfologi

Dari ujicoba di atas, dapat disimpulkan bahwa secara visual metode ini melaksanakan tugasnya dengan baik, yaitu mengeliminasi *noise* dengan lebih agresif. Pada setiap *frame* uji Tabel 4.6, *salt and peper noise* dapat dieliminasi dengan baik untuk semua ukuran *structuring element* yang digunakan. Poin penting lain yang dapat diobservasi adalah skor metode yang tampak menurun seiring dengan besarnya ukuran *structuring element* yang digunakan (Tabel 4.7). Hal ini masih masuk ke dalam ekspektasi penulis dikarenakan penjelasan dari metode itu sendiri, dimana *trade-off* yang dihasilkan dari semakin besarnya ukuran *structuring element* yang digunakan, adalah semakin banyak piksel latar depan yang hilang, akan tetapi semakin banyak juga *noise* yang tereliminasi.

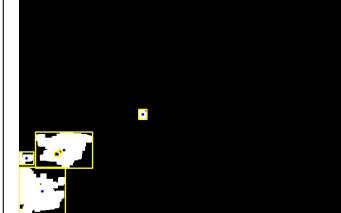
Metode ini hanya berfokus terhadap seberapa banyak *noise* yang dapat

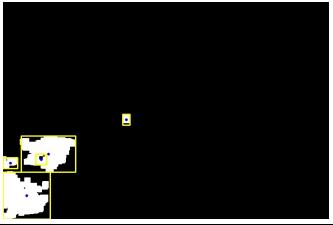
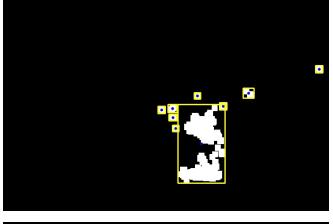
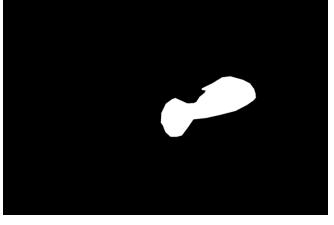
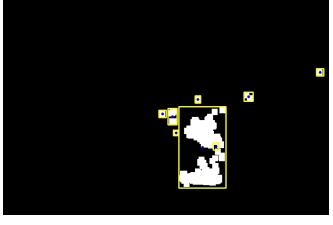
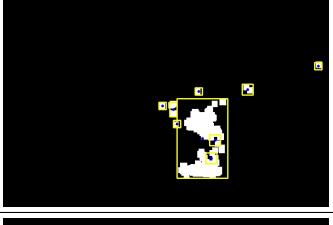
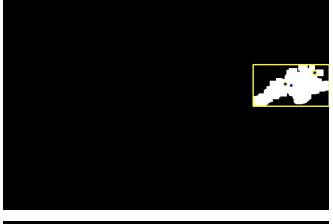
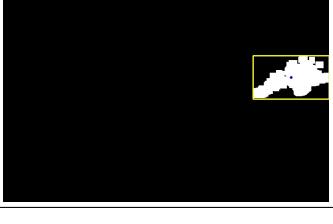
dieliminasi dengan tetap mempertahankan bentuk objek sebaik mungkin. Maka dari itu, skor *metric* yang semakin menurun bukan semata-mata pertanda bahwa performa sistem buruk, semua bergantung pada hasil metode selanjutnya, yaitu *Contour Tracing* dan *Kalman Filter* yang akan memvalidasi jumlah objek hasil deteksi, agar *noise* yang dihasilkan GMM tidak ikut terhitung oleh metode tersebut.

4.4 Contour Tracing dan Downsampling

Pada uji coba CT, setiap *frame* data diuji dengan skala *Downsampling* yang berbeda-beda. Hal ini akan berpengaruh terhadap performa metode CT. Semakin besar skala yang digunakan, maka semakin cepat performa yang didapat, begitu juga sebaliknya. Pemilihan besaran skala *Downsampling* juga tetap memperhatikan hasil *bounding box* dari metode CT tersebut.

Table 4.8: Hasil uji coba metode CT yang ditingkatkan oleh *Downsampling* pada video indeks 9908 dengan ukuran *kernel* Operasi Morfologi 7x13

Indeks	Frame	Groundtruth	Skala	Hasil
9908	120		$\times 2$	
			$\times 4$	

Indeks	Frame	<i>Groundtruth</i>	Skala	Hasil
			$\times 8$	
			$\times 2$	
9908	230		$\times 4$	
			$\times 8$	
9908	290		$\times 2$	
			$\times 4$	

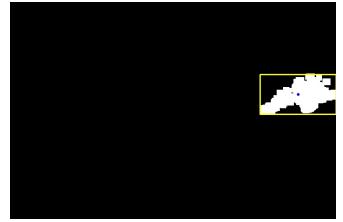
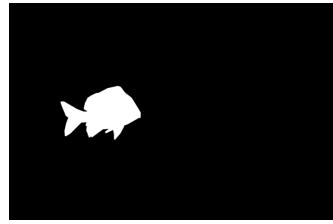
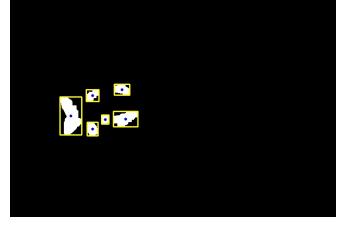
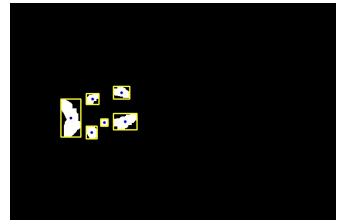
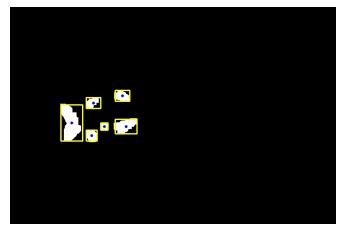
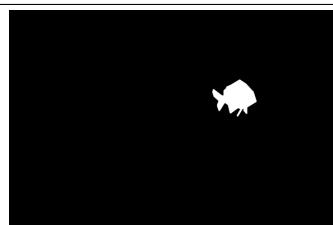
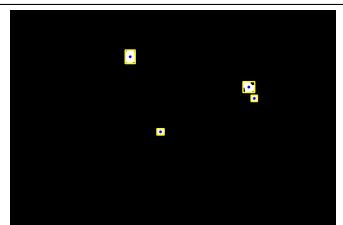
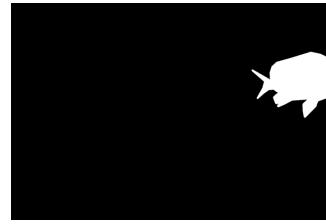
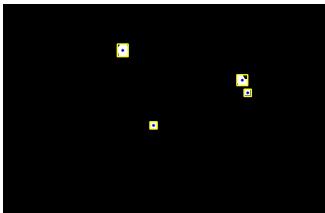
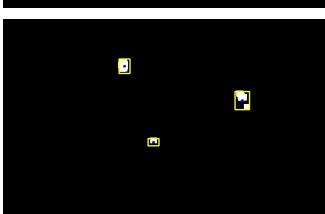
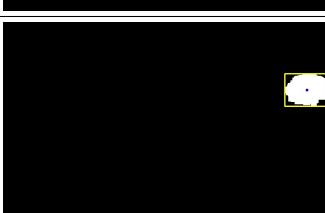
Indeks	Frame	<i>Groundtruth</i>	Skala	Hasil
			$\times 8$	

Table 4.9: Hasil uji coba metode CT yang ditingkatkan oleh *Downsampling* pada video indeks 9866 dengan ukuran *Kernel* Operasi Morfologi 7x13

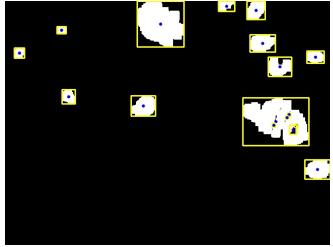
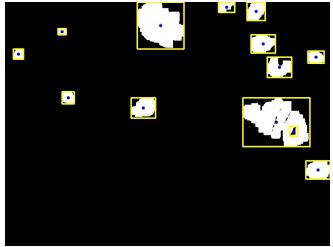
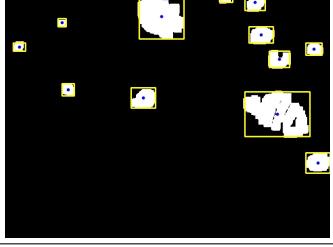
Indeks	Frame	<i>Groundtruth</i>	Skala	Hasil
9866	509		$\times 2$	
				
				
9866	789		$\times 2$	

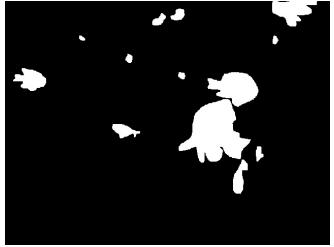
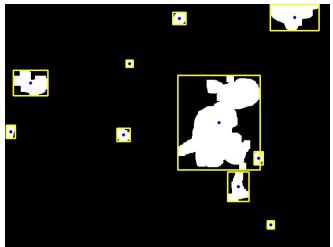
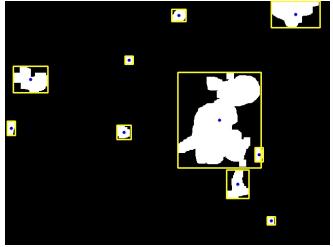
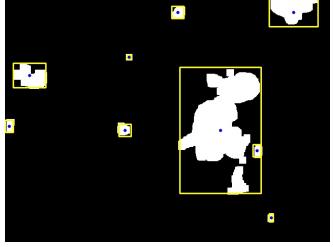
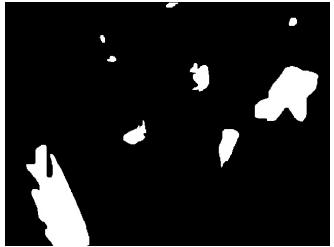
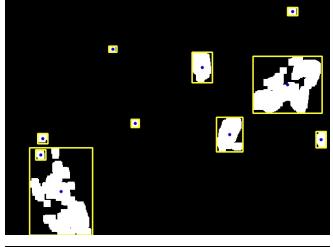
Indeks	Frame	<i>Groundtruth</i>	Skala	Hasil
9866	849		$\times 4$	
			$\times 8$	
			$\times 2$	
			$\times 4$	
			$\times 8$	

Pada Tabel 4.8 dan Tabel 4.9, setiap dataset uji (*frame*) dibandingkan terhadap *groundtruth*-nya masing-masing. Dapat diperhatikan bahwa tidak terdapat perbedaan *bounding box* yang cukup signifikan untuk setiap skala yang digunakan. *Bounding box* hanya tampak sedikit bergeser seiring dengan membesarnya skala *Downsampling* tersebut. Maka dari itu, penggunaan skala yang besar sangat setimpal dengan performa yang dihasilkan, dikarenakan semakin besar skala yang digunakan, akan semakin cepat juga metode CT berjalan.

Hal yang sama juga akan ditemukan pada Tabel 4.10 dan Tabel 4.11. Pengujian metode CT berfokus pada kecepatan serta keberhasilan metode dalam melacak *contour* masing-masing objek. Terkait jumlah objek yang tampak, sepenuhnya bergantung kepada metode sebelumnya, yaitu *background subtraction* menggunakan GMM yang disempurnakan oleh Operasi Morfologi. Walaupun dalam praktiknya, selisih jumlah objek juga dapat terjadi akibat penggunaan skala *downsampling* yang berbeda-beda (lihat Tabel??).

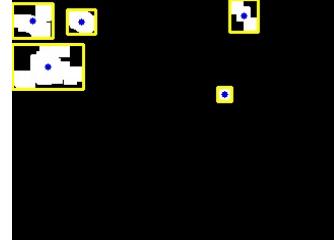
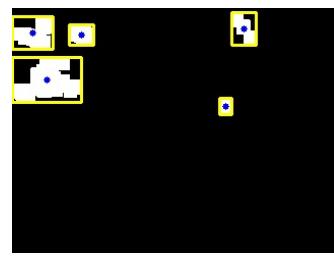
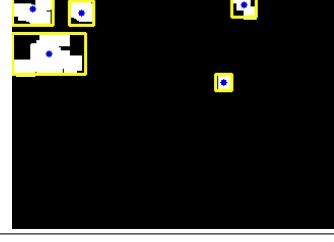
Table 4.10: Hasil uji coba metode CT yang ditingkatkan oleh *Downsampling* pada video indeks gt_124 dengan ukuran *kernel* Operasi Morfologi 7x13

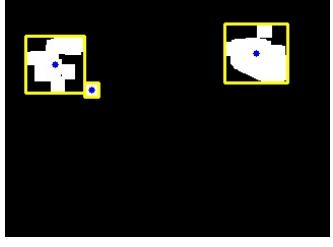
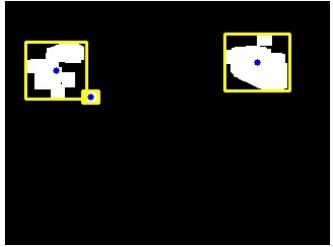
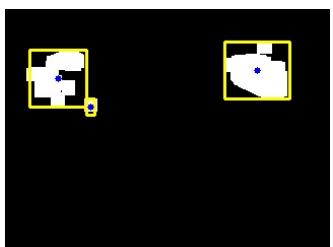
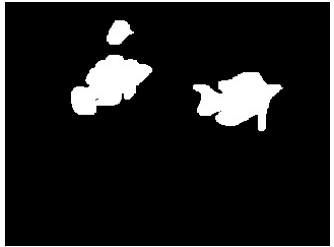
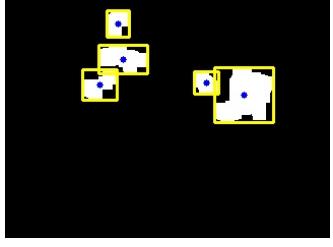
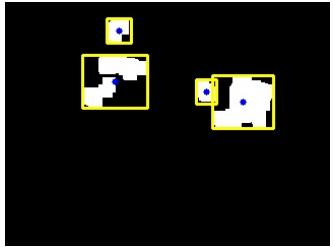
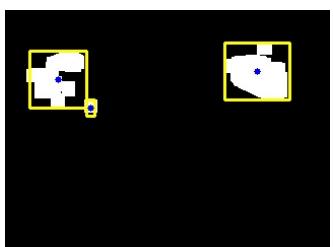
Indeks	Frame	Groundtruth	Skala	Hasil
gt_124	705		×2	
			×4	
			×8	

Indeks	Frame	<i>Groundtruth</i>	Skala	Hasil
gt_124	1173		$\times 2$	
				
				
gt_124	1191		$\times 2$	
				
				

Performa metode CT akan benar-benar diuji ketika citra masukan berasal dari video kategori ke-3 dan ke-4, yaitu video dengan objek lebih dari satu. Proses *contour tracing* yang penulis implementasi memiliki kompleksitas yang cukup tinggi. Semakin banyak objek yang perlu di-*tracing* dalam setiap *frame*, semakin besar juga waktu yang dibutuhkan sistem untuk menyelesaikan hal tersebut. Maka dari itu, diperlukanlah metode *Downsampling* yang dapat mempercepat performa metode CT tanpa merusak citra yang sudah ada. Digunakan skala $\times 2$, $\times 4$, dan $\times 8$ terhadap masing-masing citra keluaran metode sebelumnya.

Table 4.11: Hasil uji coba performa metode CT yang ditingkatkan oleh *Downsampling* pada video indeks gt_116 dengan ukuran *kernel* Operasi Morfologi 7x13

Indeks	Frame	Groundtruth	Skala	Hasil
gt_116	803		$\times 2$	
			$\times 4$	
			$\times 8$	

Indeks	Frame	<i>Groundtruth</i>	Skala	Hasil
gt_116	859		$\times 2$	
				
				
gt_116	1167		$\times 2$	
				
				

Tabel 4.12 dan Tabel 4.13 memperlihatkan evaluasi performa metode CT terhadap ukuran *kernel* morfologi dan skala *Downsampling* yang berbeda-beda. Kolom "Waktu" merepresentasikan waktu eksekusi metode CT dalam detik, kolom "Jumlah" merepresentasikan jumlah tepi objek yang berhasil diekstrak, kolom "Jumlah Sebenarnya" merepresentasikan jumlah objek sebenarnya dari citra *groundtruth*, sementara kolom "*Selisih*" merepresentasikan selisih nilai antara kalkulasi jumlah hasil CT dengan jumlah sebenarnya.

Table 4.12: Evaluasi hasil dan performa metode CT video kategori objek tunggal terhadap ukuran *kernel* Operasi Morofologi serta skala *Downsampling* yang berbeda-beda

Indeks	Frame	<i>Kernel</i>	Skala	Waktu	Jumlah	Jumlah Sebenarnya	<i>Selisih</i>
9908	120	3×9	$\times 2$	0,28	36	1	35
			$\times 4$	0,07	34	1	33
			$\times 8$	0,01	30	1	29
		5×11	$\times 2$	0,26	6	1	5
			$\times 4$	0,07	6	1	5
			$\times 8$	0,02	6	1	5
		7×13	$\times 2$	0,39	6	1	5
			$\times 4$	0,07	5	1	4
			$\times 8$	0,01	5	1	4
9908	230	3×9	$\times 2$	0,22	25	1	24
			$\times 4$	0,06	16	1	15
			$\times 8$	0,01	14	1	13

Indeks	Frame	<i>Kernel</i>	Skala	Waktu	Jumlah	Jumlah Sebenarnya	Selisih
9908	290	5×11	$\times 2$	0,24	10	1	9
			$\times 4$	0,1	10	1	9
			$\times 8$	0,02	7	1	6
		7×13	$\times 2$	0,28	9	1	8
			$\times 4$	0,08	8	1	7
			$\times 8$	0,02	9	1	8
		3×9	$\times 2$	0,23	22	1	21
			$\times 4$	0,05	21	1	20
			$\times 8$	0,01	15	1	14
		5×11	$\times 2$	0,39	7	1	6
			$\times 4$	0,07	5	1	4
			$\times 8$	0,07	4	1	3
9866	509	7×13	$\times 2$	0,35	3	1	2
			$\times 4$	0,06	1	1	0
			$\times 8$	0,01	1	1	0
		3×9	$\times 2$	0,29	4	1	3
			$\times 4$	0,07	4	1	3
			$\times 8$	0,01	3	1	2
		5×11	$\times 2$	0,26	6	1	5
			$\times 4$	0,06	6	1	5
			$\times 8$	0,01	5	1	4

Indeks	Frame	<i>Kernel</i>	Skala	Waktu	Jumlah	Jumlah Sebenarnya	Selisih
9866	789	7×13	$\times 2$	0,19	6	1	5
			$\times 4$	0,05	6	1	5
			$\times 8$	0,01	6	1	5
		3×9	$\times 2$	0,27	18	1	17
			$\times 4$	0,07	18	1	17
			$\times 8$	0,02	18	1	17
		5×11	$\times 2$	0,22	5	1	4
			$\times 4$	0,05	5	1	4
			$\times 8$	0,01	3	1	2
		7×13	$\times 2$	0,2	4	1	3
			$\times 4$	0,05	4	1	3
			$\times 8$	0,01	3	1	2
9866	849	3×9	$\times 2$	0,21	3	1	2
			$\times 4$	0,07	3	1	2
			$\times 8$	0,01	3	1	2
		5×11	$\times 2$	0,22	1	1	0
			$\times 4$	0,05	1	1	0
			$\times 8$	0,01	1	1	0
		7×13	$\times 2$	0,18	1	1	0
			$\times 4$	0,05	1	1	0
			$\times 8$	0,01	1	1	0

Jika diperhatikan, dari skala terkecil hingga skala terbesar terdapat peningkatan yang cukup drastis dari segi waktu eksekusi dengan rata-rata diatas 90%. Sementara itu, dari sisi jumlah objek terdapat selisih perbedaan yang cukup besar pada beberapa sampel uji. Selisih jumlah objek yang dihasilkan cenderung mengalami penurunan seiring dengan bertambahnya jumlah *frame*, membesarnya ukuran *Structuring Element*, serta membesarnya skala *Downsampling* yang digunakan.

Table 4.13: Evaluasi hasil dan performa metode CT video kategori objek lebih dari satu terhadap ukuran *kernel* Operasi Morofologi serta skala *Downsampling* yang berbeda-beda

Indeks	Frame	Kernel	Skala	Waktu	Jumlah	Jumlah Sebenarnya	Selisih
gt_124	705	3×9	$\times 2$	0,22	19	13	6
			$\times 4$	0,05	18	13	5
			$\times 8$	0,01	15	13	2
		5×11	$\times 2$	0,22	15	13	2
			$\times 4$	0,05	14	13	1
			$\times 8$	0,02	13	13	0
		7×13	$\times 2$	0,28	17	13	4
			$\times 4$	0,04	13	13	0
			$\times 8$	0,01	12	13	1
gt_124	1173	3×9	$\times 2$	0,22	23	13	10
			$\times 4$	0,06	21	13	8
			$\times 8$	0,01	22	13	9

Indeks	Frame	<i>Kernel</i>	Skala	Waktu	Jumlah	Jumlah Sebenarnya	Selisih
			×2	0,21	15	13	2
			×4	0,05	14	13	1
			×8	0,01	14	13	1
			×2	0,26	10	13	3
			×4	0,06	10	13	3
			×8	0,01	9	13	4
			×2	0,22	25	9	16
			×4	0,06	21	9	12
			×8	0,01	17	9	8
			×2	0,24	17	9	8
			×4	0,05	18	9	9
			×8	0,01	15	9	6
			×2	0,23	12	9	3
			×4	0,06	10	9	1
			×8	0,01	12	9	3
			×2	0,06	9	5	4
			×4	0,018	8	5	3
			×8	0,004	9	5	4
			×2	0,064	7	5	2
			×4	0,012	6	5	1
			×8	0,003	6	5	1

Indeks	Frame	<i>Kernel</i>	Skala	Waktu	Jumlah	Jumlah Sebenarnya	Selisih
gt_116	859	7×13	$\times 2$	0,049	5	5	0
			$\times 4$	0,012	5	5	0
			$\times 8$	0,003	5	5	0
		3×9	$\times 2$	0,055	14	3	11
			$\times 4$	0,015	11	3	8
			$\times 8$	0,004	11	3	8
		5×11	$\times 2$	0,066	10	3	7
			$\times 4$	0,011	8	3	5
			$\times 8$	0,044	7	3	4
		7×13	$\times 2$	0,051	3	3	0
			$\times 4$	0,013	3	3	0
			$\times 8$	0,003	3	3	0
gt_116	1167	3×9	$\times 2$	0,054	10	3	7
			$\times 4$	0,014	10	3	7
			$\times 8$	0,004	8	3	5
		5×11	$\times 2$	0,061	6	3	3
			$\times 4$	0,013	5	3	2
			$\times 8$	0,003	6	3	3
		7×13	$\times 2$	0,047	5	3	2
			$\times 4$	0,013	4	3	1
			$\times 8$	0,003	4	3	1

Video kategori objek tunggal (Tabel 4.12) secara garis besar meghasilkan keluaran yang buruk terkait *jumlah objek* hasil deteksi yang berhasil dihitung, terutama dengan parameter ukuran *structuring element* 3×9 dan skala $\times 2$. Hal ini dikarenakan dataset yang kurang optimal pada video kategori 1 (indeks 9908), serta gagalnya deteksi objek serta proses *noise removal* pada video kategori 2 (indeks 9866). Akan tetapi, *trend* yang dihasilkan adalah selisih objek cenderung menurun semakin besar ukuran *structuring element* dan skala yang digunakan, serta jumlah *frame* yang diproses oleh sistem.

Nilai selisih 0 dapat dijumpai pada penggunaan parameter *structuring element* 7×13 dan skala $\times 8$ serta *frame* uji terakhir pada masing-masing video kategori objek tunggal. Dari hasil observasi juga dapat dilihat bahwa penggunaan skala *Downsampling* yang berbeda-beda juga berdampak pada jumlah objek yang berhasil dihitung walaupun tidak begitu signifikan.

Sementara itu, *trend* yang sama akan dijumpai pada video kategori objek lebih dari satu. Untuk setiap sampel *frame* uji, selisih yang dihasilkan mengalami penurunan seiring dengan membesarnya ukuran *structuring element* serta skala yang digunakan. Nilai selisih 0 juga dapat dijumpai pada penggunaan parameter *structuring element* 7×13 dan skala $\times 8$ di beberapa sampel *frame* uji.



```

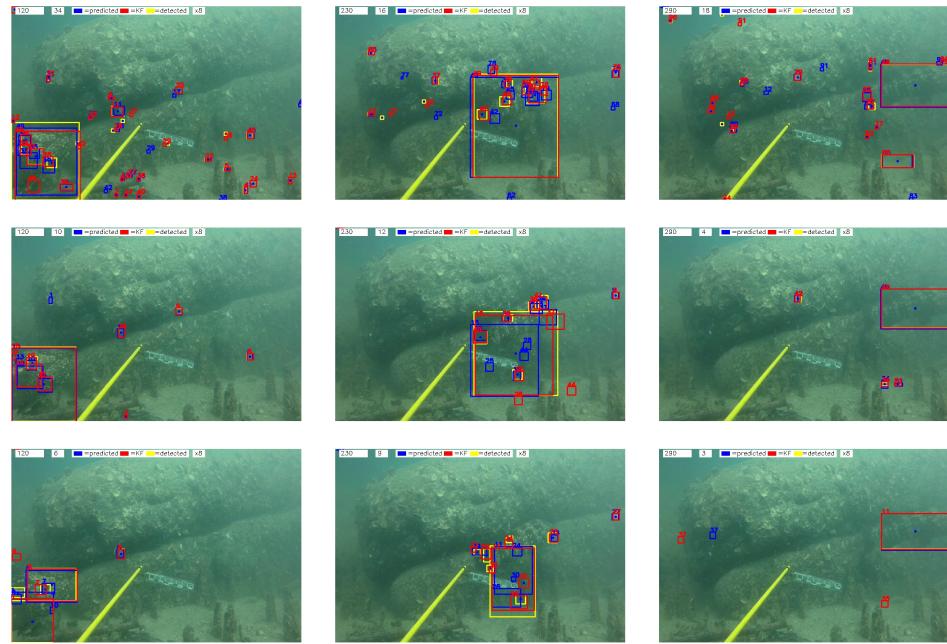
1 def raster_scan(self, img):
2     """
3         Take 2d binary image (bw) as an input. Raster scan will create an additional 0 padded
4         border around the original image. This will handle all true pixel (255) that sit at the edge
5         of the image
6
7     Parameters
8     -----
9     img : number[][][]
10    Binary image.
11
12    Returns
13    -----
14    List of contours
15    """
16    (ret, thresh2) = cv2.threshold(img, 127, 1, cv2.THRESH_BINARY | cv2.THRESH_OTSU)
17    padded_image = np.pad(thresh2, pad_width=[(1, 1), (1, 1)], mode="constant")
18    rows, cols = padded_image.shape
19    contours = []
20
21    LNBD = 0
22    for i in range(1, rows - 1):
23        for j in range(1, cols - 1):
24            # Check if pixel is a starting point or not
25            if padded_image[i][j] == 1 and padded_image[i][j - 1] == 0:
26                if LNBD == 4 or LNBD == 0:
27                    contour = self.border_following(padded_image, [i, j], [i, j - 1])
28                    contours.append(contour)
29
30            # Check LNBD
31            if padded_image[i][j] != 1:
32                LNBD = padded_image[i][j]
33            LNBD = 0
34
35    return np.array(contours)
36
37 def findCountourCustom(self, img):
38     return self.raster_scan(copy.copy(img))

```

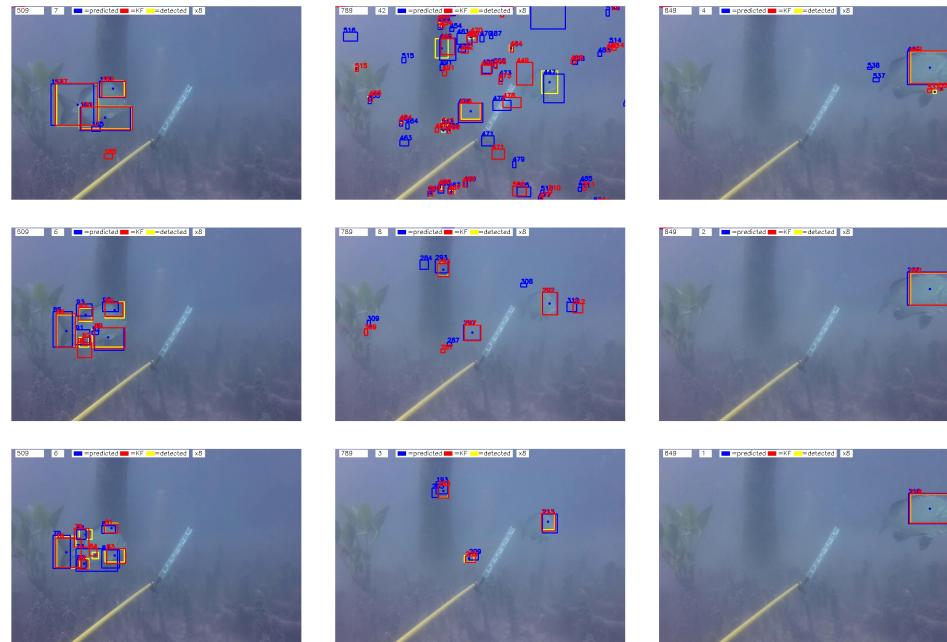
Gambar 4.5: Potongan sampel kode *Contour Tracing*

4.5 Kalman Filter

Kalman Filter adalah tahap uji yang paling penting. Keluaran dari metode ini adalah *frame* video yang masing-masing objeknya sudah terdapat tiga *bounding box* serta label objek. Ketiga *bounding box* tersebut dibedakan berdasarkan warna yaitu, biru untuk hasil prediski KF, merah untuk prediksi KF sebelumnya, dan kuning untuk deteksi sebenarnya. Untuk sampel hasil keluaran metode KF dapat dilihat pada gambar-gambar di bawah ini



Gambar 4.6: Sample *frame* hasil *tracking Kalman Filter* video indeks 9908

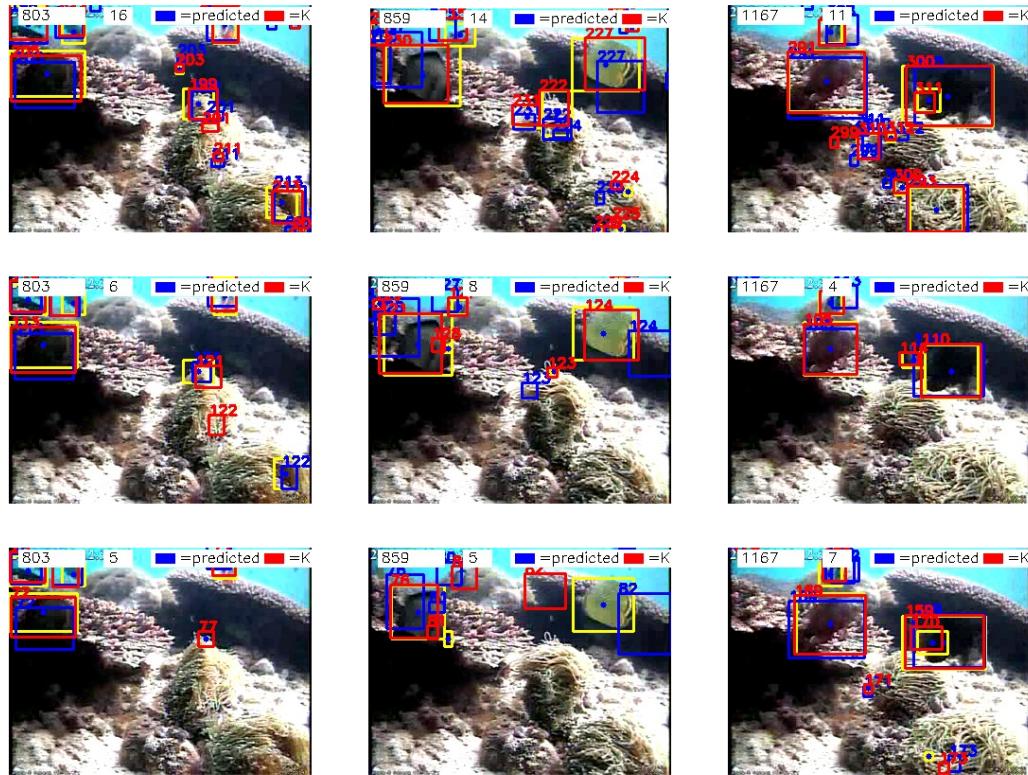


Gambar 4.7: Sample *frame* hasil *tracking Kalman Filter* video indeks 9866

Pada tahap ini, masing-masing video diuji performa KF-nya terhadap parameter *structuring element* serta skala *downsampling* yang berbeda-beda. Diambil data uji setiap 10 *frame* sekali selama video berlangsung. Kemudian dihitung RMSE koordinat piksel objek (x, y) dari seluruh data uji untuk masing-masing kategori pengujian. Secara garis besar, hasil yang diharapkan adalah nilai RMSE yang kecil dimana nilai 0 menandakan model yang akurat. Selain itu dihitung juga jumlah objek rata-rata seluruh data uji hasil dari prediksi KF.



Gambar 4.8: Sample *frame* hasil tracking *Kalman Filter* video indeks gt_124



Gambar 4.9: Sample frame hasil tracking Kalman Filter video indeks gt_116

Tabel 4.14 dan Tabel 4.15 menampilkan hasil evaluasi metode KF terhadap ke-empat dataset dengan parameter *kernel* Operasi Morfologi serta skala *Downsampling* yang berbeda-beda. Tabel 4.14 memperlihatkan skor *counting* metode KF, sementara Tabel 4.15 memperlihatkan skor *tracking*-nya. Variabel jumlah rata-rata didapatkan dari hasil menghitung rata-rata jumlah *track* setiap 10 *frame* sekali selama video berlangsung. Sementara variabel $RMSE_x$, $RMSE_y$, dan Total $RMSE$ dihitung berdasarkan nilai koordinat titik tengah x, y setiap 10 *frame* sekali selama video berlangsung.

Table 4.14: Evaluasi hasil dan performa *counting* metode KF terhadap ukuran *kernel* Operasi Morofologi serta skala *Downsampling* yang berbeda-beda

Indeks	Kernel	Skala	Rata-rata Jumlah	Rata-rata Jumlah Sebenarnya	Error
9908	3 × 9	×2	32,9	1	96,96
		×4	28,9		96,54
		×8	23		95,65
	5 × 11	×2	13,8		92,75
		×4	12,7		92,13
		×8	10,6		90,57
	7 × 13	×2	8		87,50
		×4	7,3		86,30
		×8	6,5		84,62
9866	3 × 9	×2	26,5	1	96,23
		×4	23,1		95,67
		×8	18,1		94,48
	5 × 11	×2	12,6		92,06
		×4	11,5		91,30
		×8	9,9		89,90
	7 × 13	×2	8,1		87,65
		×4	7,4		86,49
		×8	6,4		84,38

Indeks	<i>Kernel</i>	Skala	Rata-rata Jumlah	Rata-rata Jumlah Sebenarnya	<i>Error</i>
gt_124	3 × 9	×2	26,6	12,7	52,26
		×4	24,5		48,16
		×8	22,2		42,79
	5 × 11	×2	19,92		36,18
		×4	18,7		32,09
		×8	17,1		25,73
	7 × 13	×2	15,8		19,62
		×4	15,3		16,99
		×8	14,2		10,56
gt_116	3 × 9	×2	17,5	4,2	76
		×4	16,4		74,39
		×8	15,2		72,37
	5 × 11	×2	6		30
		×4	5,9		28,81
		×8	5,7		26,32
	7 × 13	×2	3,4		23,53
		×4	3,3		27,27
		×8	3,1		35,48

Pada video indeks 9908 (objek tunggal latar belakang sederhana), rata-rata *counting error* yang dihasilkan berada di angka yang sangat besar yaitu 91,44%, dengan *error* tertinggi mencapai 96,96% dan *error* terendah di angaka 84,62%. Angka yang tinggi tersebut didapatkan karena beberapa faktor. Pertama, tidak

optimalnya dataset yang digunakan untuk percobaan objek tunggal latar belakang sederhana. Latar belakang video indeks 9908 masih banyak mengandung *noise* bergerak yang dapat mengganggu proses deteksi objek seperti iluminasi cahaya, dan partikel-partikel kecil yang terbawa oleh air. Kedua, kumpulan *noise* tersebut pada akhirnya ikut terbaca sebagai objek bergerak oleh proses deteksi objek menggunakan GMM, dikarenakan GMM belum dapat membedakan objek bergerak ikan dengan objek bergerak lainnya. Kedua hal inilah yang paling berdampak pada buruknya akurasi yang dihasilkan.

Video indeks 9866 juga mengalami hal yang serupa seperti video indeks 9908. Pada percobaan video objek tunggal latar belakang kompleks didapatkan rata-rata *counting error* yang sangat besar yaitu sekitar 90,90%. Hal ini dikarekanan proses objek deteksi menggunakan GMM yang belum bisa membedakan objek bergerak ikan dengan objek bergerak lainnya pada video indeks 9866 yang sudah sangat *noisy*. *Trend* yang sama akan kita jumpai pada kedua indeks video di atas, yaitu *erorr* yang dihasilkan cenderung mengalami penurunan seiring dengan membesarnya ukuran *kernel* Operasi Morfologi yang digunakan. Hal ini dikarenakan ukuran *kernel* yang lebih besar dapat menghapus lebih banyak *noise* secara lebih agresif, yang pada akhirnya dapat mengurangi objek *noise* ikut terbaca oleh proses deteksi objek.

Pada indeks video selanjutnya, yaitu video indeks gt_124 yang masuk ke dalam kategori video objek lebih dari satu dan latar belakang sederhana, didapatkan rata-rata *counting error* sebesar 31,59%. *Error* tertinggi berada diangka 52,26% dan *error* terendah berada diangka 10,56%. *Error* yang kecil ini didapat karena video dataset yang digunakan sudah cukup optimal. Pada pengujian ini dapat kita lihat bagaimana video dengan *noise* yang sangat minim dapat menghasilkan keluaran yang lebih baik dari pada video dengan *noise* yang cukup banyak (9908, 9866), terutama dengan menggunakan parameter *kernel* Operasi Morfologi 7×13 dan Skala *Downsampling* $\times 8$.

Video indeks gt_116 adalah video kategori objek lebih dari satu dan latar belakang kompleks. Jika diperhatikan, rata-rata *counting error* yang dihasilkan lebih besar dibandingkan video indeks gt_124, yaitu sebesar 43,79%. Seperti pada pengujian tiga kategori video sebelumnya, video yang mengandung banyak *noise* tentu akan menghasilkan keluaran yang lebih buruk dari pada video dengan sedikit *noise*. Hal serupa pada pengujian video indeks 9908 dan 9866 juga terjadi pada pengujian video indeks gt_124 dan gt_116, semakin besar ukuran *kernel* Operasi Morfologi serta skala *Downsampling* yang digunakan, maka akan semakin banyak *noise* yang tereliminasi.

Table 4.15: Evaluasi hasil dan performa *tracking* metode KF terhadap ukuran *kernel* Operasi Morfologi serta skala *Downsampling* yang berbeda-beda

Indeks	Kernel	Skala	$RMSE_x$	$RMSE_y$	Total $RMSE$
9908	3×9	$\times 2$	12,4	6,9	14,2
		$\times 4$	13,3	7,3	15,2
		$\times 8$	16,4	9,2	18,8
	5×11	$\times 2$	12,8	6,4	14,3
		$\times 4$	8,7	5,9	10,5
		$\times 8$	11	6,6	12,8
	7×13	$\times 2$	19,4	9,8	21,8
		$\times 4$	14	12,2	18,6
		$\times 8$	10,8	10,6	15,1
9866	3×9	$\times 2$	12,8	8,8	15,5
		$\times 4$	12	9,5	15,3

Indeks	<i>Kernel</i>	Skala	<i>RMSE</i> _x	<i>RMSE</i> _y	Total <i>RMSE</i>
gt_124	5 × 11	×8	13,9	10,8	17,7
		×2	14,8	11	18,5
		×4	15	11,48	18,8
		×8	12,8	10,4	16,5
	7 × 13	×2	16,2	14	21,4
		×4	17,6	13,3	22,1
		×8	11,8	10,5	15,8
	3 × 9	×2	14,4	11,4	18,4
		×4	12,4	12,1	17,3
		×8	12,5	9,2	15,5
	5 × 11	×2	12,8	9,1	15,7
		×4	12,3	10,2	16
		×8	10,4	8,2	13,6
	7 × 13	×2	11,8	8,9	14,8
		×4	13,3	8,7	15,9
		×8	14,3	9,7	17,3
gt_116	3 × 9	×2	8,78	7,5	11,5
		×4	8	6,9	10,6
		×8	7,8	6,9	10,4
	5 × 11	×2	10,8	7,1	12,9
		×4	12,5	8,4	15,1
		×8	11,1	8,3	13,9

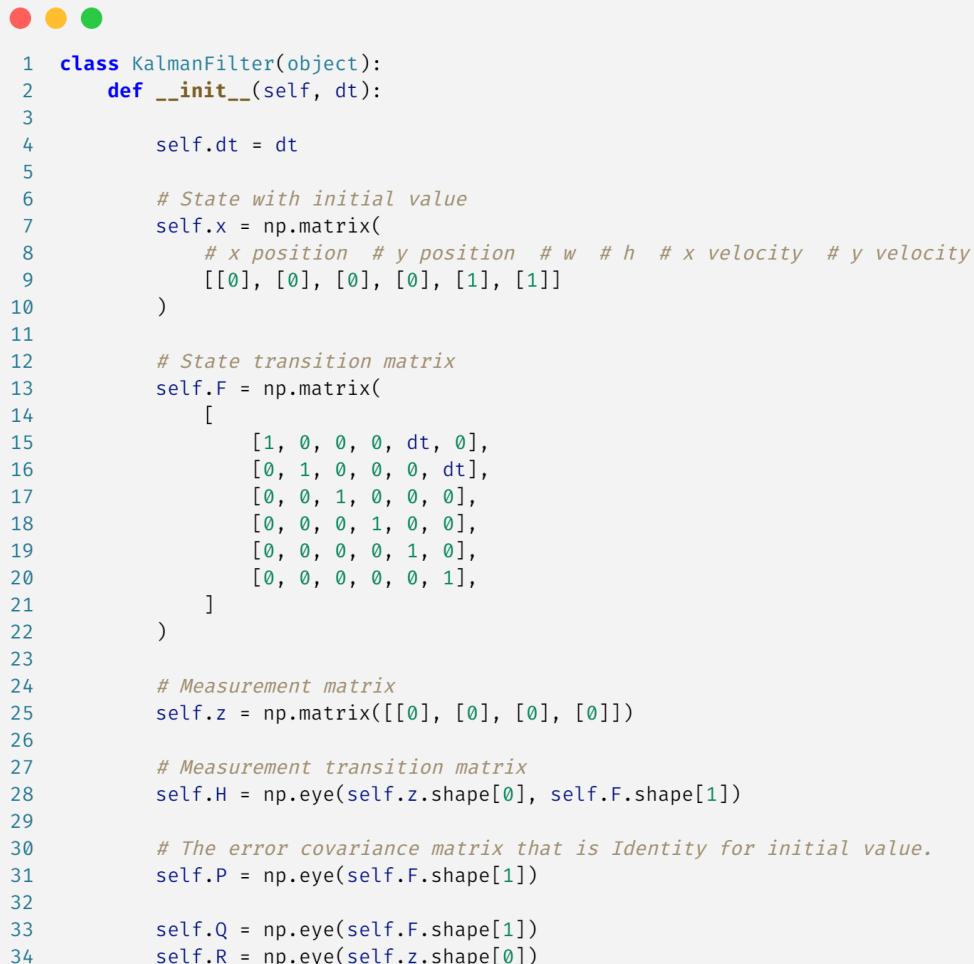
Indeks	<i>Kernel</i>	Skala	$RMSE_x$	$RMSE_y$	Total $RMSE$
	7×13	$\times 2$	9,5	7	11,8
		$\times 4$	10,6	8,1	13,4
		$\times 8$	9,7	7,2	12,1

Untuk video indeks 9908 dan 9866, rata-rata nilai $RMSE_x$ (koordinat titik tengah x) antara koordinat piksel sesungguhnya dan koordinat piksel yang diprediksi adalah 13,6. Nilai ini menunjukan hasil yang cukup baik mengingat nilai piksel maksimal koordinat x (lebar citra) adalah 720 piksel. Begitu juga dengan koordinat titik tengah y, didapat rata-rata nilai $RMSE_y$ sebesar 9,7, dimana nilai piksel maksimal koordinat y (panjang citra) adalah 480 piksel.

Video indeks gt_124 menghasilkan nilai rata-rata $RMSE_x = 12,6$ dan $RMSE_y = 10,7$, dimana nilai piksel maksimal koordinat x (lebar citra) adalah 640 piksel dan nilai piksel maksimal koordinat titik y (panjang citra) adalah 480 piksel. Kemudian dihasilkan juga rata-rata jumlah objek sebesar 19,3 objek. Pada video indeks gt_116, nilai rata-rata $RMSE_x$ dan $RMSE_y$ yang dihasilkan adalah 9,8 dan 7,4, dimana nilai piksel maksimal koordinat x (lebar citra) adalah 320 piksel dan nilai piksel maksimal koordinat y (panjang citra) adalah 240 piksel.

Dari hasil ujicoba diatas, dapat disimpulkan bahwa penggunaan parameter *structuring element* 7×13 dan serta skala *Downsampling* $\times 8$ menghasilkan keluaran rata-rata jumlah objek yang paling mendekati rata-rata jumlah sebenarnya. Pada video indeks 9908 dan 9866 ditemukan hal yang sama terkait skor performa seperti metode CT sebelumnya, dimana video kategori objek tunggal ini secara garis besar megnhasilkan keluaran yang cukup buruk terkait jumlah rata-rata objek, terutama dengan parameter ukuran *structuring element* 3×9 dan skala $\times 2$. Hal ini dikarenakan dataset yang kurang optimal, serta gagalnya deteksi objek serta proses *noise removal*.

Sementara itu dari segi pelacakan / (*tracking*) menggunakan *Kalman Filter*, sistem berhasil melakukan pelacakan dengan nilai *RMSE* yang cukup baik untuk masing-masing koordinat. Variabel *RMSE* dihitung dengan cara membandingkan koordinat titik tengah objek yang berhasil dideteksi oleh metode sebelumnya dengan koordinat titik tengah objek yang diprediksi oleh *Kalman Filter*. Nilai *RMSE* merepresentasikan seberapa jauh rata-rata *offside* kedua koordinat titik tengah tersebut. Semakin kecil nilainya, maka dapat dikatakan bahwa hasil prediksi oleh *Kalman Filter* semakin akurat.



```

1  class KalmanFilter(object):
2      def __init__(self, dt):
3          self.dt = dt
4
5          # State with initial value
6          self.x = np.matrix(
7              # x position # y position # w # h # x velocity # y velocity
8              [[0], [0], [0], [0], [1], [1]])
9
10
11         # State transition matrix
12         self.F = np.matrix(
13             [
14                 [1, 0, 0, 0, dt, 0],
15                 [0, 1, 0, 0, 0, dt],
16                 [0, 0, 1, 0, 0, 0],
17                 [0, 0, 0, 1, 0, 0],
18                 [0, 0, 0, 0, 1, 0],
19                 [0, 0, 0, 0, 0, 1],
20             ]
21         )
22
23
24         # Measurement matrix
25         self.z = np.matrix([[0], [0], [0], [0]])
26
27         # Measurement transition matrix
28         self.H = np.eye(self.z.shape[0], self.F.shape[1])
29
30         # The error covariance matrix that is Identity for initial value.
31         self.P = np.eye(self.F.shape[1])
32
33         self.Q = np.eye(self.F.shape[1])
34         self.R = np.eye(self.z.shape[0])

```

Gambar 4.10: Potongan sampel kode *Kalman Filter*

```
● ● ●  
1 def predict(self):  
2     # Predict state  
3     # self.x = np.dot(self.F, self.x) + np.dot(self.B, self.u)  
4     # print("old state\n", self.x)  
5     self.x = np.dot(self.F, self.x)  
6     # print("Predicted state\n", self.x)  
7  
8     # Predict state cov matrix  
9     # print("old cov matrix\n", self.P)  
10    self.P = np.dot(np.dot(self.F, self.P), self.F.T) + self.Q  
11    # print("Predicted cov matrix\n", self.P)  
12    return self.x  
13  
14  
15 def update(self, z):  
16     # Residual  
17     y = z - np.dot(self.H, self.x)  
18  
19     # Kalman Gain  
20     K = np.dot(np.dot(self.P, self.H.T), np.linalg.inv(np.dot(self.H, np.dot(self.P, self.H.T)) + self.R))  
21  
22     # Update state  
23     self.x = np.round(self.x + np.dot(K, y))  
24     # print("Updated state\n", self.x)  
25  
26     I = np.eye(self.H.shape[1])  
27  
28     # Update state cov matrix  
29     self.P = (I - K * self.H) * self.P  
30     # print("Updated cov matrix\n", self.P)  
31  
32     return self.x
```

Gambar 4.11: Potongan sampel kode *Kalman Filter*

BAB V

PENUTUP

5.1 Kesimpulan

Penelitian *Fish Movement Tracking* Menggunakan Metode GMM dan *Kalman Filter* secara garis besar dimulai dari proses *input* data ke dalam sistem. Data kemudian diubah kedalam bentuk citra (*frame*) yang dilanjutkan oleh proses *pre-processing*, yaitu mengkonversi ruang warna citra dari RGB *grayscale*. Lalu, citra dimasukkan sebagai *input* metode *background subtraction* dengan bantuan GMM untuk memodelkan latar belakang. Proses ekstraksi latar depan kemudian dilakukan terhadap model latar belakang yang sudah dihasilkan. Setelah itu, keluaran dari metode sebelumnya disempurnakan oleh Operasi Morfologi untuk mengeliminasi *noise* secara lebih agresif. Teknik *downsampling* kemudian dilakukan, proses ini akan meningkatkan performa metode selanjutnya yaitu *Contour Tracing* untuk menghasilkan tepi dari masing-masing objek yang berhasil dideteksi. Terakhir adalah proses prediksi dan juga *assigment* masing-masing objek prediksi dengan objek deteksi. Proses ini akan menghasilkan citra original yang seluruh objeknya sudah mempunyai *bounding box* dan labelnya masing-masing. Dari hasil metode KF inilah yang kemudian dapat menghasilkan rata-rata jumlah objek selama video berlangsung.

Dari hasil perancangan, implementasi, serta uji coba sistem *Fish Movement Tracking* Menggunakan Metode GMM dan *Kalman Filter* diperoleh kesimpulan sebagai berikut:

1. Sistem mampu melakukan pelacakan dengan lebih baik seiring dengan bertambahnya jumlah *frame* yang diproses.
2. Sistem mampu melakukan pelacakan dengan cukup baik menggunakan parameter *structuring element* ukuran 7×13 dan *downsampling* sebesar $x8$.

3. Pada dataset video indeks 9908 dan 9866 (objek tunggal), sistem pada akhirnya dapat menghasilkan rata-rata jumlah objek yang jauh dari rata-rata sebenarnya. Hal ini dikarenakan dataset yang digunakan kurang optimal pada video indeks 9908, serta gagalnya proses deteksi dan *noise removal* pada video indeks 9866.
4. Pada dataset video indeks gt_124 dan gt_116 (objek lebih dari satu), sistem pada akhirnya dapat menghasilkan rata-rata jumlah objek yang mendekati rata-rata sebenarnya.

5.2 Saran

Adapun beberapa saran yang dapat penulis sampaikan untuk penelitian selanjutnya adalah:

1. Penelitian ini belum dapat membedakan objek ikan dengan objek bergerak lainnya. Maka dari itu, perlu adanya pengembangan lanjutan yang menyempurnakan proses deteksi objek dengan metode yang lebih kompleks seperti *Neural-Network*, sehingga sistem dapat membedakan objek ikan dengan objek lainnya dengan baik.
2. Proses asosiasi data dapat ditingkatkan *cost function*-nya. Selain ukuran dan posisi, fungsi tersebut dapat ditambahkan variabel lain seperti warna.
3. Penelitian ini dapat digunakan untuk menguji objek selain ikan.

DAFTAR PUSTAKA

- Affandy, A. P. 2016. *Model Alat Penghitung Benih Ikan Air Tawar Ikan Mas (Fish Counter)*. Skripsi. Universitas Padjadjaran, Jatinagor.
- Al-Amri, C. F. 2020. *Rancang Bangun Fish Counter untuk Menghitung Bibit Ikan Lele*. Skripsi. Universitas Islam Indonesia, Yogyakarta.
- Benezeth, Y., Jodoin, P.-M., Emile, B., Laurent, H., dan Rosenberger, C. 2010. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*.
- Challa, S., Morelande, M. R., Musicki, D., dan Evans, R. J. 2011. *Fundamentals of Object Tracking*. Cambridge University Press, New York.
- Chavan, R. dan Gengaje, S. R. 2017. Multiple object detection using gmm technique and tracking using kalman filter. *International Journal of Computer Applications* (0975 – 8887).
- Desai, M. H. M. dan Gandhi, V. 2016. A survey: Background subtraction techniques. *International Journal of Scientific and Engineering Research*, 5.
- Devi, R. B. dan Singh, K. M. 2016. A survey on different background subtraction method for moving object detection. *International Journal for Research in Emerging Science and Technology*, 3:7–13.
- Friedman, N. dan Russell, S. 1997. Image segmentation in video sequences: A probabilistic approach. *Thirteenth Conference on Uncertainty in Artificial Intelligence*.
- Gonzalez, R. C. dan Woods, R. E. 2018. *Digital Image Processing*. Pearson Education, New York, 330 Hudson Street, fourth edition.
- Goodfellow, I., Bengio, Y., dan Courville, A. 2016. *Deep Learning*. MIT Press.
- Jeong, J.-M., Yoon, T.-S., dan Park, J.-B. 2014. Kalman filter based multiple objects detection-tracking algorithm robust to occlusion. *SICE Annual Conference*.
- Kalman, R. E. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*.

- Kavasidis, I., Palazzo, S., Salvo, R. D., Giordano, D., dan Spampinato, C. 2012. A semi-automatic tool for detection and tracking ground truth generation in videos. *Proceedings of the 1st International Workshop on Visual Interfaces for Ground Truth Collection in Computer Vision Applications*, pp. 6:1-6:5.
- Kavasidis, I., Palazzo, S., Salvo, R. D., Giordano, D., dan Spampinato, C. 2013. An innovative web-based collaborative platform for video annotation. *Multimedia Tools and Applications*, vol. 70, pp. 413-432.
- Kim, Y. dan Bang, H. 2018. *Introduction and Implementations of the Kalman Filter*. IntechOpen, Daejeon, Korea Advanced Institute of Science and Technology, Daejeon, South Korea.
- Kulchandani, J. S. dan Dangarwala, K. J. 2015. Moving object detection: Review of recent research trends. *International Conference on Pervasive Computing*.
- Li, X., Wang, K., Wang, W., dan Li, Y. 2010. A multiple object tracking method using kalman filter. *Proceedings of the 2010 IEEE International Conference on Information and Automation June 20 - 23, Harbin, China*.
- Manjula, S. dan Lakshmi, K. 2016. A study on object detection. *International Journal of Pharmacy and Technology*.
- Munir, R. 2004. *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Informatika, Bandung.
- Paul, C. dan Godambe, M. 2021. Image downsampling and upsampling. *Department of Electronics and Telecommunication*.
- Saleh, A., Laradji, I. H., Konovalov, D. A., Bradley, M., Vazquez, D., dan Sheaves, M. 2020. A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis. *Nature's Scientific Reports*.
- Saravanakumar, S., Vadivel, A., dan Ahmed, C. G. S. 2010. Multiple human object tracking using background subtraction and shadow removal techniques. *International Conference on Signal and Image Processing*.
- Smith, L. N. 2017. Cyclical learning rates for training neural networks. *IEEE Winter Conference on Applications of Computer Vision (WACV)*.

- Srisha, R. dan Khan, A. 2013. Morphological operations for image processing : Understanding and it applications. *National Conference on VLSI, Signal processing and Communications.*
- Statistik, B. P. 2016a. Jumlah rumah tangga perikanan budidaya menurut provinsi dan jenis budidaya. <https://www.bps.go.id/statictable/2013/12/31/1707/jumlah-rumah-tangga-perikanan-budidaya-menurut-provinsi-dan-jenis-budidaya-2000-2016.html>. Di akses pada, 21 September 2021.
- Statistik, B. P. 2016b. Konsumsi kalori dan protein penduduk indonesia dan provinsi. <https://www.bps.go.id/publication/2020/06/29/78ae644b0d37f4e1329d522f/konsumsi-kalori-dan-protein-penduduk-indonesia-dan-provinsi-september-2019.html>. Di akses pada, 21 September 2021.
- Statistik, B. P. 2016c. Produksi perikanan budidaya menurut provinsi dan jenis budidaya. <https://www.bps.go.id/statictable/2009/10/05/1706/produksi-perikanan-budidaya-menurut-provinsi-dan-jenis-budidaya-2000-2018.html>. Di akses pada, 21 September 2021.
- Stauffer, C. dan Grimson, W. E. L. 1999. Adaptive background mixture models for real-time tracking. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition.*
- Suzuki, S. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30:32–46.
- Wantimpres 2017. Potensi perikanan indonesia. <https://wantimpres.go.id/id/potensi-perikanan-indonesia/>. Di akses pada, 21 September 2021.
- Yilmaz, A., Javed, O., dan Shah, M. 2006. Object tracking: A survey. *ACM Computing Surveys*, 38.

LAMPIRAN

Lampiran 1 Kode Program

Semua *source code program* dapat diakses melalui link berikut: [github/cah-kangkung/object-tracking](https://github.com/cah-kangkung/object-tracking).

DAFTAR RIWAYAT HIDUP