DETEKSI *SHUTTLECOCK* MENGGUNAKAN ALGORITMA ORB (*ORIENTED FAST AND ROTATED BRIEF*)

SKRIPSI

Oleh :
ARVIN AZARIA MUNSYI
NIM. 210605110043

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

DETEKSI *SHUTTLECOCK* MENGGUNAKAN ALGORITMA ORB (*ORIENTED FAST AND ROTATED BRIEF)*

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
ARVIN AZARIA MUNSYI
NIM. 210605110043

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025

HALAMAN PERSETUJUAN

DETEKSI *SHUTTLECOCK* MENGGUNAKAN ALGORITMA ORB (*ORIENTED FAST AND ROTATED BRIEF*)

SKRIPSI

Oleh :
ARVIN AZARIA MUNSYI
NIM. 210605110043

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 17 November 2025

Pembimbing I,	Pembimbing II,
Johan Ericka Wahyu Prakasa, M.Kom	Shoffin Nahwa Utama, M.T
NIP. 19831213 201903 1 004	NIP. 19860703 202012 1 003

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Supriyono, M. Kom
NIP. 19841010 201903 1 012

HALAMAN PENGESAHAN

DETEKSI *SHUTTLECOCK* MENGGUNAKAN ALGORITMA ORB (*ORIENTED FAST AND ROTATED BRIEF)*

SKRIPSI

Oleh :
ARVIN AZARIA MUNSYI
NIM. 210605110043

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 03 Desember 2025

Susunan Dewan Penguji

Ketua Penguji	: Dr. M. Amin Hariyadi, M.T NIP. 19670118 200501 1 001	( )
Anggota Penguji I	: Ajib Hanani, M.T NIP. 19840731 202321 1 013	( )
Anggota Penguji II	: Johan Ericka Wahyu P, M. Kom NIP. 19831213 201903 1 004	( )
Anggota Penguji III	: Shoffin Nahwa Utama, M.T NIP. 19860703 202012 1 003	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Suprivono, M. Kom
NIP. 19841010 201903 1 012

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Arvin Azaria Munsyi
NIM	: 210605110043
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Deteksi *Shuttlecock* Menggunakan Algoritma ORB *(ORIENTED FAST AND ROTATED BRIEF)*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 03 Desember 2025
Yang membuat pernyataan,

Arvin Azaria Munsyi
NIM.210605110043

MOTTO

... dunia tidak pernah menunggu; kitalah yang belajar mengejar dengan makna…

HALAMAN PERSEMBAHAN

Dengan penuh rasa syukur, penulis mempersembahkan skripsi yang berjudul “Deteksi Shuttlecock Menggunakan Algoritma ORB (Oriented FAST and Rotated BRIEF)” sebagai bentuk ikhtiar dan tanggung jawab akademik yang dapat diselesaikan atas izin dan kehendak Allah Subhanahu wa Ta’ala. Shalawat serta salam senantiasa tercurah kepada Nabi Muhammad Shallallahu ‘alaihi wasallam, yang ajarannya menjadi pedoman dalam menuntut ilmu dan menapaki proses kehidupan.

Karya ini penulis persembahkan kepada kedua orang tua dan saudara-saudara penulis, yang dengan kesabaran, doa, dan kepercayaan telah menjadi sumber kekuatan utama dalam setiap langkah penulis. Persembahan ini juga ditujukan kepada keluarga, para pendidik, serta rekan seperjuangan yang telah menghadirkan dukungan, pemikiran, dan kebersamaan bermakna selama perjalanan akademik, sehingga proses penyusunan skripsi ini dapat dilalui dengan ketekunan dan kesungguhan. Tanpa doa, bantuan, dan dorongan dari mereka, pencapaian ini tidak akan mungkin terwujud.

KATA PENGANTAR

Assalamualaikum Warahmatullahi Wabarakatuh.

Segala puji syukur penulis panjatkan ke hadirat Allah Subhanahu wa Ta'ala atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan penulisan skripsi yang berjudul "Deteksi *Shuttlecock* Menggunakan Algoritma ORB *(ORIENTED FAST AND ROTATED BRIEF)*" dengan baik.

Penulis juga menyampaikan rasa terima kasih sebesar-besarnya kepada semua pihak yang telah memberikan dukungan, bantuan, serta motivasi selama proses penyusunan skripsi ini. Ucapan terima kasih ini secara khusus penulis tujukan kepada:

1. Prof. Dr. Hj. Ilfi Nur Diana, M.Si., CAHRM., CRMP selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim.
2. Dr. Agus Mulyono, M.Kes., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim
3. Supriyono, M.Kom, selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Johan Ericka Wahyu Prakasa, M.Kom**.** selaku Dosen Pembimbing I dan Shoffin Nahwa Utama, M.T. selaku Dosen Pembimbing II atas bimbingan, arahan, dan masukan yang sangat berarti selama penyusunan skripsi.
5. Dr. M. Amin Hariyadi, M.T. selaku Dosen Penguji I serta Ajib Hanani, M.T. selaku Dosen Penguji II atas masukan dan evaluasi yang membangun hingga skripsi ini dapat diselesaikan.

6. Ayah, Abd. Kadir Munsyi, dan mama, Noer Riskiyah, yang dengan doa, kesabaran, dan pengorbanan tulus menjadi sumber kekuatan penulis serta senantiasa memberikan dukungan hingga skripsi ini dapat diselesaikan.
7. Saudara penulis, Yauri Yusuf Munsyi dan Raisya Raditya Munsyi, atas dukungan, perhatian, dan kebersamaan yang menguatkan penulis selama menjalani proses perkuliahan hingga penyusunan skripsi ini.
8. Nabila Nur Wardani "Caynie", yang dengan ketulusan, kesabaran, dan keikhlasan selalu memberikan dukungan, semangat, serta menjadi tempat berbagi bagi penulis selama proses penyusunan skripsi ini.
9. Keluarga kedua penulis, teman kost "Samawa", atas kebersamaan, rasa kekeluargaan, dan saling menguatkan yang membantu penulis menjalani masa perantauan dan penyusunan skripsi ini.
10. Teman-teman Teknik Informatika Angkatan 21 "Aster", atas kebersamaan, semangat, dan kenangan berharga selama di tanah rantau.
11. Seluruh pihak yang tidak dapat penulis sebutkan satu per satu, yang telah memberikan dukungan secara langsung maupun tidak langsung.

Wassalamu'alaikum warahmatullahi wabarakatuh.

Malang, 03 Desember 2025

Arvin Azaria Munsyi

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ABSTRAK

Munsyi, Arvin Azaria. 2025. **Deteksi *Shuttlecock* Menggunakan Algoritma ORB (*ORIENTED FAST AND ROTATED BRIEF*).** Skripsi. Jurusan Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Johan Ericka Wahyu Prakasa, M.Kom (II) Shoffin Nahwa Utama, M.T.

Kata kunci: FLANN, KCF *Tracker*, ORB, Pengolahan Citra Digital, *Shuttlecock*.

Perkembangan teknologi pengolahan citra digital telah memungkinkan penerapan sistem deteksi objek secara otomatis pada berbagai bidang, termasuk olahraga. Salah satu cabang olahraga yang membutuhkan analisis berbasis citra adalah bulu tangkis, di mana deteksi *shuttlecock* menjadi aspek penting dalam proses evaluasi dan pelatihan. Penelitian ini bertujuan untuk mendeteksi obyek *shuttlecock* di antara benda-benda lain yang ada di lapangan menggunakan algoritma ORB (*Oriented FAST and Rotated BRIEF*). Proses penelitian mencakup tahap konversi citra, ekstraksi dan deskripsi fitur menggunakan ORB, pencocokan fitur dengan metode FLANN (*Fast Library for Approximate Nearest Neighbor*), serta pelacakan posisi *shuttlecock* menggunakan KCF (*Kernelized Correlation Filter*) yang dikombinasikan dengan penyaringan *Exponential Moving Average* (EMA) untuk menstabilkan hasil antar frame. Evaluasi dilakukan terhadap performa sistem dalam mendeteksi *shuttlecock* pada berbagai kondisi dan latar belakang, menggunakan metrik akurasi, *precision*, *recall*, dan *F1-score*. Hasil pengujian menunjukkan bahwa sistem mencapai akurasi rata-rata sebesar 81,1%, *precision* sebesar 85,4%, *recall* sebesar 92,6%, dan *F1-score* sebesar 88,9%, dengan rata-rata waktu pemrosesan 20,6 ms per frame. Hasil tersebut membuktikan bahwa algoritma ORB memiliki kemampuan yang andal dalam mengenali *shuttlecock* berdasarkan fitur visual. Penelitian ini memberikan kontribusi dalam pengembangan teknologi visi komputer yang dapat dimanfaatkan sebagai sistem bantu analisis latihan bulu tangkis untuk meningkatkan efektivitas dan efisiensi pelatihan.

ABSTRACT

Munsyi, Arvin Azaria. 2025. **Deteksi *Shuttlecock* Menggunakan Algoritma ORB (*ORIENTED FAST AND ROTATED BRIEF)*.** Thesis. Informatics Engineering Study Program Faculty of Science and Technology Maulana Malik Ibrahim State Islamic University Malang. Advisor: (I) Johan Ericka Wahyu Prakasa, M.Kom (II) Shoffin Nahwa Utama, M.T.

Key words: *Digital Image Processing,* FLANN, KCF *Tracker*, ORB, *Shuttlecock*.

The development of digital image processing technology has enabled the application of automatic object detection systems in various fields, including sports. One of the sports that requires image-based analysis is badminton, where *shuttlecock* detection is an important aspect in the evaluation and training process. This study aims to detect *shuttlecock* objects among other objects on the field using the ORB (Oriented FAST and Rotated BRIEF) algorithm. The research process includes image conversion, feature extraction and description using ORB, feature matching using the FLANN (Fast Library for Approximate Nearest Neighbor) method, and *shuttlecock* position tracking using KCF (Kernelized Correlation Filter) combined with Exponential Moving Average (EMA) filtering to stabilize the results between frames. Evaluation is carried out on the system's performance in detecting *shuttlecock*s under various conditions and backgrounds, using accuracy, *precision*, recall, and F1-score metrics. The test results show that the system achieved an average accuracy of 81.1%, *precision* of 85.4%, recall of 92.6%, and F1-score of 88.9%, with an average processing time of 20.6 ms per frame. These results prove that the ORB algorithm has a reliable ability to recognize *shuttlecock*s based on visual features. This research contributes to the development of computer vision technology that can be used as an aid system for badminton training analysis to improve the effectiveness and efficiency of training.

مستخلص البحث

منشي، أرفين أزاريا. 2025. كشف الريشة أو الشطكوك باستخدام خوارزمية ORB (سريع موجه ومختصر دوار). البحث الجامعي. قسم الهندسة المعلوماتية، كلية العلوم والتكنولوجيا بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرف الأول: جوهان إريكا وحي براكاسا، الماجستير؛ المشرف الثاني: صفين نحو أوتاما، الماجستير.

الكلمات الرئيسية: FLANN، متعقبKCF، ORB، معالجة صور رقمية، ريشة.

لقد أتاح تطور تكنولوجيا معالجة الصور الرقمية إمكانية تطبيق أنظمة الكشف التلقائي عن الأجسام في مجالات متعددة، بما في ذلك الرياضة. ومن بين الرياضات التي تتطلب تحليلاً معتمداً على الصور رياضة كرة الريشة، حيث يُعد كشف الريشة جانباً مهماً في عملية التقييم والتدريب. هدف هذا البحث إلى اكتشاف جسم الريشة بين الأجسام الأخرى الموجودة في الملعب باستخدام خوارزمية ORB (سريع موجه ومختصر دوار). تشمل عملية البحث مراحل تحويل الصورة، واستخراج ووصْف الخصائص باستخدام ORB، ومطابقة الخصائص باستخدام طريقة مكتبة سريعة للبحث عن أقرب جار تقريبي (FLANN)، بالإضافة إلى تتبع موضع الريشة باستخدام مرشح الترابط النواة (KCF) مع دمجه بفلترة المتوسط المتحرك الأسي (EMA) لاستقرار النتائج بين الإطارات .تم إجراء تقييم لأداء النظام في اكتشاف كرة الريشة تحت ظروف وخلفيات مختلفة، باستخدام مقاييس الدقة، الضبط، الاستدعاء، وقيمة ف1. أظهرت نتائج الاختبار أن النظام حقق دقة متوسطة بلغت 81.1%، ضبط بنسبة 85.4%، استدعاء بنسبة 92.6%، وقيمة ف1 بنسبة 88.9%، مع متوسط زمن معالجة بلغ 20.6 مللي ثانية لكل إطار. وأثبتت هذه النتائج أن خوارزمية ORB تتمتع بقدرة موثوقة على التعرف على كرة الريشة استنادًا إلى الملامح البصرية. وتساهم هذا البحث في تطوير تكنولوجيا الرؤية الحاسوبية التي يمكن الاستفادة منها كنظام مساعد لتحليل التدريب في رياضة كرة الريشة بهدف زيادة فعالية وكفاءة التدريب.

.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Permainan bulu tangkis adalah salah satu olahraga yang sangat populer di Indonesia dengan ribuan pertandingan yang digelar setiap tahun baik di level amatir maupun profesional. Bulu tangkis tidak hanya melibatkan fisik yang kuat, tetapi juga memerlukan kecepatan dan ketangkasan dalam bergerak (Pardiwala et al., 2020). Salah satu perangkat penting dalam permainan bulu tangkis adalah *shuttlecock* atau kok. *Shuttlecock* berfungsi sebagai objek yang dipukul oleh pemain dengan raket untuk mencetak poin. Meskipun kecil dan ringan, *shuttlecock* memiliki peran yang sangat besar dalam jalannya pertandingan (Ramadhan & Muzaffar, 2023).

Dalam sesi latihan intensif, *shuttlecock* sering tersebar di berbagai sudut lapangan dalam jumlah besar dan dalam posisi acak. Proses identifikasi, penentuan lokasi, dan pengambilan *shuttlecock* yang tersebar ini sering kali masih dilakukan secara manual oleh manusia, hal ini dapat menyita waktu yang cukup banyak (Khotimah & Rahmandika, 2020). Posisi *shuttlecock* yang acak menyulitkan dalam proses identifikasi dan pemantauan. Meski tampak sederhana, memastikan keberadaan *shuttlecock* secara cepat dan akurat memiliki pengaruh pada efektivitas latihan, dan efisiensi waktu.

Permasalahan utama yang sering dihadapi adalah banyaknya *shuttlecock* yang tercecer di berbagai sudut lapangan sehingga sulit untuk diindetifikasi dan dipantau secara cepat serta akurat. Permasalahan ini semakin jelas terlihat pada

sesi latihan intensif, seperti teknik *multishuttle* atau *smash repetition*, di mana jumlah *shuttlecock* yang digunakan dapat mencapai puluhan hingga ratusan dalam satu kali latihan (De Alwis et al., 2020). Kondisi tersebut membuat proses identifikasi dan pemantauan secara manual menjadi kurang efisien dan dapat mengurangi waktu istirahat pemain, menyulitkan pelatih dalam mengatur ritme latihan, serta menambah beban dalam manajemen peralatan. Oleh karena itu, dibutuhkan sistem yang mampu mendeteksi *shuttlecock* secara otomatis agar proses latihan dapat berjalan lebih efektif dan terstruktur.

Teknologi pengolahan citra digital menawarkan solusi untuk permasalahan ini. Dengan menggunakan kamera dan sistem visual, *shuttlecock* dapat dikenali melalui karakteristik bentuk, tekstur, atau fitur unik lainnya yang dapat diolah menggunakan algoritma komputer. Salah satu algoritma yang dapat digunakan dalam pengenalan objek adalah ORB (*Oriented FAST and Rotated BRIEF*). ORB merupakan penggabungan dari dua teknik, yaitu deteksi fitur FAST (*Features from Accelerated Segment Test*) dan deskripsi fitur BRIEF (*Binary Robust Independent Elementary Features*) yang telah dimodifikasi agar tahan terhadap rotasi dan perubahan skala (Imsaengsuk & Pumrin, 2021). Keunggulan ORB dibanding algoritma lain seperti SIFT atau SURF adalah kecepatannya yang tinggi, tidak berbayar (*open-source*), dan ringan dalam implementasi tanpa mengorbankan akurasi (Yao et al., 2023).

Pemilihan ORB dalam penelitian ini didasarkan pada karakteristik *shuttlecock* yang sering kali terdeteksi dalam berbagai sudut dan ukuran akibat letaknya yang tidak teratur. ORB mampu menangani tantangan tersebut karena

algoritma ini mendeteksi titik-titik kunci (*keypoints*) dan membandingkannya berdasarkan *deskriptor* yang tahan terhadap transformasi rotasi dan skala (Mahmudin et al., 2023). Dengan demikian, sistem deteksi *shuttlecock* berbasis ORB akan mampu mendeteksi keberadaan *shuttlecock* secara akurat.

Penelitian ini penting dilakukan karena dapat membantu meningkatkan efisiensi dalam proses latihan bulu tangkis. Deteksi *shuttlecock* secara otomatis menjadi langkah awal yang perlu dilakukan sebelum nantinya dikembangkan menuju sistem yang lebih lanjut.

Dalam surat *Al-Ma'idah* ayat 2 :

وَتَعَاوَنُوْا عَلَى الْبِرِّ وَالتَّقْوٰىۖ وَلَا تَعَاوَنُوْا عَلَى الْاِثْمِ وَالْعُدْوَانِۖ وَاتَّقُوا اللّٰهَۗ اِنَّ اللّٰهَ شَدِيْدُ الْعِقَابِ

"Tolong-menolonglah kamu dalam (mengerjakan) kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan permusuhan. Bertakwalah kepada Allah, sesungguhnya Allah sangat berat siksaan-Nya." *(QS. Al-Ma'idah ayat 2).*

Sesuai dengan tafsir tahlili(Tahlili, 2011), ayat ini menjelaskan kewajiban umat manusia, khususnya kaum mukmin, untuk saling membantu dalam segala bentuk kebaikan (*al-birr*) dan ketakwaan (*at-taqwā*), termasuk dalam bidang ilmu pengetahuan dan pengembangan teknologi yang membawa manfaat bagi kehidupan manusia. Sebaliknya, Islam melarang segala bentuk kerja sama dalam hal yang dapat menimbulkan kerugian, permusuhan, atau dosa.

Dalam konteks penelitian ini, nilai yang terkandung dalam ayat tersebut dapat diimplementasikan melalui pengembangan teknologi yang mendukung efisiensi, kerja sama, dan peningkatan produktivitas dalam olahraga bulu tangkis. Sistem deteksi *shuttlecock* otomatis yang dikembangkan menggunakan algoritma

ORB ini merupakan bentuk penerapan ilmu dan teknologi untuk membantu manusia bekerja lebih efektif baik pelatih, atlet, maupun pihak pengelola fasilitas olahraga. Dengan sistem yang cerdas, waktu latihan dapat dimanfaatkan secara optimal, beban kerja manusia berkurang, dan efisiensi kegiatan meningkat.

Dengan demikian, penelitian ini sejalan dengan semangat ayat *Al-Mā'idah* ayat 2, yakni menghadirkan teknologi sebagai sarana tolong-menolong dalam kebaikan dan ketakwaan, bukan untuk merugikan atau menimbulkan mudarat. Penerapan nilai-nilai Islam ini menunjukkan bahwa kemajuan teknologi, termasuk di bidang pengolahan citra digital, dapat diarahkan untuk menciptakan kemaslahatan bagi masyarakat dan mendukung tercapainya tujuan pembelajaran yang bermanfaat sesuai tuntunan agama.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan diatas, rumusan masalah pada penelitian ini adalah bagaimana mendeteksi obyek *shuttlecock* di lapangan menggunakan algoritma ORB ?

1.3 Batasan Masalah

Batasan masalah pada penelitian ini mencakup:

1. Penelitian ini hanya berfokus pada deteksi *shuttlecock* menggunakan algoritma ORB dan tidak mencakup pengembangan sistem robot pengumpulan *shuttlecock*.

2. Penelitian ini terbatas pada uji coba deteksi *shuttlecock* dengan mengukur akurasi deteksi, waktu pemrosesan, *precision*, *recall*, dan *F1-score* pada berbagai orientasi yang umum ditemui di lapangan bulu tangkis.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mendeteksi obyek *shuttlecock* di antara benda – benda lain yang ada di lapangan dengan menggunakan algoritma ORB *(Oriented FAST and Rotated BRIEF).*

1.5 Manfaat Penelitian Tujuan Penelitian

Manfaat penelitan dari deteksi *shuttlecock* menggunakan algoritma ORB *(Oriented FAST and Rotated BRIEF)* adalah :

1. Menambah literatur dalam bidang pengolahan citra digital, khususnya terkait implementasi algoritma ORB untuk objek kecil dengan orientasi dinamis.
2. Mendukung efisiensi pelatihan bulutangkis dengan adanya sistem yang dapat membantu mendeteksi *shuttlecock* secara otomatis.
3. Menjadi dasar bagi penelitian selanjutnya dalam mengembangkan sistem deteksi objek yang lebih kompleks dengan tingkat akurasi dan kecepatan yang lebih tinggi.

BAB II
STUDI PUSTAKA

2.1 Penelitian terkait

Dalam penelitian yang dilakukan oleh (Aglave & Kolkure, 2015) Implementasi ORB terdiri dari detektor cepat yang memiliki kecepatan dan akurasi komponen orientasi yang cepat dan akurat untuk *deskriptor* yang cepat dan singkat. ORB adalah lebih baik daripada SIFT dan SURF untuk menemukan tetangga yang berdekatan pencocokan pada *database* gambar yang besar. *Kueri* yang diusulkan metode deteksi objek berbasis *kueri* diuji untuk berbagai rotasi dan ukuran gambar yang berbeda. Eksperimental hasil percobaan menunjukkan bahwa metode ini mendeteksi objek untuk setiap sudut rotasi dan ukuran gambar yang dilatih. Dengan demikian, diusulkan metode deteksi objek berbasis *kueri* yang menggunakan ORB untuk ekstraksi fitur memecahkan masalah invarian rotasi dan skala dalam varians ke tingkat maksimum dalam deteksi objek.

Penelitian yang dilakukan oleh (Md. Shabbir et al., 2021) dijelaskan bahwa ekstraksi fitur ORB memiliki akurasi yang lebih tinggi, kompleksitas yang lebih rendah, sensitivitas tertinggi, dan tingkat kesalahan terendah. Alasan di balik penggunaan metode ini adalah metode ini lebih cepat daripada metode lain yang digunakan sebelumnya dan tidak terlalu rumit untuk digunakan. Akurasi dalam analisis tekstur juga tinggi. Selain itu, metode SURF dan SIFT telah dipatenkan, sehingga diperlukan pembayaran kepada pemegang hak paten untuk menggunakan metode tersebut. Karena menggunakan *Gaussian Naïve Bayes* dalam *Python*, metode prediksi menjadi lebih mudah. *OpenCV* memiliki berbagai

fungsi matematika yang lebih mudah digunakan dalam *Python* dibandingkan dengan platform lain. ORB mengatasi semua keterbatasan metode *FAST* dan *BRIEF* dan lebih cepat daripada ekstraksi fitur lainnya. Akurasi yang diperoleh oleh model yang kami latih dari 160 gambar adalah 91,67% dan sensitivitas atau *recall* 90,32% yang lebih tinggi dari penelitian lain yang dibandingkan. Selain itu, ORB dapat digunakan pada perangkat berdaya rendah, alasan inilah yang membuatnya menjadi pilihan yang lebih baik daripada metode lainnya.

Algoritma ORB (*Oriented FAST and Rotated BRIEF*) digunakan dalam penelitian yang dilakukan oleh (Singh et al., 2025) untuk lokalizasi robot TurtleBot4 berbasis *Visual Odometry* (VO) di lingkungan dalam ruangan menggunakan kamera RGB-D. ORB mengekstraksi sekitar 2000 fitur per *frame* dengan detektor *FAST* dan *deskriptor BRIEF* yang tahan terhadap rotasi. Pencocokan fitur dilakukan dengan *brute-force matching* menggunakan *Hamming distance* untuk mengestimasi pergerakan robot berdasarkan matriks esensial dan *5-point algorithm*. Untuk mengurangi kesalahan akumulatif (*drift*), optimasi *pose graph* diterapkan dengan algoritma *Levenberg-Marquardt*. Hasil eksperimen menunjukkan bahwa metode ORB lebih akurat dibandingkan *wheel odometry*, dengan *error* RMSE sekitar 4-5%. Algoritma ini terbukti efisien untuk lokalizasi robot berbasis kamera dengan sumber daya komputasi terbatas.

Penelitian yang dilakukan oleh (Bao et al., 2022) menyajikan metode pencocokan gambar baru untuk estimasi pose kamera berdasarkan segmentasi point *cloud*. *Oriented FAST and Rotated BRIEF* (ORB) digunakan untuk mengekstrak titik-titik kunci, yang kemudian diekstrak berdasarkan bidang point

cloud yang cocok. Bidang-bidang point *cloud* disegmentasi berdasarkan gambar kedalaman, dan kemudian dicocokkan dengan jarak pusat antar bidang.

Penelitian lain oleh (Tilaksono et al., 2022) menggunakan metode ORB dan MSER untuk mendeteksi citra serta posisi *barcode* secara otomatis menggunakan *drone*. Dalam penelitian ini, metode ORB digunakan untuk mengekstraksi fitur visual dari gambar *barcode* yang diambil melalui kamera. Hasilnya menunjukkan bahwa ORB mampu melakukan deteksi dengan efisien dalam lingkungan nyata seperti *warehouse*, serta mampu mengurangi waktu pengecekan *barcode* dari 8 menit secara manual menjadi 4 menit secara otomatis. Keunggulan ORB dalam mengenali fitur objek dengan akurat menjadikannya cocok untuk diterapkan dalam sistem deteksi objek lain yang memerlukan kecepatan dan efisiensi tinggi, seperti sistem pengambilan *shuttlecock* otomatis di lapangan bulu tangkis.

Berdasarkan berbagai penelitian sebelumnya, penelitian ini berfokus pada pengembangan sistem deteksi *shuttlecock* menggunakan algoritma ORB. Sistem ini dirancang untuk meningkatkan akurasi serta efisiensi dalam mendeteksi *shuttlecock* secara otomatis pada citra maupun video uji. Hasil penelitian ini diharapkan dapat menjadi dasar bagi pengembangan sistem pemantauan atau analisis permainan bulu tangkis yang lebih cerdas dan efektif. Dengan penerapan teknologi ini, proses identifikasi *shuttlecock* dapat dilakukan secara lebih cepat dan akurat, sehingga mendukung peningkatan efisiensi dalam kegiatan latihan serta analisis performa atlet.

Tabel 2.1 Penelitian Terdahulu

No.	Penulis	Judul	Objek dan Metode	Hasil
1.	Junqi Bao, Xiaochen Yuan, Chan Tong Lam (Bao et al., 2022)	*Robust Image Matching for Camera Pose Estimation Using Oriented FAST and Rotated BRIEF*	Pencocokan gambar yang kuat untuk estimasi pose kamera menggunakan metode *Oriented-Rotated Brief* (Orb)	Dalam penelitian ini pendekatan pencocokan gambar berdasarkan segmentasi dan pencocokan point *cloud* menggunakan ORB lebih kuat dan akurat dibandingkan dengan metode pencocokan metode tradisional
2.	Xu Liu, Yongshun Zhang, Qiancheng Wang (X. Liu et al., 2025)	*Posture Detection of Dual-Hemisphere Capsule Robot Based on Magnetic Tracking Effects and ORB-AEKF Algorithm*	Dalam metode ini, DHCR menyadari penyesuaian postur titik tetap berdasarkan efek pelacakan, dan titik fitur dikenali dan dicocokkan dengan bantuan algoritma ORB pada gambar GI yang diperoleh oleh sensor penglihatan.	Hasilnya, metode deteksi postur tubuh berdasarkan efek pelacakan dan algoritma ORB-AEKF terbentuk. Efektivitas dan keunggulan metode yang diusulkan diverifikasi melalui eksperimen, yang memberikan dasar yang baik untuk kontrol loop tertutup yang akurat dari DHCR.
3.	Hafizhal Tilaksosno, Berlian Al Khindi, Fivitria Istiqomah (Tilaksono et al., 2022)	Deteksi Citra dan Posisi *Barcode* Menggunakan Metode *Oriented FAST and Rotated BRIEF* (ORB) dan Maximally Stable Extremal Regions (MSER)	Sistem ini dirancang untuk mengatasi masalah dengan *drone* yang secara otomatis memeriksa dan memindai *barcode* menggunakan metode ORB dan MSER.	Berdasarkan hasil yang diperoleh, sistem ini mampu memindai rata-rata 12 *barcode* dalam waktu 4 menit. Sebelumnya, proses manual membutuhkan 8 menit untuk jumlah yang sama, sehingga terdapat penghematan waktu sebesar 4 menit.
4.	Chaoqun Ma, Xiaoguang Hu, Jin Xiao, Guofeng Zhang (Ma et al., 2021)	*Homogenized* ORB *Algorithm Using Dynamic Threshold and Improved Quadtree*	Algoritma ORB yang dihomogenisasi menggunakan ambang batas dinamis dan metode quadtree yang lebih baik diusulkan dalam penelitian ini	Hasil penelitian menunjukkan bahwa, dengan mempertimbangkan akurasi dan efisiensi waktu nyata, QTORB dapat secara efektif meningkatkan keseragaman distribusi titik-titik fitur.

No.	Penulis	Judul	Objek dan Metode	Hasil
5.	Antonio Jose Oliveira, Bruno Miguel Ferreira, Nuno Alexandre Cruz (Oliveira et al., 2021)	*A Performance Analysis of Feature Extraction Algorithms for Acoustic Image-Based Underwater Navigation*	Navigasi bawah air melalui akuisisi gambar akustik dengan membandingkan algoritma SURF (***Speeded-Up Robust Features***), ORB (*Oriented FAST and Rotated BRIEF*), BRISK (*Binary Robust Invariant Scalable Keypoints*), dan SURF-Harris, berdasarkan kinerja prosedur deteksi dan deskripsi fitur mereka	Perbandingan yang dilakukan juga memberikan bukti bahwa pengembangan lebih lanjut dari metodologi deskripsi fitur saat ini mungkin diperlukan untuk analisis citra akustik bawah air.

Penelitian terdahulu telah banyak memanfaatkan algoritma ORB (*Oriented FAST and Rotated BRIEF*) dalam berbagai bidang, seperti deteksi objek, pencocokan citra, hingga estimasi posisi kamera. ORB dikenal memiliki kelebihan dalam hal kecepatan dan efisiensi, serta mampu mendeteksi fitur yang tetap stabil meskipun mengalami rotasi. ORB lebih ringan secara komputasi sehingga cocok digunakan pada perangkat dengan spesifikasi terbatas. Namun, sebagian besar dari penelitian-penelitian tersebut hanya memfokuskan penerapan ORB pada objek yang statis dan kondisi lingkungan yang relatif terkendali. Selain itu, objek yang diteliti umumnya memiliki bentuk yang jelas, pencahayaan yang stabil, dan jarang mengalami rotasi atau perubahan orientasi ekstrem.

Penelitian ini memiliki perbedaan yang signifikan karena secara khusus meneliti penerapan algoritma ORB dalam mendeteksi *shuttlecock* objek kecil yang sering kali berubah posisi, orientasi, dan skala saat berada dalam lingkungan dinamis seperti lapangan bulu tangkis. Fokus penelitian ini tidak hanya pada

kemampuan ORB dalam mengenali pola citra, tetapi juga pada evaluasi kinerjanya dalam kondisi nyata yang melibatkan sudut pandang yang bervariasi.

2.2 Pengolahan Citra dalam Deteksi Objek

Pengolahan citra digital adalah teknik pemrosesan gambar menggunakan algoritma komputer untuk meningkatkan kualitas citra dan mengekstrak informasi. Salah satu aspek penting dalam pengolahan citra adalah ekstraksi fitur, yang membantu mengidentifikasi karakteristik unik suatu objek seperti tepi dan sudut, sehingga dapat dikenali dalam berbagai kondisi pencahayaan, rotasi, dan skala (Susim & Darujati, 2021).

***Oriented FAST and Rotated BRIEF* (ORB)** adalah algoritma ekstraksi fitur yang dikembangkan sebagai alternatif lebih cepat dari ***Scale-Invariant Feature Transform* (SIFT)** dan ***Speeded-Up Robust Features* (SURF)**. ORB mengombinasikan deteksi fitur FAST dengan *deskriptor* BRIEF yang dimodifikasi agar tahan terhadap rotasi dan perubahan skala, memungkinkan deteksi dan pencocokan fitur yang lebih efisien (H. Liu et al., 2020). Algoritma ini sering digunakan dalam sistem berbasis visi komputer karena kecepatan pemrosesannya yang tinggi (Yan et al., 2023).

Dalam deteksi objek dinamis seperti *shuttlecock*, ORB memiliki beberapa keunggulan. Algoritma ini dapat mengenali objek yang mengalami perubahan orientasi secara signifikan, yang merupakan karakteristik utama *shuttlecock* saat melayang atau jatuh di lapangan. Selain itu, ORB lebih efisien dibandingkan metode lain dalam pemrosesan citra, sehingga sangat sesuai untuk sistem robotik yang membutuhkan deteksi cepat dan akurat (Sun et al., 2020). Dengan

keunggulan ini, ORB menjadi algoritma yang tepat untuk penelitian ini dalam mendeteksi *shuttlecock* secara otomatis guna meningkatkan efisiensi pengumpulan *shuttlecock* di lapangan bulu tangkis.

2.3 Algoritma ORB (*Oriented FAST and Rotated BRIEF*)

Algoritma ORB (*Oriented FAST and Rotated BRIEF*) adalah metode ekstraksi fitur yang menggabungkan deteksi fitur **FAST (*Features from Accelerated Segment Test*)** dengan *deskriptor* **BRIEF (*Binary Robust Independent Elementary Features*)** yang telah dimodifikasi agar lebih tahan terhadap rotasi dan penskalaan (Dai & Wu, 2023). FAST digunakan dalam ORB untuk mendeteksi sudut atau titik fitur pada gambar dengan cepat, sementara BRIEF bertanggung jawab dalam mendeskripsikan fitur tersebut dalam bentuk vektor biner yang lebih efisien. Salah satu keunggulan utama ORB adalah kemampuannya dalam menangani objek yang mengalami perubahan orientasi dan skala. ORB menggunakan teknik ***rotation-aware* BRIEF** yang memungkinkan *deskriptor* tetap akurat meskipun objek mengalami rotasi (Corresp, 2020).

Selain itu, ORB menerapkan ***pyramidal scaling***, yang membuatnya mampu mengenali objek dalam berbagai ukuran tanpa kehilangan informasi penting. Hal ini menjadikan ORB sangat efektif dalam mendeteksi *shuttlecock* yang sering berubah posisi dan ukuran selama permainan bulu tangkis. Kemampuan ORB dalam menghadapi variasi pencahayaan juga menjadi faktor penting dalam pemilihannya untuk penelitian ini. Studi sebelumnya menunjukkan bahwa ORB lebih tahan terhadap perubahan iluminasi dibandingkan metode lain seperti SIFT dan SURF (Chen et al., 2021). Dengan efisiensi komputasi yang

tinggi serta kemampuannya dalam menangani rotasi dan penskalaan, ORB menjadi pilihan ideal dalam deteksi *shuttlecock* secara otomatis untuk sistem robotik yang dirancang dalam penelitian ini.

2.4 Penerapan ORB dalam Deteksi Objek

Algoritma ORB (*Oriented FAST and Rotated BRIEF*) telah diterapkan dalam berbagai penelitian yang berfokus pada deteksi objek dalam kondisi dinamis dan nyata. Keunggulan ORB terletak pada kemampuannya untuk mendeteksi dan mengenali objek yang mengalami rotasi, perubahan skala, serta kondisi pencahayaan yang bervariasi, yang menjadikannya sangat relevan untuk digunakan dalam sistem yang membutuhkan kecepatan dan akurasi tinggi, seperti dalam sistem konveyor. Sebagai contoh, Penelitian oleh (Tilaksono et al., 2022) juga menunjukkan efektivitas ORB dalam sistem *drone* untuk deteksi *barcode*, di mana waktu pemrosesan berhasil dipangkas setengahnya dibandingkan proses manual. Hal ini menunjukkan bahwa ORB mampu memberikan efisiensi tinggi dalam proses pengolahan citra yang cepat dan adaptif terhadap perubahan lingkungan.

Dari penelitian tersebut, dapat disimpulkan bahwa algoritma ORB sangat efektif digunakan dalam deteksi objek, terutama untuk sistem yang memerlukan kecepatan, efisiensi komputasi, serta ketahanan terhadap rotasi dan skala, seperti pada deteksi *shuttlecock* dalam permainan bulu tangkis.

2.5 Cara Kerja Algoritma ORB dalam Mendeteksi *Shuttlecock*

Algoritma ORB (*Oriented FAST and Rotated BRIEF*) bekerja dengan

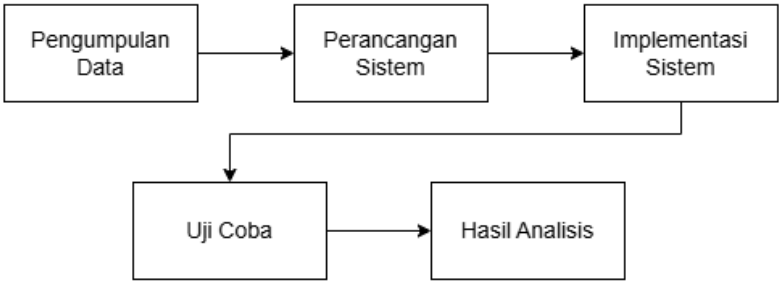
menggabungkan deteksi fitur menggunakan metode *FAST* dan deskripsi fitur dengan *BRIEF* yang telah disesuaikan agar tahan terhadap rotasi dan skala (Wu, 2023). Dalam proses deteksi *shuttlecock*, citra pertama-tama dikonversi ke *grayscale* untuk menyederhanakan informasi dan mempercepat pemrosesan. Kemudian, algoritma *FAST* digunakan untuk mendeteksi titik-titik kunci (*keypoints*) pada *shuttlecock*, seperti tepi kepala dan susunan bulu. Setelah titik-titik ini ditemukan, ORB menghitung arah orientasi untuk setiap titik kunci guna memastikan bahwa *deskriptor* yang dihasilkan tetap akurat meskipun *shuttlecock* mengalami rotasi atau perubahan sudut. *Deskriptor BRIEF* yang telah dimodifikasi menghasilkan representasi biner dari area sekitar titik kunci, yang sangat efisien untuk proses pencocokan fitur (Forero et al., 2021).

Proses pencocokan fitur dilakukan antara citra referensi *shuttlecock* dan citra dari kamera menggunakan metode FLANN (*Fast Library for Approximate Nearest Neighbors*). Jika sejumlah kecocokan yang cukup ditemukan, sistem akan menghitung transformasi spasial seperti rotasi dan translasi untuk menentukan posisi *shuttlecock* secara akurat dalam citra. ORB juga mendukung teknik pyramidal scaling untuk mendeteksi objek dalam berbagai ukuran, serta memiliki ketahanan terhadap kondisi pencahayaan yang bervariasi (Adhinata et al., 2021). Oleh karena itu, algoritma ini sangat sesuai untuk mendeteksi *shuttlecock* yang memiliki bentuk kompleks dan dapat bergerak bebas dengan orientasi dan pencahayaan yang berubah-ubah. Dengan pendekatan ini, ORB dapat memberikan solusi deteksi objek yang cepat, efisien, dan tangguh dalam konteks permainan bulu tangkis.

BAB III
DESAIN DAN IMPLEMENTASI

3.1 Desain Penelitian

Penelitian ini dilakukan secara bertahap dan sistematis dengan tujuan untuk merancang dan membangun sistem deteksi *shuttlecock* otomatis menggunakan algoritma ORB (*Oriented FAST and Rotated BRIEF*). Setiap tahap dalam penelitian ini saling berkaitan untuk mencapai hasil akhir berupa sistem yang mampu mendeteksi *shuttlecock* secara akurat dan efisien. Adapun tahapan penelitian dalam bentuk diagram desain penelitian dapat digambarkan pada Gambar 3.1 berikut:



Gambar 3.1 Diagram Desain Penelitian

Penjelasan dari masing-masing tahap pada gambar 3.1 adalah sebagai berikut:

1. Pengumpulan Data

   Tahap awal dilakukan dengan menyiapkan gambar referensi *shuttlecock* yang diambil dari berbagai sudut pandang, jarak, dan orientasi. Gambar-gambar ini digunakan sebagai data acuan dalam proses pencocokan fitur

menggunakan algoritma ORB. Gambar referensi ini nantinya diproses untuk ekstraksi *keypoint* dan *deskriptor*, dan akan dibandingkan dengan citra dari video untuk mendeteksi keberadaan *shuttlecock*. Proses ini menjadi dasar bagi sistem dalam mengenali *shuttlecock* secara akurat..

2. Perancangan Sistem

   Setelah data terkumpul, tahap ini dilakukan untuk menyusun struktur teknis sistem. Pada tahap ini dirancang alur kerja sistem, modul pemrosesan citra, pemilihan metode ORB untuk ekstraksi fitur dan FLANN untuk pencocokan fitur, KCF untuk pelacakan *shuttlecock*, EMA untuk menstabilkan hasil pelacakan, serta rancangan antarmuka pengguna (UI). Perancangan sistem dilakukan berdasarkan kebutuhan untuk mendeteksi *shuttlecock* melalui proses pemrosesan citra secara otomatis.

3. Implementasi Sistem

   Setelah desain sistem selesai, tahap implementasi dilakukan dengan membangun program menggunakan bahasa pemrograman *Python* dan pustaka *OpenCV*. Pada tahap ini, setiap komponen seperti *preprocessing* citra, deteksi fitur, pencocokan fitur, pelacakan, dan penyaringan mulai diterapkan dan diintegrasikan.

4. Uji Coba

   Sistem yang telah dibangun diuji untuk menilai kinerjanya dalam mendeteksi *shuttlecock*. Pengujian dilakukan dengan memperhatikan 5 aspek utama, yaitu akurasi deteksi, waktu pemrosesan, *precision*, *recall*, *F1-score*.

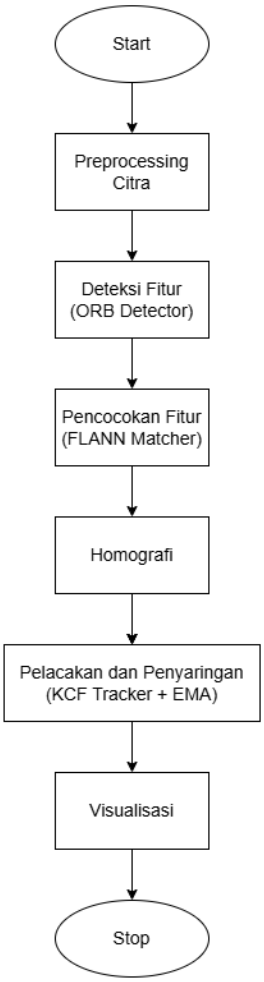
5. Hasil Analisis

Data hasil uji coba dianalisis untuk mengevaluasi efektivitas sistem secara menyeluruh. Analisis ini digunakan untuk mengetahui kekuatan dan kelemahan sistem serta sebagai dasar untuk pengembangan dan perbaikan di masa mendatang.

3.2 Perancangan Sistem Deteksi *Shuttlecock*

Sistem deteksi *shuttlecock* ini dirancang menggunakan algoritma ORB (*Oriented FAST and Rotated BRIEF*) untuk mengenali dan melacak *shuttlecock*.. Sistem bekerja dengan memproses setiap *frame* dari video uji untuk mendeteksi keberadaan *shuttlecock* secara otomatis., lalu melakukan *preprocessing* berupa konversi ke *grayscale* dan pengurangan *noise*. Setelah itu, sistem mengekstraksi fitur dari setiap *frame* menggunakan ORB, yang mencakup deteksi *keypoint* dengan metode FAST dan deskripsi fitur dengan BRIEF. *Deskriptor* dari *frame* kemudian dicocokkan dengan *deskriptor* gambar referensi menggunakan FLANN. Jika sejumlah kecocokan fitur yang memadai ditemukan, sistem menghitung homografi untuk menentukan posisi *shuttlecock,* sehingga posisi *shuttlecock* dapat diperkirakan dan ditandai dengan *bounding box*. Selanjutnya, *shuttlecock* yang telah terdeteksi dipantau secara berkelanjutan menggunakan KCF (*Kernelized Correlation Filter*) Tracker agar tidak perlu mendeteksi ulang di setiap *frame*, dan hasil pelacakan diperhalus dengan EMA (*Exponential Moving Average*) untuk menjaga stabilitas tampilan. Kombinasi ORB, FLANN, homografi, dan KCF Tracker memungkinkan sistem mendeteksi serta melacak *shuttlecock* secara konsisten dan efisien sepanjang video uji.

Diagram sistem ini ditunjukkan pada Gambar 3.2 :



Gambar 3.2 Diagram Sistem

Sistem menggunakan metode pencocokan fitur berbasis *FLANN (Fast Library for Approximate Nearest Neighbors)*, yang dirancang untuk mempercepat proses pencarian kecocokan fitur di antara sejumlah besar data visual. FLANN bekerja dengan membandingkan *deskriptor* fitur yang telah diekstraksi dari setiap *frame* video dengan *deskriptor* yang berasal dari citra referensi *shuttlecock*. Proses pencocokan dilakukan dengan mencari pasangan fitur yang memiliki jarak terkecil

di ruang vektor, biasanya menggunakan metrik jarak seperti *Hamming distance* untuk mengukur tingkat kemiripan antar fitur. Karena *shuttlecock* dapat muncul dengan variasi sudut, ukuran, dan posisi, FLANN sangat efektif dalam melakukan pencocokan fitur secara cepat dan akurat, bahkan ketika terjadi rotasi atau perubahan skala pada gambar.

Setelah fitur-fitur yang cocok ditemukan, FLANN menghasilkan pasangan titik-titik koordinat antara citra referensi dan citra yang sedang dianalisis. Koordinat ini digunakan untuk menghitung transformasi spasial, seperti translasi, rotasi, dan perubahan skala objek melalui algoritma *homografi*. Transformasi tersebut memungkinkan sistem memperkirakan posisi aktual *shuttlecock* pada setiap *frame* video dan membentuk *bounding box* yang menandai area deteksi.

Selanjutnya, posisi *shuttlecock* yang telah terdeteksi akan dipantau secara berkelanjutan menggunakan algoritma *KCF (Kernelized Correlation Filter)* agar sistem tidak perlu melakukan deteksi ulang di setiap *frame*. Untuk menjaga stabilitas tampilan, hasil pelacakan diperhalus dengan metode *Exponential Moving Average (EMA)*, sehingga pergerakan *bounding box* terlihat lebih halus dan tidak bergetar. Kombinasi antara FLANN, homografi, dan KCF Tracker menjadikan sistem mampu mendeteksi serta melacak *shuttlecock* secara konsisten dan efisien pada keseluruhan video uji.

Selain deteksi visual, sistem juga mencatat semua hasil dalam sebuah file log. File log ini menyimpan informasi penting seperti nomor *frame*, koordinat posisi *shuttlecock*, serta waktu pemrosesan untuk setiap *frame*. Data tersebut dapat dianalisis lebih lanjut untuk mengevaluasi performa sistem dan sebagai dasar

untuk pengembangan atau optimasi metode deteksi yang digunakan.

Untuk mengukur performa sistem, file log yang telah tersimpan dapat dianalisis menggunakan perangkat lunak seperti Microsoft Excel, *Python* (dengan pustaka Pandas dan Matplotlib), atau MATLAB. Beberapa metrik utama yang dapat dihitung antara lain:

- **Rata-rata akurasi deteksi**: digunakan untuk menilai sejauh mana sistem dapat mendeteksi *shuttlecock* dengan benar dari keseluruhan data, sebagai ukuran umum performa deteksi.
- ***Precision*:** mengukur ketepatan sistem dalam mendeteksi *shuttlecock* dari seluruh objek yang dideteksi. Nilai *precision* yang tinggi menunjukkan bahwa sistem jarang melakukan kesalahan deteksi (*false positive*).
- ***Recall*:** mengukur kemampuan sistem dalam mendeteksi semua *shuttlecock* yang muncul selama pengujian. Nilai *recall* yang tinggi menunjukkan bahwa sistem jarang melewatkan *shuttlecock* yang seharusnya terdeteksi (*false negative*).
- ***F1-score*:** digunakan untuk melihat keseimbangan antara nilai *precision* dan *recall*. Nilai *F1-score* yang tinggi menunjukkan sistem mampu mendeteksi *shuttlecock* dengan tepat dan konsisten di seluruh *frame* video.
- **Waktu pemrosesan rata-rata per *frame***: mengukur efisiensi sistem dalam memproses citra dari setiap *frame* video, serta menunjukkan kecepatan kerja algoritma deteksi dalam menyelesaikan proses identifikasi *shuttlecock*.

  Visualisasi hasil analisis, seperti grafik garis atau histogram, dapat

digunakan untuk menggambarkan performa sistem secara keseluruhan. Dari hasil evaluasi tersebut, kelemahan sistem dapat diidentifikasi dan digunakan sebagai dasar perbaikan lanjutan pada algoritma serta parameter pendukung lainnya.

Untuk memastikan sistem bekerja optimal dalam berbagai kondisi lingkungan, beberapa teknik tambahan diterapkan, seperti *Gaussian Blur* untuk mengurangi *noise* pada citra, serta penggunaan metode *homografi* untuk mengoreksi perspektif *shuttlecock* saat di lapangan. Kombinasi teknik ini diharapkan dapat meningkatkan akurasi deteksi serta mempercepat waktu pemrosesan sistem dalam konteks pengolahan citra berbasis video.

3.3 Rancangan Pengolahan Citra untuk Deteksi *Shuttlecock*

Proses pengolahan citra dalam sistem ini terdiri dari beberapa tahap utama yang dilakukan secara berulang pada setiap *frame* video yang diterima dari kamera:

1. **Akuisisi Gambar** Sistem memperoleh input berupa video melalui pustaka OpenCV, kemudian membaca setiap *frame* untuk dianalisis pada tahap pemrosesan berikutnya.
2. ***Preprocessing*** Preprocessing citra dimulai dengan penerapan ***Gaussian Blur*** untuk mengurangi *noise* pada citra. Filter *Gaussian* digunakan dengan fungsi yang dijelaskan pada rumus 3.1:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad (3.1)$$

Sumber : (Ibrahim et al., 2021)

di mana:

- $G(x, y)$ adalah hasil konvolusi dari citra dengan kernel *Gaussian*
- σ adalah standar deviasi dari distribusi *Gaussian* yang menentukan seberapa besar perataan diterapkan
- x dan y adalah koordinat piksel dalam citra.

Setelah citra dihaluskan, dilakukan konversi ke *grayscale* untuk menghilangkan informasi warna yang tidak diperlukan dan meningkatkan efisiensi pemrosesan. Konversi ini menggunakan rumus:

$$I_{gray} = 0.299 \times R + 0.587 \times G + 0.114 \times B \quad (3.2)$$

Sumber : (Khudhair et al., 2023)

Teknik ini penting untuk mengurangi gangguan dari latar belakang yang kompleks dan meningkatkan akurasi deteksi fitur.

3. **Ekstraksi Fitur dengan ORB** Tahap pertama ORB adalah mendeteksi titik kunci (*keypoints*) menggunakan FAST-9. FAST bekerja dengan memeriksa 16 piksel yang membentuk lingkaran di sekitar piksel pusat. Sebuah titik dianggap sebagai *corner* apabila terdapat sekuens berurutan dari piksel yang intensitasnya lebih terang atau lebih gelap dari pusat, melebihi ambang batas (*threshold*). FAST menentukan *corner* melalui komparasi intensitas, jika terdapat K piksel bertetangga yang intensitasnya memenuhi syarat perbedaan tertentu (lebih terang/gelap dari pusat), maka titik pusat adalah *corner*. FAST sendiri tidak menghasilkan informasi

orientasi. Oleh karena itu, ORB menambahkan metode perhitungan arah (*orientation*) menggunakan *intensity centroid*

Orientasi patch dihitung menggunakan momen intensitas :

$$m_{10} = \sum_{x} x\, I(x, y), m_{10} = \sum_{y} y\, I(x, y) \tag{3.3}$$

Sumber :(Rublee & Bradski, 2014)

Sudut orientasi dihitung dengan:

$$\theta = atan2(m_{01}, m_{10}) \tag{3.4}$$

Sumber :(Rublee & Bradski, 2014)

Dimana :

- m10: seberapa berat piksel terang condong ke arah kanan-kiri.
- m01: seberapa berat piksel terang condong ke arah atas-bawah.

Orientasi ini membuat fitur menjadi rotation-invariant.

Rumus di atas secara langsung diimplementasikan dalam Pseudocode 3.1 fungsi *ICAngles()* pada orb.cpp:

Pseudocode 3.1 Fungsi ICAngles pada orb.cpp

```
int m_01 = 0, m_10 = 0;

// garis pusat (v = 0)
for (int u = -half_k; u <= half_k; ++u)
m_10 += u * center[u];

// baris lain (v != 0)
m_10 += u * (val_plus + val_minus);
m_01 += v * v_sum;

pts[ptidx].angle = fastAtan2((float)m_01, (float)m_10);
```

Nilai orientasi ini kemudian digunakan untuk merotasi pola sampling BRIEF sehingga deskriptor menjadi tahan rotasi. BRIEF menghasilkan deskriptor biner dengan membandingkan intensitas pasangan piksel:

$$\mathcal{T}(p; x, y = \begin{cases} 1, & p(x) < p(y) \\ 0, & p(x) \geq p(y) \end{cases} \quad (3.5)$$

Sumber : (Rublee & Bradski, 2014)

yang kemudian disusun menjadi deskriptor 256-bit:

$$f_n(p) = \sum_{i=1}^{n} 2^{i-1} \tau(p; x_i, y_i) \quad (3.6)$$

Sumber :(Rublee & Bradski, 2014)

Untuk membuatnya invarian rotasi, BRIEF diputar menggunakan matriks rotasi:

$$S_\theta = R_\theta S, \quad R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \quad (3.7)$$

Sumber :(Rublee & Bradski, 2014)

sehingga diperoleh rBRIEF yang lebih stabil dan diskriminatif, rumus diatas secara langsung diimplementasikan dalam Pseudocode 3.2 :.

Pseudocode 3.2 fungsi rBRIEF

```
float angle = kpt.angle * factor;
float a = cos(angle), b = sin(angle);
...
x1 = cvRound(a * pattern[i].x - b * pattern[i].y);
y1 = cvRound(b * pattern[i].x + a * pattern[i].y);
```

Implementasi seluruh proses ORB ini terjadi pada fungsi *detectAndCompute()* yang dipanggil dalam kode penelitian, sehingga sesuai dengan teori dan algoritma asli ORB.

4. **Pencocokan Fitur dengan FLANN** Setelah fitur diekstraksi, sistem mencocokkannya dengan fitur dari gambar referensi menggunakan *FLANN Based Matcher* yang dikonfigurasi dengan *Locality Sensitive Hashing*

(LSH). Karena ORB menghasilkan *deskriptor* biner, ukuran kesamaan antar *deskriptor* dihitung menggunakan *Hamming distance* pada rumus 3.3

$$d_{\mathcal{H}} = \sum_{i=0}^{n} (x_{\mathrm{i}} \oplus y_i) \tag{3.8}$$

Sumber : (Halla et al., 2025)

di mana x dan y adalah dua vektor biner (*deskriptor*) yang dibandingkan, sedangkan $\oplus$ adalah operasi XOR (*exclusive OR*). Semakin kecil nilai d_H, semakin besar kemungkinan kedua *deskriptor* tersebut berasal dari objek yang sama. Proses diatas secara langsung diimplementasikan pada Pseudocode 3.3 berikut:

Pseudocode 3.3 Pencocokan Fitur Dengan FLANN

```
matches = flann.knnMatch(des1, des2, k=2)
```

Dimana :

- des1 dan des2 adalah vektor biner (deskriptor ORB).
- FLANN LSH menghitung *Hamming distance* otomatis antara setiap deskriptor des1[i] dengan semua deskriptor di des2
- Hasil perhitungan jarak disimpan di matches, di mana setiap elemen m.distance adalah nilai $d_H$antara dua deskriptor.
- Jadi, operasi XOR ($\oplus$) yang ada di rumus secara internal dilakukan oleh FLANN untuk menghitung jarak biner antar deskriptor.

Untuk memastikan bahwa pasangan deskriptor yang dipilih benar-benar cocok, digunakan *ratio test*. *Ratio test* membandingkan Hamming distance dari deskriptor terdekat (m_1) dengan deskriptor kedua terdekat (m_2)

Secara matematis, ratio test dapat dituliskan sebagai:

$$d(m_1) < r \cdot d(m_2) \quad (3.9)$$

Sumber :(Lowe, 2004)

Proses diatas secara langsung implementasi seperti pada Pseudocode 3.4 berikut:

Pseudocode 0.4

```
if m.distance < ratio * n.distance:
     good.append(m)
return good
```

Dimana :

- m adalah deskriptor terdekat (nearest neighbor).
- n adalah deskriptor kedua terdekat (second nearest neighbor).
- m.distance = $d_H(m_1)$, n.distance = $d_H(m_2)$.
- Ratio test membandingkan apakah jarak terdekat cukup **lebih kecil** daripada jarak kedua terdekat (threshold 0,75).
- Hanya pasangan yang lolos kondisi ini dimasukkan ke good_matches, yang berarti deskriptor tersebut kemungkinan besar benar-benar cocok.

Dengan demikian, ratio test membantu **mengurangi false positive** akibat deskriptor yang mirip namun bukan dari objek yang sama

5. **Estimasi Posisi *Shuttlecock* dengan Homografi** Jika jumlah fitur yang cocok melebihi ambang batas, sistem menghitung transformasi *homografi* untuk menentukan posisi *shuttlecock* pada *frame* video. Metode ini

memproyeksikan area citra referensi ke posisi sebenarnya di *frame* uji, menghasilkan koordinat *bounding box* yang mewakili lokasi *shuttlecock*..

6. **Pelacakan *Shuttlecock (Tracking*)** Setelah posisi awal *shuttlecock* ditemukan, sistem menggunakan algoritma KCF (*Kernelized Correlation Filter*) untuk melacak posisi *shuttlecock* pada *frame-frame* berikutnya. Pelacakan ini menjaga agar deteksi tetap berkelanjutan tanpa harus melakukan pencocokan fitur dari awal pada setiap *frame*. Selain itu, diterapkan metode *Exponential Moving Average* (EMA) untuk menghaluskan pergerakan *bounding box* agar tampilan deteksi terlihat stabil dan tidak bergetar.

7. **Visualisasi dan Evaluasi** Setelah *shuttlecock* berhasil dideteksi, sistem menampilkan hasil deteksi pada antarmuka pengguna dengan menampilkan *bounding box* dan koordinat *shuttlecock* di layar. Selain itu, sistem mencatat data seperti tingkat akurasi deteksi dan waktu pemrosesan untuk setiap *frame* guna mengevaluasi performa sistem.

Dengan menerapkan alur kerja ini, sistem akan dapat melakukan deteksi *shuttlecock* secara akurat dan efisien dalam berbagai kondisi. Faktor-faktor seperti akurasi deteksi, waktu pemrosesan, *precision*, *recall*, dan *F1-score* dapat mempengaruhi efektivitas sistem, sehingga sistem harus diuji dalam berbagai skenario untuk memastikan kinerja optimalnya dalam hal akurasi, efisiensi waktu, konsistensi deteksi, dan responsivitas.

3.4 Pengujian dan Evaluasi Deteksi *Shuttlecock*

Untuk memastikan keandalan dan efektivitas sistem deteksi *shuttlecock*, dilakukan serangkaian uji coba yang mencakup lima aspek utama: akurasi deteksi, waktu pemrosesan, *precision*, *recall*, dan *F1-score*. Evaluasi ini bertujuan untuk menilai kinerja sistem dalam mendeteksi *shuttlecock* dalam kondisi nyata dengan mempertimbangkan faktor-faktor seperti posisi *shuttlecock*, durasi pemantauan, dan responsivitas sistem.

1. **Uji Akurasi Berdasarkan Confusion Matrix**

   Uji ini bertujuan untuk menilai seberapa akurat sistem dalam mendeteksi *shuttlecock* berdasarkan prinsip evaluasi model klasifikasi biner. Pengujian dilakukan dengan membandingkan hasil deteksi sistem terhadap ground truth yang telah dianotasi secara manual pada setiap *frame*.

   Langkah – langkah pengujian :

   - **Menyiapkan data ground truth** dengan cara melakukan anotasi manual yang dilakukan oleh seorang ahli terhadap setiap 25 *frame* sampling per video dimana seorang ahli tersebut menentukan ada atau tidaknya *shuttlecock* disetiap *frame* dengan memberikan keterangan 1 jika ada dan 0 jika tidak ada.
   - **Menjalankan sistem deteksi ORB** untuk mendeteksi keberadaan *shuttlecock* pada setiap *frame*.
   - **Membandingkan hasil deteksi sistem dengan ground truth**, lalu mengklasifikasikan hasilnya ke dalam empat kategori:
     - *True Positive* (TP): *Shuttlecock* ada dan berhasil terdeteksi.

- *False positive* (FP): *Shuttlecock* tidak ada, tetapi sistem salah mendeteksi.
- *False negative* (FN): *Shuttlecock* ada, tetapi tidak terdeteksi oleh sistem.
- *True Negative* (TN): *Shuttlecock* tidak ada dan sistem juga tidak mendeteksi apa pun.

- Menghitung akurasi:

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (3.10)$$

Sumber : (Hasnain et al., 2020)

Contoh Perhitungan:

Jika dari total 40 *frame*, diperoleh :

- TP = 18
- TN = 12
- FP = 5
- FN = 5

Maka :

$$Akurasi = \frac{18 + 12}{18 + 12 + 5 + 5} \times 100\% = \frac{30}{40} \times 100\% = 75\%$$

- Hasil evaluasi dicatat pada tabel 3.1 Uji Akurasi Deteksi *Shuttlecock.*

Tabel 3.1 Uji Akurasi Deteksi *Shuttlecock.*

File	*Frame*	TP	TN	FP	FN	Akurasi
1.csv	25	25	0	0	0	100%
8.csv	25	24	0	0	1	96%
9.csv	25	22	0	0	3	88%

File	*Frame*	TP	TN	FP	FN	Akurasi
19.csv	25	21	0	0	4	84%
20.csv	25	24	0	0	1	96%
ALL	**650**	**494**	**33**	**84**	**39**	**81,1%**

2. **Uji *Precision* Berdasarkan Confusion Matrix**

Pengujian *precision* dilakukan untuk mengetahui tingkat ketepatan sistem dalam mendeteksi *shuttlecock* dari seluruh deteksi yang dilakukan. *Precision* menunjukkan proporsi deteksi yang benar terhadap total seluruh deteksi positif yang dihasilkan sistem.

Langkah – langkah pengujian :

- Jalankan sistem deteksi dalam waktu tertentu.
- Catat jumlah *shuttlecock* yang benar terdeteksi (***True Positive* / TP**) dan jumlah kesalahan deteksi (***False positive* / FP**)
- Hitung nilai *precision* menggunakan rumus berikut:

$$Precision = \left(\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}\right) \times 100\% \qquad (3.11)$$

Sumber : (Heydarian et al., 2022)

Contoh: Jika dari 100 objek yang terdeteksi, 95 benar merupakan *shuttlecock* dan 5 salah, maka *precision* = 95%.

- Hasilnya dicatat pada tabel 3.2 Uji *Precision* Sistem

Tabel 3.2 Uji *Precision* Sistem

File	*Frame*	TP	FP	*Precision*
1.csv	25	25	0	100%
2.csv	25	24	0	100%
18.csv	25	21	0	100%
20.csv	25	24	0	100%
ALL	**650**	**494**	**84**	**85,4%**

3. **Uji *Recall* Berdasarkan Confusion Matrix**

Pengujian *recall* digunakan untuk mengukur kemampuan sistem dalam mendeteksi semua *shuttlecock* yang seharusnya terdeteksi. *Recall* menunjukkan seberapa lengkap sistem mendeteksi objek target.

Langkah-langkah pengujian:

- Jalankan sistem dan catat jumlah *shuttlecock* yang benar-benar muncul dalam *frame* (**TP** + **FN**), serta jumlah yang berhasil terdeteksi (**TP**).
- Menghitung *recall* dengan rumus berikut:

$$Recall = \left(\frac{TP}{TP + FN}\right) \times 100\% \qquad (3.12)$$

Sumber : (Heydarian et al., 2022)

Contoh: Jika dari 100 *shuttlecock* yang muncul, 88 berhasil dideteksi (TP) dan 12 tidak terdeteksi (FN), maka *recall* = 88%

- Hasilnya dicatat pada tabel 3.3 Uji *Recall* Sistem

Tabel 3.3 Uji *Recall* Sistem

File	***Frame***	**TP**	**FN**	***Recall***
1.csv	25	25	0	100%
9.csv	25	22	3	88%
10.csv	25	24	1	96%
17.csv	25	24	1	96%
ALL	**650**	**494**	**39**	**92,6%**

4. **Uji *F1-score***

Uji *F1-score* dilakukan untuk melihat keseimbangan antara ***precision*** dan ***recall***. Nilai *F1-score* yang tinggi menunjukkan sistem mampu mendeteksi *shuttlecock* dengan tepat dan konsisten.

Langkah-langkah pengujian:

- Hitung nilai *precision* dan *recall* terlebih dahulu.
- Gunakan hasil tersebut untuk menghitung *F1-score* dengan rumus:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3.13)$$

Sumber :(Palupi et al., 2023)

Contoh : jika *precision* = 90% dan *recall* = 85%, maka :

$$F1 - Score = 2 \times \frac{90\% \times 85\%}{90\% + 85\%} = 87{,}4\%$$

- Hasilnya dicatat pada tabel 3.4 Uji *F1-score* Sistem.

Tabel 3.4 Uji *F1-score* Sistem

File	***Precision* (%)**	***Recall* (%)**	***F1-score* (%)**
1.csv	100%	100%	100%
4.csv	100%	96%	97,9%
21.csv	100%	96%	97,9%
ALL	**85,4%**	**92,6%**	**88,9%**

5. **Uji Waktu Pemrosesan**

Pengujian ini mengukur waktu yang dibutuhkan sistem untuk memproses *shuttlecock*, yang diukur dalam milidetik (ms). Sistem harus memproses gambar dengan cepat agar tidak terjadi keterlambatan signifikan dalam mendeteksi *shuttlecock*.

Langkah – langkah pengujian:

- Menjalankan sistem deteksi ORB.
- Mencatat waktu pemrosesan setiap *frame* :
  - Mencatat waktu sebelum pemrosesan dimulai.
  - Mencatat waktu setelah pemrosesan selesai.
- Menghitung waktu pemrosesan per *frame* menggunakan rumus:

$$Waktu\ Pemrosesan\ (ms) = Waktu\ Setelah\ Pemrosesan - Waktu\ Sebelum\ Pemrosesan \quad (3.14)$$

Sumber : (Hapsari et al., 2022)

Contoh *Frame* 1 :

- Waktu sebelum pemrosesan (start): 5.000 detik
- Waktu setelah pemrosesan (end): 5.012 detik

Maka waktu pemrosesan = 5.012 – 5.000 = 0.012 detik = 12 ms

- Hasilnya dicatat pada tabel 3.5 Uji Waktu Pemrosesan.
- Menghitung rata-rata waktu pemrosesan per *frame* setelah pengujian selesai dengan cara menghitung rata-rata dari seluruh waktu pemrosesan yang tercatat.

Tabel 3.5 Uji Waktu Pemrosesan

File	*Frame*	Min (ms)	Max (ms)	Rata-rata (ms)	Std (ms)
1.csv	935	4.16	195.04	16.4	34.42
21.csv	957	3.99	216.41	22.46	51.21
ALL	26.658	2.35	242.02	20.6	42.8

Dengan melakukan pengujian ini, sistem deteksi *shuttlecock* dapat dievaluasi secara menyeluruh untuk memastikan bahwa sistem ini dapat digunakan dalam berbagai situasi nyata. Apabila ditemukan kelemahan dalam

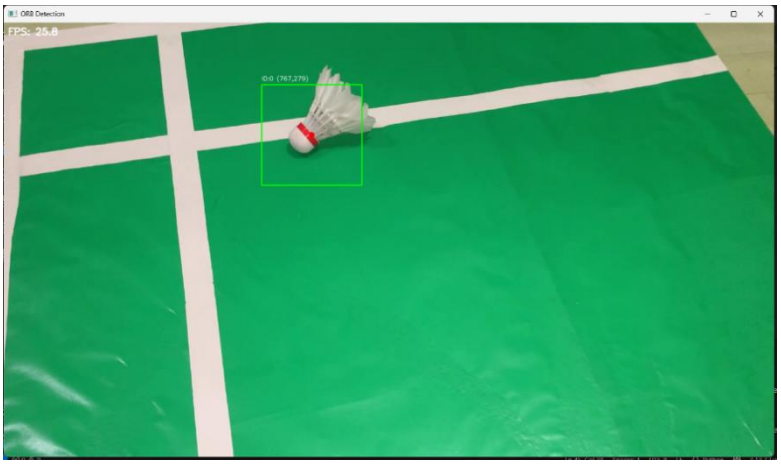
sistem, maka akan dilakukan perbaikan dan optimasi lebih lanjut agar sistem bekerja secara lebih efisien dan akurat dalam mendeteksi *shuttlecock* dalam berbagai posisi yang mungkin terjadi di lapangan

3.5 Rancangan Antarmuka Pengguna (UI)

Meskipun sistem ini berfokus pada deteksi *shuttlecock*, antarmuka pengguna dirancang untuk menampilkan hasil deteksi secara visual. Antarmuka ini menampilkan tampilan video yang sedang diproses beserta hasil deteksi *shuttlecock* dalam bentuk *bounding box* berwarna hijau, titik centroid berwarna merah di bagian tengah objek, serta informasi identitas dan koordinat *shuttlecock* yang ditampilkan di atas kotak deteksi. Selain itu, informasi kecepatan pemrosesan (*frame per second* atau FPS) juga ditampilkan pada bagian kiri atas tampilan sebagai indikator performa sistem.

3.5.1 Struktur Tampilan Antarmuka

Struktur tampilan UI dapat digambarkan seperti pada gambar 3.3 berikut :



Gambar 3.3 Tampilan UI

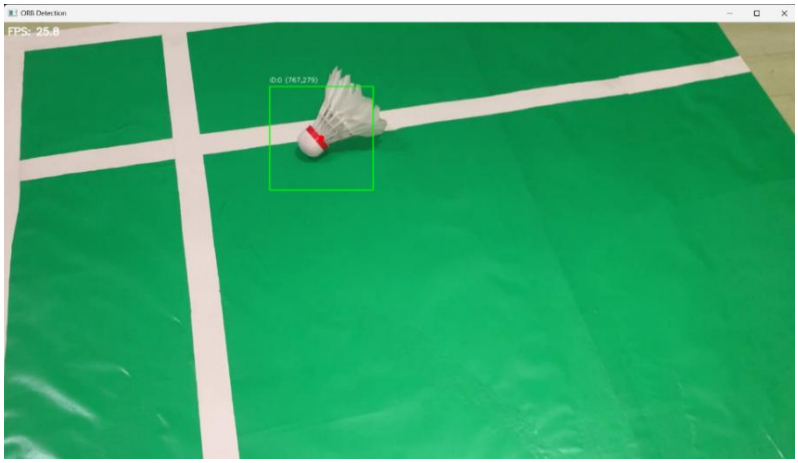
Dengan adanya antarmuka ini, pengguna dapat memantau hasil deteksi dan kinerja sistem secara visual selama proses pemrosesan video berlangsung. Melalui desain ini, sistem diharapkan mampu mendeteksi *shuttlecock* secara akurat dan stabil, serta memberikan informasi yang cepat dan jelas kepada pengguna mengenai posisi dan performa deteksi.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Hasil Implementasi Sistem

Sistem deteksi *shuttlecock* berbasis algoritma ORB (*Oriented FAST and Rotated BRIEF*) berhasil diimplementasikan. Sistem menerima input berupa video, kemudian mengekstrak *frame* untuk diproses secara berurutan. Proses implementasi terdiri dari:

1. **Pre-processing**: setiap *frame* diubah ke *grayscale* untuk mempercepat proses pencocokan fitur.
2. **Ekstraksi fitur ORB**: sistem mendeteksi titik-titik kunci (*keypoints*) pada *shuttlecock* dan mengekstrak deskripsi (descriptor).
3. **Pencocokan fitur**: fitur yang terdeteksi dibandingkan dengan template *shuttlecock* yang telah ditentukan sebelumnya.
4. **Estimasi posisi *shuttlecock***: posisi *shuttlecock* ditentukan menggunakan transformasi homografi antara citra referensi dan *frame* video untuk membentuk *bounding box* pada area yang terdeteksi
5. **Pelacakan *shuttlecock*:** posisi *shuttlecock* dipantau pada *frame* berikutnya menggunakan algoritma KCF, dan pergerakannya dihaluskan dengan metode *Exponential Moving Average* (EMA).
6. **Pencatatan hasil**: setiap hasil deteksi disimpan dalam file log CSV yang berisi nomor *frame*, status deteksi, dan waktu pemrosesan (ms).



Gambar 4..1 Hasil Deteksi

Gambar 4.1 memperlihatkan contoh hasil implementasi, *shuttlecock* yang berhasil dideteksi ditandai dengan kotak hijau dan titik centroid berwarna merah.

Analisis awal menunjukkan bahwa sistem bekerja dengan baik pada video dengan latar sederhana, tetapi mulai mengalami kesulitan ketika terdapat objek lain yang menyerupai *shuttlecock*, seperti sepatu atau gelas kecil warna putih.

Setiap proses pada sistem ini memiliki peran yang saling terhubung, sehingga membentuk alur kerja deteksi yang lengkap, mulai dari pembacaan video hingga penyimpanan hasil.

4.1.1 Tahap *Preprocessing*

Tahap *preprocessing* merupakan langkah awal yang sangat penting dalam proses deteksi karena menentukan kualitas fitur yang akan diekstraksi pada tahap berikutnya. Dalam penelitian ini, setiap *frame* video terlebih dahulu dikonversi menjadi citra keabuan (*grayscale*) untuk menyederhanakan informasi warna dan mempercepat proses perhitungan.

Selain itu, sistem juga menerapkan *Gaussian Blur* untuk mengurangi noise

yang mungkin muncul akibat kualitas kamera atau pencahayaan yang tidak stabil. Proses ini membantu algoritma ORB agar lebih fokus pada pola tekstur *shuttlecock* tanpa terganggu oleh *piksel* acak.

Proses *preprocessing* dilakukan melalui fungsi preprocess() yang berasal dari modul orb_detector.py.

Berikut pseudocode 4.1 yang menggambarkan proses tersebut:

Pseudocode 4.1 Tahap *Preprocessing*

```
Input: frame_video
Output: frame_gray

if blur_kernel != None AND blur_kernel > 0 AND blur_kernel
mod 2 == 1 then
    frame_video ← ApplyGaussianBlur(frame_video,
kernel=(blur_kernel, blur_kernel))
end if

frame_gray ← ConvertToGrayscale(frame_video)
return frame_gray
```

Dengan langkah ini, sistem memperoleh citra yang bersih, konsisten, dan siap digunakan pada proses deteksi fitur berikutnya.

4.1.2 Ekstraksi Fitur ORB

Setelah citra berhasil diproses, sistem melanjutkan ke tahap ekstraksi fitur menggunakan algoritma ORB *(Oriented FAST and Rotated BRIEF).* Proses ini diawali dengan membangun objek ORB melalui fungsi *build_orb()* yang mengatur parameter detektor. Beberapa parameter penting yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Parameter

Parameter	**Nilai**	**Keterangan**
nFeatures	3000	Jumlah maksimum titik fitur yang akan dideteksi
scaleFactor	1.2	Skala piramida untuk multi-level deteksi
nLevels	8	Jumlah level piramida yang digunakan

Parameter	Nilai	Keterangan
fastThreshold	20	Ambang batas deteksi FAST
edgeThreshold	31	Batas tepi area pencarian fitur

Setelah parameter terinisialisasi, setiap *frame* akan dipindai untuk mencari *keypoint* (titik fitur) yang unik, misalnya tepi atau pola tekstur khas dari *shuttlecock*. Kemudian, sistem menghitung *deskriptor biner* yang mewakili karakteristik setiap *keypoint* tersebut.

Berikut Pseudocode 4.2 yang menggambarkan proses tersebut:

Pseudocode 4.2 Ekstraksi Fitur ORB

```
Input: frame_gray
Output: keypoints, descriptors

orb ← ORB_Create(n_features=3000, scale_factor=1.2,
n_levels=8)

(keypoints, descriptors) ←
orb.detectAndCompute(frame_gray, mask=None)
return keypoints, descriptors
```

Hasil dari tahap ini berupa kumpulan *keypoints* dan *descriptors* yang menggambarkan pola unik dari *shuttlecock*..

4.1.3 Proses Pencocokan Fitur

Tahapan berikutnya adalah melakukan pencocokan antara fitur citra referensi *shuttlecock* dengan fitur pada *frame* video. Proses ini bertujuan untuk menemukan area pada *frame* yang memiliki kemiripan karakteristik dengan *shuttlecock*

Metode yang digunakan adalah *FLANN-based Matcher* dengan algoritma LSH (*Locality Sensitive Hashing*). Proses ini dilakukan secara efisien menggunakan fungsi *build_flann_lsh()* pada modul *orb_detector.py*. Sistem

melakukan perbandingan *K-Nearest Neighbors (KNN)* dengan k = 2, kemudian menerapkan *ratio test* untuk memfilter hasil pencocokan yang ambigu.

Secara umum, proses pencocokan fitur dapat dijelaskan melalui pseudocode 4.3 berikut:

Pseudocode 4.3 Proses Pencocokan Fitur

```
Input: descriptors_ref, descriptors_frame
Output: good_matches

matches ← matcher.knnMatch(descriptors_ref, descriptors_frame,
k=2)
good_matches ← []

for (m, n) in matches:
    if m.distance < ratio * n.distance:
        good_matches.append(m)

return good_matches
```

Dengan pendekatan ini, sistem hanya akan menyimpan hasil pencocokan yang paling akurat.

4.1.4 Estimasi Posisi *Shuttlecock* dengan Homografi

Setelah fitur yang cocok ditemukan, sistem perlu menentukan lokasi *shuttlecock* pada *frame* video. Proses ini dilakukan melalui *estimasi homografi*, yaitu perhitungan transformasi geometris antara citra referensi dan *frame* video. Dengan metode ini, area *shuttlecock* pada template dapat "diproyeksikan" ke posisi sebenarnya di *frame* video.

Proses ini dilakukan di fungsi *bbox_from_matches()* dengan langkah-langkah sebagai berikut:

1. Mengambil koordinat pasangan fitur dari hasil pencocokan terbaik.

2. Menghitung matriks homografi H menggunakan metode **RANSAC (*Random Sample Consensus*)**.
3. Jika homografi valid dan jumlah *inliers* (fitur yang benar-benar cocok) ≥ 10, sistem memproyeksikan *bounding box* citra referensi ke *frame* video.
4. Dari hasil proyeksi tersebut, sistem menghasilkan **koordinat persegi panjang (x, y, w, h)** yang menandai area *shuttlecock*.

Berikut Pseudocode 4.4 yang menggambarkan proses tersebut:

Pseudocode 4.4

```
Input: good_matches, ref_keypoints, frame_keypoints
Output: rect (x, y, w, h)

src_pts ← [ref_kp[m.queryIdx].pt for m in good_matches]
dst_pts ← [kp_frame[m.trainIdx].pt for m in good_matches]

H, mask ← FindHomography(src_pts, dst_pts, method=RANSAC,
threshold=T)

if H valid and count(mask == 1) ≥ min_inliers:
    bbox_pts ← PerspectiveTransform(ref_shape, H)
    rect ← BoundingRect(bbox_pts)
    return rect
else:
    return None
```

Dengan cara ini, sistem mampu menentukan letak *shuttlecock* secara presisi meskipun terjadi pergeseran atau rotasi pada citra.

4.1.5 Pelacakan *Shuttlecock* (Tracking)

Agar posisi *shuttlecock* tetap terpantau di setiap *frame*, sistem dilengkapi dengan pelacak (*tracker*) berbasis algoritma KCF (*Kernelized Correlation Filter*). Pelacakan ini memungkinkan sistem untuk mengikuti pergerakan *shuttlecock* secara kontinu tanpa harus melakukan deteksi dari awal setiap kali *frame* baru

muncul.

Selain itu, sistem juga menggunakan *Exponential Moving Average* (EMA) untuk menghaluskan pergerakan *bounding box* agar tampilan lebih stabil dan tidak bergetar.

Proses pelacakan berlangsung seperti pada Pseudocode 4.5 berikut:

Pseudocode 4.5

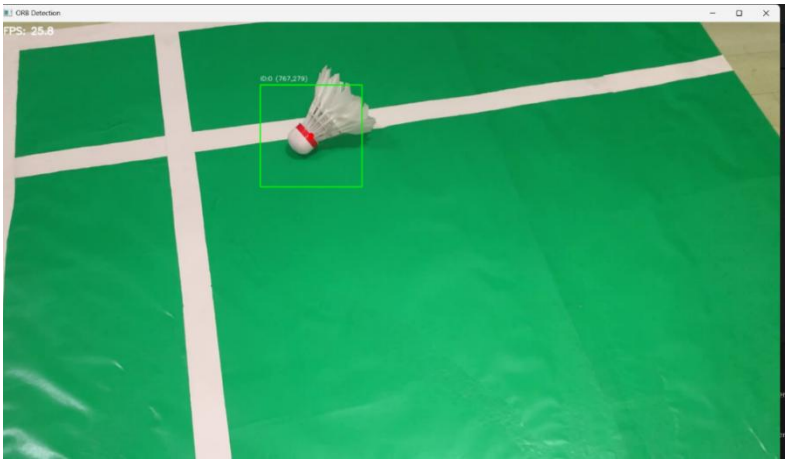
```
Input: frame, rect_awal
Output: rect_terbaru (x, y, w, h)

tracker ← CreateTracker("KCF")
tracker.init(frame, rect_awal)
smoother ← EMA_Create(alpha=0.2)
lost_count ← 0

while frame tersedia:
    ok, box ← tracker.update(frame)

    if ok:
        (x, y, w, h) ← box
        (sx, sy, sw, sh) ← EMA_Update((x, y, w, h), alpha)
        rect_terbaru ← (sx, sy, sw, sh)
        tampilkan(rect_terbaru)
        lost_count ← 0
    else:
        lost_count ← lost_count + 1
        if lost_count > 10:
            break
```

Hasil pelacakan divisualisasikan dalam bentuk kotak hijau tetap yang mengelilingi *shuttlecock*, dengan titik merah di tengahnya sebagai pusat posisi *(centroid)* seperti ditunjukkan pada Gambar 4.2.



Gambar 4.2 Hasil Pelacakan

4.1.6 Logging dan Visualisasi

Sistem juga dilengkapi dengan fitur pencatatan hasil deteksi otomatis ke dalam file CSV.

Setiap *frame* yang terdeteksi akan disimpan dengan atribut seperti pada tabel 4.2 berikut:

Tabel 4.2 Atribut Hasil Logging

Kolom	**Keterangan**
frame	Indeks *frame*
id	Nomor identitas objek
cx, cy	Koordinat pusat *shuttlecock*
x, y, w, h	Posisi dan ukuran *bounding box*
proc_ms	Lama waktu pemrosesan (dalam milidetik)

Selain itu, hasil deteksi juga divisualisasikan secara real-time pada jendela tampilan dengan indikator FPS dan ID objek. Hal ini memudahkan pengguna untuk melihat performa sistem secara langsung.

4.2 Gambaran Umum Pengujian

Pengujian dilakukan untuk menilai kinerja sistem dari 5 aspek utama, yaitu **akurasi deteksi**, **tingkat ketepatan (*precision*), sensitivitas (*recall*), serta keseimbangan kinerja (*F1-score*)**, dan **efisiensi waktu pemrosesan**.

Dataset yang digunakan terdiri dari **26 video uji (26.658 frame)** dengan variasi kondisi:

1. Video yang hanya menampilkan *shuttlecock.*
2. Video yang menampilkan *shuttlecock* bersama objek lain seperti sepatu, jeruk, dan gelas putih.
3. Video tanpa *shuttlecock* sama sekali.

Kombinasi variasi ini dilakukan agar sistem diuji tidak hanya dalam kondisi ideal, tetapi juga pada skenario yang menyerupai keadaan nyata di lapangan. Total *frame* yang dievaluasi berjumlah **650 *frame* (25 *frame* sample per vidio)** untuk pengujian akurasi dan metrik evaluasi, serta **26.658 *frame*** untuk pengujian waktu pemrosesan.

Tujuan utama dari pengujian ini adalah untuk mengetahui sejauh mana algoritma ORB mampu mendeteksi *shuttlecock* secara konsisten dan efisien pada berbagai kondisi dan distraktor visual.

4.3 Uji Akurasi Berdasarkan Confusion Matrix

4.3.1 Konsep Akurasi

Akurasi adalah ukuran dasar yang menunjukkan seberapa sering sistem memberikan hasil deteksi yang benar dibandingkan total prediksi yang dilakukan.

Dalam penelitian ini, setiap *frame* video dievaluasi berdasarkan hasil deteksi *shuttlecock*, kemudian dibandingkan dengan **ground truth** yang ditentukan oleh ahli.

Empat kategori hasil klasifikasi yang digunakan :

- ***True Positive*** **(TP)** – *shuttlecock* ada dan berhasil terdeteksi.
- ***True Negative*** **(TN)** – *shuttlecock* tidak ada dan sistem juga tidak mendeteksi.
- ***False positive*** **(FP)** – *shuttlecock* tidak ada, tetapi sistem salah mendeteksi.
- ***False negative*** **(FN)** – *shuttlecock* ada, tetapi tidak terdeteksi oleh sistem.

Rumus akurasi :

$$Akurasi = \frac{TP + TN}{TP + TN + FP + FN} \times 100\% \qquad (4.1)$$

Sumber : (Hasnain et al., 2020)

4.3.2 Hasil Pengujian

Hasil uji akurasi diperoleh dari 26 video yang masing-masing diambil sebanyak 25 *frame* secara sampling. Berdasarkan hasil perhitungan diperoleh nilai total :

- TP = 494
- TN = 33
- FP = 84
- FN = 39

Dengan demikian, nilai akurasi keseluruhan sistem adalah :

$$Akurasi = \frac{494 + 33}{494 + 33 + 84 + 39} \times 100\% = \frac{527}{650} \times 100\% = 81,1\ \%$$

Hasil dari pengujian ini dicatat pada tabel 4.3 Hasil Uji Akurasi Deteksi *Shuttlecock*.

Tabel 4.3 Hasil Uji Akurasi Deteksi *Shuttlecock*

File	***Frame***	TP	TN	FP	FN	Akurasi
1.csv	25	25	0	0	0	100%
2.csv	25	24	0	0	1	96%
3.csv	25	24	0	0	1	96%
4.csv	25	24	0	0	1	96%
5.csv	25	21	0	0	4	84%
6.csv	25	24	0	0	1	96%
7.csv	25	19	0	0	6	76%
8.csv	25	24	0	0	1	96%
9.csv	25	22	0	0	3	88%
10.csv	25	24	0	0	1	96%
11.csv	25	24	0	0	1	96%
12.csv	25	25	0	0	0	100%
13.csv	25	25	0	0	0	100%
14.csv	25	22	0	0	3	88%
15.csv	25	0	12	13	0	48%
16.csv	25	0	6	19	0	24%
17.csv	25	24	0	0	1	96%
18.csv	25	21	0	0	4	84%
19.csv	25	21	0	0	4	84%
20.csv	25	24	0	0	1	96%
21.csv	25	24	0	0	1	96%
22.csv	25	0	11	14	0	44%
23.csv	25	13	0	11	1	52%
24.csv	25	13	2	8	2	60%
25.csv	25	14	0	10	1	56%
26.csv	25	13	2	9	1	60%
ALL	**650**	**494**	**33**	**84**	**39**	**81,1%**

4.3.3 Analisis Hasil

Hasil pengujian menunjukkan bahwa sebagian besar video memiliki tingkat akurasi yang tinggi, bahkan beberapa mencapai **100%**, terutama pada video dengan kondisi sederhana dan tanpa distraktor. Namun, beberapa video seperti **15.csv, 16.csv, dan 22.csv** memperlihatkan akurasi rendah (di bawah 50%).

Performa sistem dapat dikelompokkan sebagai berikut :

- **Kategori A (*shuttlecock* tunggal)** : akurasi rata – rata di atas 95 %.
- **Kategori B (*shuttlecock* + *distraktor*)** : akurasi berkisar 70 – 85%.
- **Kategori C (tanpa *shuttlecock*)** : cenderung menurun akibat munculnya *false positive*.

Penyebab utama penurunan akurasi adalah adanya objek pengganggu dengan bentuk dan warna mirip *shuttlecock*, seperti sepatu atau benda kecil berwarna cerah. Selain itu, motion blur pada *shuttlecock* yang bergerak cepat juga menjadi faktor penyebab *false negative*.

4.4 Uji *Precision*

4.4.1 Konsep *Precision*

Precision adalah ukuran seberapa tepat sistem dalam melakukan deteksi *shuttlecock*. Nilai ini menunjukkan proporsi deteksi yang benar (*True Positive*) terhadap seluruh deteksi positif yang dihasilkan sistem (termasuk yang salah).

Rumus *precision*:

$$Precision = \left(\frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}\right) \times 100\% \quad (4.2)$$

Sumber : (Heydarian et al., 2022)

4.4.2 Hasil pengujian *Precision*

Berdasarkan hasil evaluasi terhadap seluruh video uji, diperoleh nilai rata – rata sebagai berikut :

$$Precision = \left(\frac{494}{494 + 84}\right) \times 100\% = 85{,}4\ \%$$

Diperoleh nilai rata – rata *precision* sebesar **85,4%**. Nilai dari hasil uji *precision* ini dicatat pada tabel 4.4 Hasil Uji *Precision* Sistem.

Tabel 4.4 Hasil Uji *Precision* Sistem

File	*Frame*	TP	FP	*Precision*
1.csv	25	25	0	100%
2.csv	25	24	0	100%
3.csv	25	24	0	100%
4.csv	25	24	0	100%
5.csv	25	21	0	100%
6.csv	25	24	0	100%
7.csv	25	19	0	100%
8.csv	25	24	0	100%
9.csv	25	22	0	100%
10.csv	25	24	0	100%
11.csv	25	24	0	100%
12.csv	25	25	0	100%
13.csv	25	25	0	100%
14.csv	25	22	0	100%
15.csv	25	0	13	0%
16.csv	25	0	19	0%
17.csv	25	24	0	100%
18.csv	25	21	0	100%
19.csv	25	21	0	100%
20.csv	25	24	0	100%
21.csv	25	24	0	100%
22.csv	25	0	14	0%
23.csv	25	13	11	54,2%
24.csv	25	13	8	61,9%
25.csv	25	14	10	58,3%
26.csv	25	13	9	59%
ALL	**650**	**494**	**84**	**85,4%**

4.4.3 Analisis Hasil *Precision*

Nilai *precision* yang mencapai 85,4% menunjukkan bahwa sebagian besar deteksi yang dilakukan oleh sistem adalah benar. Namun, masih terdapat sejumlah *false positive*, terutama pada video yang mengandung objek pengganggu seperti sepatu, jeruk, gelas putih atau benda kecil lain dengan warna terang.

Kesalahan tersebut muncul karena algoritma ORB mendeteksi titik-titik fitur serupa pada objek lain yang teksturnya mirip *shuttlecock*. Hal ini merupakan

karakteristik alami ORB yang sensitif terhadap pola intensitas lokal, bukan semata bentuk keseluruhan objek.

Secara umum, nilai *precision* di atas 80% sudah menunjukkan performa yang cukup baik untuk sistem deteksi berbasis fitur, namun masih bisa ditingkatkan dengan filter spasial atau threshold pencocokan fitur yang lebih ketat.

4.5 Uji *Recall*

4.5.1 Konsep *Recall*

Recall mengukur kemampuan sistem untuk menemukan semua *shuttlecock* yang benar-benar ada pada setiap *frame* video. Nilai *recall* tinggi menandakan sistem jarang melewatkan objek yang seharusnya terdeteksi (minim *false negative*).

Rumus *recall*:

$$Recall = \left(\frac{TP}{TP + FN}\right) \times 100\% \quad (4.3)$$

Sumber : (Heydarian et al., 2022)

4.5.2 Hasil Pengujian *Recall*

Dari hasil pengujian terhadap 650 *frame* sampel, diperoleh nilai rata-rata sebagai berikut:

$$Recall = \left(\frac{494}{494 + 39}\right) \times 100\% = 92{,}6\ \%$$

Diperoleh nilai rata – rata *recall* sebesar **92,6%**. Nilai dari hasil uji *recall* ini dicatat pada tabel 4.5 Hasil Uji *Recall* Sistem.

Tabel 4.5 Hasil Uji *Recall* Sistem

File	*Frame*	TP	FN	*Recall*
1.csv	25	25	0	100%
2.csv	25	24	1	96%
3.csv	25	24	1	96%
4.csv	25	24	1	96%
5.csv	25	21	4	84%
6.csv	25	24	1	96%
7.csv	25	19	6	76%
8.csv	25	24	1	96%
9.csv	25	22	3	88%
10.csv	25	24	1	96%
11.csv	25	24	1	96%
12.csv	25	25	0	100%
13.csv	25	25	0	100%
14.csv	25	22	3	88%
15.csv	25	0	0	0%
16.csv	25	0	0	0%
17.csv	25	24	1	96%
18.csv	25	21	4	84%
19.csv	25	21	4	84%
20.csv	25	24	1	96%
21.csv	25	24	1	96%
22.csv	25	0	0	0%
23.csv	25	13	1	92,8%
24.csv	25	13	2	86,6%
25.csv	25	14	1	93,3%
26.csv	25	13	1	92,8%
ALL	**650**	**494**	**39**	**92,6%**

‘

4.5.3 Hasil Pengujian *Recall*

Nilai *recall* yang tinggi menunjukkan bahwa sistem mampu mendeteksi hampir semua *shuttlecock* yang muncul di dalam video. Ini menunjukkan bahwa algoritma ORB cukup mampu mengenali pola bentuk dan tekstur yang khas yang dimiliki *Shuttlecock*.

Namun, beberapa *frame* dengan nilai *recall* rendah disebabkan oleh *motion blur* akibat kecepatan *shuttlecock* yang tinggi, sehingga fitur tidak terdeteksi dengan baik. Selain itu, keberadaan objek atau ciri lain yang mirip dengan *shuttlecock* juga memengaruhi kemampuan algoritma dalam mencocokkan

fitur dengan template.

Nilai *recall* yang mencapai lebih dari 90% membuktikan bahwa sistem jarang gagal mendeteksi *shuttlecock*, yang berarti sistem cukup andal dalam kondisi latar dan keberadaan objek serupa yang bervariasi.

4.6 Uji *F1-score*

4.6.1 Konsep *F1-score*

F1-score digunakan untuk untuk melihat keseimbangan antara ***precision*** dan ***recall***. Nilai *F1-score* yang tinggi menunjukkan sistem mampu mendeteksi *shuttlecock* dengan tepat dan konsisten.

Rumus *F1-score* :

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4.4)$$

Sumber :(Palupi et al., 2023)

4.6.2 Hasil Pengujian *F1-score*

Berdasarkan hasil perhitungan *Precision* dan *Recall* sistem, diperoleh nilai rata-rata sebagai berikut :

$$F1 \ = \ 2 \times \frac{85{,}5\% \times 92{,}7\%}{85{,}5\% + 92{,}7\%} = 88{,}9\%$$

Diperoleh nilai rata – rata *F1-score* sebesar **88,9%**. Nilai dari hasil uji *F1-score* ini dicatat pada tabel 4.6 Hasil *F1-score* Sistem.

Tabel 4.6 Hasil *F1-score* Sistem.

File	***Precision* (%)**	***Recall* (%)**	***F1-score* (%)**
1.csv	100%	100%	100%
2.csv	100%	96%	97,9%
3.csv	100%	96%	97,9%
4.csv	100%	96%	97,9%
5.csv	100%	84%	91,3%

File	*Precision* (%)	*Recall* (%)	*F1-score* (%)
6.csv	100%	96%	97,9%
7.csv	100%	76%	86,3%
8.csv	100%	96%	97,9%
9.csv	100%	88%	93,6%
10.csv	100%	96%	97,9%
11.csv	100%	96%	97,9%
12.csv	100%	100%	100%
13.csv	100%	100%	100%
14.csv	100%	88%	93,6%
15.csv	0%	0%	0%
16.csv	0%	0%	0%
17.csv	100%	96%	97,9%
18.csv	100%	84%	91,3%
19.csv	100%	84%	91,3%
20.csv	100%	96%	97,9%
21.csv	100%	96%	97,9%
22.csv	0%	0%	0%
23.csv	54,2%	92,8%	70,7%
24.csv	61,9%	86,6%	72,1%
25.csv	58,3%	93,3%	71,7%
26.csv	59%	92,8%	72,1%
ALL	**85,4%**	**92,6%**	**88,9%**

4.6.3 Analisis Hasil *F1-score*

Nilai *F1-score* sebesar 88,9% menunjukkan bahwa sistem memiliki kinerja yang seimbang antara ketepatan dan sensitivitas. Artinya, sistem cukup andal dalam mengenali *shuttlecock* tanpa terlalu banyak kesalahan deteksi.

F1-score yang tinggi juga menegaskan bahwa kombinasi metode ORB dengan *threshold* pencocokan yang diterapkan sudah optimal, meskipun beberapa kondisi masih menimbulkan kesalahan akibat kemiripan tekstur antara *shuttlecock* dan objek lain.

Secara umum, nilai F1 di atas 85% dapat dikategorikan sangat baik untuk sistem berbasis deteksi fitur (*feature-based matching*), mengingat ORB bekerja tanpa pelatihan model dan murni mengandalkan pencocokan *deskriptor* visual.

4.7 Uji Waktu pemrosesan

4.7.1 Konsep Waktu Pemrosesan

Uji waktu pemrosesan dilakukan untuk mengukur efisiensi sistem dalam memproses citra dari setiap *frame* video, serta untuk menunjukkan kecepatan kerja algoritma deteksi dalam menyelesaikan proses identifikasi *shuttlecock*. Pengujian ini bertujuan untuk mengetahui seberapa cepat sistem dapat menjalankan serangkaian tahap pemrosesan, mulai dari ekstraksi fitur menggunakan algoritma ORB, pencocokan fitur dengan metode FLANN, hingga penentuan posisi *shuttlecock* melalui transformasi homografi.

Melalui pengukuran waktu pemrosesan, dapat diketahui performa komputasi sistem dalam kondisi yang bervariasi pada setiap video uji, seperti perbedaan sudut pandang, pencahayaan, atau jarak objek terhadap kamera. Nilai waktu pemrosesan rata-rata dihitung dalam satuan milidetik (ms) untuk setiap *frame*, sehingga dapat memberikan gambaran yang objektif mengenai efisiensi dan stabilitas algoritma yang digunakan.

Hasil pengujian ini kemudian dianalisis untuk menilai sejauh mana sistem mampu mempertahankan kecepatan pemrosesan yang konsisten pada seluruh video uji. Dengan demikian, uji waktu pemrosesan tidak hanya mengukur kinerja teknis sistem, tetapi juga menjadi indikator penting dalam mengevaluasi kelayakan algoritma ORB dan FLANN sebagai metode utama deteksi *shuttlecock* berbasis pengolahan citra video.

4.7.2 Hasil Pengujian

Dari hasil pengujian terhadap 26.658 *frame*, diperoleh:

- Waktu pemrosesan rata-rata: **20,6 ms/*frame***
- Waktu minimum: **2,35 ms/*frame***
- Waktu maksimum: **242,02 ms/*frame***
- Standar deviasi: **42,8 ms**

Hasil dari uji waktu pemrosesan dicatat pada tabel 4.7 Hasil pengujian Waktu Pemrosesan

Tabel 4.7 Hasil Pengujian Waktu Pemrosesan.

File	*Frame*	Min (ms)	Max (ms)	Rata-rata (ms)	Std (ms)
1.csv	935	4.16	195.04	16.4	34.42
2.csv	1342	8.51	231.59	25.47	46.27
3.csv	1829	4.27	194.32	17.31	35.06
4.csv	986	8.12	167.59	19.49	30.17
5.csv	1220	5.82	222.11	22.86	47.85
6.csv	1783	3.7	217.6	20.22	47.33
7.csv	707	12.08	195.01	26.19	39.69
8.csv	1831	9.22	208.32	24.49	42.47
9.csv	373	5.67	170.36	19.54	36.67
10.csv	813	7.99	198.96	22.08	39.5
11.csv	204	9.17	211.19	26.21	47.03
12.csv	1145	2.87	217.11	20.57	49.62
13.csv	1248	3.8	213.13	22.46	52.44
14.csv	1340	4.11	221.91	19.78	39.85
15.csv	958	3.73	109.68	9.76	14.28
16.csv	1032	3.6	206.07	18.82	42.87
17.csv	1015	4.4	242.02	24.5	56.14
18.csv	869	3.8	231.78	26.19	53.12
19.csv	647	5.15	190.71	19.13	39.33
20.csv	944	2.52	206.3	19.2	46.44
21.csv	957	3.99	216.41	22.46	51.21
22.csv	865	5.67	181.32	18.86	36.78
23.csv	970	4.95	182.38	17.95	36.39
24.csv	823	4.2	178.34	17.97	36.6
25.csv	896	3.96	185.81	17.47	33.38
26.csv	926	2.35	216.08	21.73	43.13
ALL	26.658	2.35	242.02	20.6	42.8

4.7.3 Analisis Hasil

Hasil pengujian waktu pemrosesan terhadap 26.658 *frame* menunjukkan adanya variasi waktu yang cukup besar antar video. Berdasarkan Tabel 4.7, waktu pemrosesan rata-rata sistem mencapai **20,6 ms per *frame***, dengan waktu minimum **2,35 ms** dan maksimum **242,02 ms**, serta standar deviasi sebesar **42,8 ms**. Nilai-nilai tersebut menunjukkan bahwa kecepatan sistem dalam memproses *frame* tidak selalu konstan, melainkan dipengaruhi oleh kondisi visual pada setiap video.

Variasi waktu pemrosesan ini dapat diamati dari perbandingan antar file video. Misalnya, **file 15.csv** memiliki waktu rata-rata tercepat sebesar **9,76 ms/*frame*** dengan standar deviasi kecil (**14,28 ms**), sedangkan **file 17.csv** menunjukkan waktu pemrosesan tertinggi sebesar **24,5 ms/*frame*** dengan standar deviasi yang lebih besar (**56,14 ms**). Perbedaan ini mengindikasikan bahwa kompleksitas visual pada setiap video sangat berpengaruh terhadap lamanya waktu pemrosesan yang dibutuhkan oleh sistem.

Video dengan latar belakang sederhana, dan sedikit objek pengganggu (distraktor) cenderung memiliki waktu pemrosesan lebih cepat. Hal ini disebabkan oleh jumlah titik fitur yang terdeteksi lebih sedikit, sehingga proses ekstraksi dan pencocokan fitur menjadi lebih ringan. Sebaliknya, pada video dengan latar kompleks, bayangan kuat, atau banyak distraktor, algoritma ORB perlu mengekstraksi dan mencocokkan lebih banyak titik fitur, yang menyebabkan waktu pemrosesan meningkat.

Secara teori, hasil ini konsisten dengan karakteristik algoritma **ORB**

(*Oriented FAST and Rotated BRIEF*) yang menggabungkan dua proses utama: deteksi fitur menggunakan **FAST** dan deskripsi fitur menggunakan **BRIEF**. FAST mendeteksi *keypoint* berdasarkan intensitas *piksel* pada area bertekstur tinggi, sementara BRIEF membentuk *deskriptor* biner dari setiap *keypoint* yang ditemukan. Semakin banyak titik fitur yang muncul pada *frame*, semakin banyak pula *deskriptor* yang harus dihitung dan dicocokkan, sehingga menambah beban komputasi.

Nilai **standar deviasi (Std)** pada tabel menunjukkan tingkat kestabilan sistem dalam memproses setiap *frame*. Standar deviasi yang kecil, seperti pada 15.csv, menandakan sistem bekerja secara konsisten dengan variasi waktu yang rendah antar *frame*. Sebaliknya, nilai deviasi yang besar pada video seperti 17.csv menunjukkan adanya fluktuasi waktu pemrosesan yang signifikan akibat perubahan kompleksitas visual dari satu *frame* ke *frame* berikutnya, seperti kemunculan objek tambahan, perubahan pencahayaan, atau efek motion blur

Secara keseluruhan, hasil pengujian ini menegaskan bahwa waktu pemrosesan sistem deteksi *shuttlecock* berbasis ORB lebih dipengaruhi oleh kompleksitas visual citra dan jumlah fitur yang terdeteksi, bukan oleh keberadaan *shuttlecock* itu sendiri. Dengan rata-rata waktu pemrosesan 20,6 ms/*frame*, sistem ini masih berada dalam kategori efisien.

4.8 Pembahasan

Berdasarkan hasil implementasi dan pengujian sistem deteksi *shuttlecock* menggunakan algoritma ORB (*Oriented FAST and Rotated BRIEF*), diperoleh bahwa sistem mampu mendeteksi obyek *shuttlecock* di antara benda-benda lain

yang ada di lapangan secara efisien dan akurat. Hasil uji menunjukkan nilai akurasi rata-rata sebesar 81,1%, *precision* sebesar 85,4%, *recall* sebesar 92,6%, dan *F1-score* sebesar 88,9%, sedangkan rata-rata waktu pemrosesan per frame mencapai 20,6 ms. Nilai akurasi yang berada pada angka 81,1% ini dipengaruhi oleh adanya beberapa video uji—seperti 15.csv, 16.csv, dan 22.csv—yang sebenarnya tidak mengandung objek *shuttlecock*, namun algoritma ORB tetap mengekstraksi fitur secara kontinu pada setiap frame. Karena sifat ORB yang akan terus mencari dan mencocokkan pola fitur yang mirip dengan referensi, sistem sesekali mendeteksi objek lain yang memiliki kemiripan tekstur atau pola dengan *shuttlecock*. Kondisi ini memunculkan deteksi positif palsu (false positive), sehingga akurasi keseluruhan mengalami penurunan.

Kemampuan sistem dalam mencapai hasil tersebut menunjukkan bahwa algoritma ORB efektif digunakan untuk mengenali *shuttlecock* berdasarkan fitur visual yang unik, seperti bentuk kepala bulat dan pola bulu kerucut. ORB mampu mempertahankan stabilitas deteksi meskipun *shuttlecock* mengalami rotasi atau perbedaan sudut pandang kamera. Hal ini membuktikan bahwa tujuan utama penelitian yaitu mendeteksi obyek *shuttlecock* di antara benda lain dengan algoritma ORB telah tercapai dengan baik.

Selain itu, kombinasi antara metode ORB dan FLANN dalam pencocokan fitur serta pelacakan berbasis KCF dengan penyaringan EMA memberikan hasil deteksi yang konsisten dan stabil. Meskipun beberapa kesalahan masih terjadi akibat kemiripan tekstur atau warna antara *shuttlecock* dan objek sekitarnya, sistem secara umum mampu mempertahankan performa yang tinggi. Hal ini

menunjukkan bahwa algoritma ORB memiliki keunggulan dalam mengenali objek berukuran kecil dengan bentuk kompleks dalam lingkungan yang tidak seragam seperti lapangan bulu tangkis.

Dari sisi praktis, sistem ini memberikan kontribusi signifikan dalam meningkatkan efisiensi kegiatan latihan bulu tangkis. Proses identifikasi *shuttlecock* yang sebelumnya dilakukan secara manual kini dapat dilakukan secara otomatis, sehingga waktu latihan dapat dimanfaatkan lebih efektif. Dengan demikian, penelitian ini bukan hanya memenuhi tujuan teknisnya, tetapi juga memberikan nilai tambah dalam bentuk kemudahan dan efisiensi bagi pengguna di lapangan.

Hasil penelitian ini juga selaras dengan prinsip pemanfaatan ilmu pengetahuan untuk kemaslahatan manusia. Dalam pandangan Islam, ilmu merupakan anugerah yang harus digunakan untuk membedakan antara yang benar dan yang salah serta diarahkan pada kebaikan. Hal ini sebagaimana dijelaskan dalam *QS. Al-Baqarah [2]: 269*:

يُؤْتِى الْحِكْمَةَ مَنْ يَّشَاۤءُۚ وَمَنْ يُّؤْتَ الْحِكْمَةَ فَقَدْ اُوْتِيَ خَيْرًا كَثِيْرًاۗ وَمَا يَذَّكَّرُ اِلَّآ اُولُوا الْاَلْبَابِ ﴿٢٦٩﴾

"Dia (Allah) menganugerahkan hikmah kepada siapa yang Dia kehendaki. Siapa yang dianugerahi hikmah, sungguh dia telah dianugerahi kebaikan yang banyak. Tidak ada yang dapat mengambil pelajaran (darinya), kecuali ululalbab." (*QS. Al-Baqarah [2]: 269)*

yang menurut tafsir tahlili (Tahlili, 2011), menggambarkan bahwa Allah mengaruniakan hikmah kepada siapa yang dikehendaki-Nya, yaitu kemampuan memahami kebenaran berdasarkan dalil, bukti, dan akal yang sehat. Hikmah tersebut memungkinkan seseorang membedakan petunjuk Allah dari bisikan

setan, serta mengarahkan dirinya pada kebaikan dunia dan akhirat. Dalam konteks penelitian ini, kemampuan manusia memanfaatkan akal dan pengetahuan untuk mengembangkan teknologi deteksi *shuttlecock* merupakan salah satu bentuk implementasi hikmah tersebut.

Penelitian ini juga berkaitan dengan pemanfaatan waktu secara efektif. Dalam *QS. Al-'Ashr [103]: 1–3:*

وَالْعَصْرِۙ ﴿١﴾ اِنَّ الْاِنْسَانَ لَفِيْ خُسْرٍۙ ﴿٢﴾ اِلَّا الَّذِيْنَ اٰمَنُوْا وَعَمِلُوا الصّٰلِحٰتِ وَتَوَاصَوْا بِالْحَقِّ ەۙ وَتَوَاصَوْا بِالصَّبْرِ ﴿٣﴾

"Demi masa. Sesungguhnya manusia itu benar-benar berada dalam kerugian, kecuali orang-orang yang beriman dan beramal saleh serta saling menasihati untuk kebenaran dan kesabaran." (QS. Al-'Ashr [103]: 1-3)

menurut tafsir tahlili (Tahlili, 2011) Allah bersumpah dengan masa (al-'ashr) sebagai tanda kekuasaan dan hikmah-Nya. Perubahan waktu, pergantian siang dan malam, serta perjalanan usia manusia merupakan bukti kebesaran Allah. Ayat tersebut mengingatkan bahwa manusia berada dalam kerugian apabila tidak memanfaatkan waktu dengan baik. Hanya mereka yang beriman, beramal saleh, dan saling menasihati dalam kebenaran dan kesabaran yang dapat terhindar dari kerugian tersebut. Sistem yang dikembangkan dalam penelitian ini mampu meningkatkan efisiensi waktu latihan, sehingga pengguna dapat memanfaatkan waktu secara lebih produktif dan terarah, sesuai dengan pesan ayat ini.

Selanjutnya, *QS. Al-Qashash [28]: 77:*

وَابْتَغِ فِيْمَآ اٰتٰىكَ اللّٰهُ الدَّارَ الْاٰخِرَةَ وَلَا تَنْسَ نَصِيْبَكَ مِنَ الدُّنْيَا وَاَحْسِنْ كَمَآ اَحْسَنَ اللّٰهُ اِلَيْكَ وَلَا تَبْغِ الْفَسَادَ فِى الْاَرْضِۗ اِنَّ اللّٰهَ لَا يُحِبُّ الْمُفْسِدِيْنَ ﴿٧٧﴾

"Dan, carilah pada apa yang telah dianugerahkan Allah kepadamu (pahala)

negeri akhirat, tetapi janganlah kamu lupakan bagianmu di dunia. Berbuat baiklah (kepada orang lain) sebagaimana Allah telah berbuat baik kepadamu dan janganlah kamu berbuat kerusakan di bumi. Sesungguhnya Allah tidak menyukai orang-orang yang berbuat kerusakan." (QS. Al-Qashash [28]: 77)

dalam tafsir tahlili (Tahlili, 2011) menjelaskan empat nasihat yang diberikan kepada Qarun, yaitu agar memanfaatkan nikmat Allah untuk kebaikan, tidak melupakan bagian dunia yang halal, berbuat baik kepada sesama, dan menjauhi segala bentuk kerusakan. Ayat tersebut menegaskan bahwa segala bentuk usaha dan inovasi manusia harus diarahkan pada kemaslahatan, bukan kerusakan. Dalam konteks penelitian ini, pengembangan sistem deteksi *shuttlecock* merupakan bentuk pemanfaatan ilmu yang diarahkan untuk kebaikan, memudahkan pekerjaan manusia, tidak menyebabkan kerusakan, dan meningkatkan keteraturan dalam aktivitas olahraga.

penelitian ini juga sejalan dengan perspektif Maqāṣid al-Syarī'ah, yaitu tujuan-tujuan utama syariat dalam menghadirkan kemaslahatan bagi manusia. Dalam konteks ini, pengembangan sistem deteksi *shuttlecock* berbasis algoritma ORB dapat dipandang sebagai bentuk kontribusi teknologi yang mendukung tercapainya nilai-nilai kemaslahatan tersebut.

Pertama, konsep **hifẓ al-'aql (menjaga akal)** tercermin dari proses penelitian ini yang memanfaatkan kemampuan berpikir, analisis data, dan inovasi teknologi. Pengembangan sistem deteksi otomatis menggunakan ORB merupakan wujud optimalisasi akal untuk menghasilkan teknologi yang membawa manfaat bagi manusia.

Kedua, nilai **hifẓ al-māl (menjaga harta)** tampak pada efisiensi yang dihasilkan sistem ini. Dengan mengotomatiskan proses identifikasi *shuttlecock*, waktu latihan menjadi lebih efektif dan penggunaan sumber daya menjadi lebih optimal. Teknologi ORB yang ringan dan ekonomis juga mendukung prinsip penghematan biaya tanpa mengurangi kinerja.

Ketiga, penelitian ini dapat dikaitkan dengan **hifẓ al-dīn (menjaga agama)**, karena pemanfaatan ilmu pengetahuan yang bertujuan untuk kebaikan merupakan bagian dari etika Islam. Selama teknologi digunakan untuk mempermudah aktivitas manusia dan tidak menimbulkan mudarat, maka upaya ini termasuk amal yang bernilai ibadah dan selaras dengan nilai syariat.

Dengan demikian, pemanfaatan algoritma ORB dalam mendeteksi *shuttlecock* tidak hanya unggul dalam aspek teknis, tetapi juga mencerminkan nilai-nilai maqāṣid yang harmonis dengan tujuan syariat, yaitu menghadirkan kemudahan, efisiensi, keselamatan, dan kemaslahatan bagi manusia.

Secara keseluruhan, penelitian ini menegaskan bahwa kemajuan teknologi dapat berjalan harmonis dengan nilai-nilai spiritual. Ilmu pengetahuan yang digunakan dengan niat baik dan tujuan bermanfaat bukan hanya menghasilkan inovasi yang cerdas secara teknis, tetapi juga bernilai ibadah di sisi Allah SWT. Dengan demikian, sistem deteksi *shuttlecock* berbasis algoritma ORB ini dapat dipandang sebagai salah satu bentuk implementasi ilmu yang berkeadaban, yaitu ilmu yang tidak hanya menekankan aspek intelektual, tetapi juga menjunjung tinggi moral, kemaslahatan, dan tanggung jawab sosial sesuai ajaran Islam.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil penelitian dan pembahasan yang telah dilakukan, dapat disimpulkan bahwa tujuan penelitian untuk mendeteksi objek *shuttlecock* di antara benda-benda lain di lapangan dengan menggunakan algoritma ORB (*Oriented FAST and Rotated BRIEF*) berhasil dicapai dengan baik. Sistem yang dikembangkan mampu mengenali *shuttlecock* secara akurat dengan rata-rata akurasi 81,1%, *precision* 85,4%, *recall* 92,6%, dan *F1-score* 88,9%, serta waktu pemrosesan rata-rata 20,6 ms per frame. Kemampuan deteksi yang stabil dan efisien tersebut didukung oleh integrasi metode pencocokan fitur FLANN serta pelacakan berbasis KCF yang dikombinasikan dengan penyaringan *Exponential Moving Average* (EMA) untuk mempertahankan konsistensi deteksi antar-frame. Temuan ini menunjukkan bahwa ORB memiliki kinerja yang andal dalam mengenali *shuttlecock* berdasarkan fitur visual khasnya meskipun berada pada lingkungan yang dinamis dan kompleks, sehingga penelitian ini membuktikan bahwa sistem deteksi berbasis visi komputer dapat diterapkan secara efektif dalam mendukung aktivitas analisis dan evaluasi pada olahraga bulu tangkis.

5.2 Saran

Berdasarkan hasil penelitian dan pengujian sistem yang telah dilakukan, terdapat beberapa hal yang perlu diperhatikan untuk pengembangan lebih lanjut agar sistem ini dapat lebih optimal dan bermanfaat bagi pengguna yang lebih luas. Adapun saran-saran yang dapat diajukan sebagai berikut:

1. **Penambahan Pengujian Video dan Variasi Bentuk *Shuttlecock***

   Disarankan untuk melakukan pengujian menggunakan video uji yang lebih beragam, terutama pada kondisi di mana bentuk *shuttlecock* berubah akibat pukulan keras atau deformasi fisik. Hal ini penting karena perubahan bentuk dapat memengaruhi fitur yang diekstraksi oleh algoritma ORB, sehingga hasil pengujian akan memberikan gambaran yang lebih akurat terhadap ketahanan sistem pada kondisi nyata di lapangan.

2. **Perbandingan dengan Metode Ekstraksi Fitur Lain**

   Untuk meningkatkan akurasi dan reliabilitas sistem, disarankan melakukan perbandingan kinerja algoritma ORB dengan metode ekstraksi fitur lainnya seperti SIFT (*Scale-Invariant Feature Transform*), SURF (*Speeded-Up Robust Features*), atau BRISK (*Binary Robust Invariant Scalable Keypoints*). Perbandingan ini bertujuan untuk menilai tingkat efisiensi, kecepatan, serta sensitivitas terhadap perubahan pencahayaan dan rotasi objek.

3. **Peningkatan Kinerja Sistem Melalui Optimasi Parameter**

   Penelitian selanjutnya dapat difokuskan pada optimasi parameter ORB dan FLANN, seperti jumlah *keypoint*, *threshold* FAST, dan rasio jarak pencocokan, guna memperoleh keseimbangan terbaik antara kecepatan dan akurasi deteksi. Langkah ini akan membantu sistem beradaptasi lebih baik terhadap variasi lingkungan dan perbedaan jarak kamera.

4. Integrasi Sistem dengan Aplikasi Pelatihan Otomatis

Pengembangan lanjutan dapat diarahkan pada pembuatan aplikasi pendukung latihan bulu tangkis yang mengintegrasikan hasil deteksi *shuttlecock* dengan analisis visual, seperti pelacakan lintasan, atau perhitungan kecepatan. Dengan demikian, sistem ini dapat dimanfaatkan langsung oleh pelatih maupun atlet untuk evaluasi performa secara objektif.

Dengan adanya saran-saran tersebut, diharapkan penelitian ini dapat menjadi dasar untuk pengembangan sistem pengolahan citra yang lebih adaptif, akurat, dan efisien, serta dapat diterapkan secara luas dalam bidang olahraga maupun aplikasi deteksi objek lain yang memerlukan ketelitian tinggi.

.

DAFTAR PUSTAKA

Adhinata, F. D., Harjoko, A., & Wahyono. (2021). Object Searching on Real-Time Video Using Oriented FAST and Rotated BRIEF Algorithm. *International Journal on Advanced Science, Engineering and Information Technology*, *11*(6), 2518–2526. https://doi.org/10.18517/ijaseit.11.6.12043

Aglave, P., & Kolkure, V. S. (2015). Implementation of High Performance Feature Extraction Method Using Oriented Fast and Rotated Brief Algorithm. *International Journal of Research in Engineering and Technology*, *04*(02), 394–397. https://doi.org/10.15623/ijret.2015.0402052

Bao, J., Yuan, X., & Lam, C. T. (2022). Robust Image Matching for Camera Pose Estimation Using Oriented Fast and Rotated Brief. *ACM International Conference Proceeding Series*, *July 2024*. https://doi.org/10.1145/3579654.3579720

Chen, J., Xi, Z., Wei, C., Lu, J., Niu, Y., & Li, Z. (2021). Multiple Object Tracking Using Edge Multi-Channel Gradient Model with ORB Feature. *IEEE Access*, *9*, 2294–2309. https://doi.org/10.1109/ACCESS.2020.3046763

Corresp, M. I. (2020). *Computer Science Manuscript to be reviewed Comprehensive Evaluation of Feature Extractors in Computer Manuscript to be reviewed Comprehensive Evaluation of Feature Extractors in Computer Vision : An Empirical Analysis*.

Dai, Y., & Wu, J. (2023). An Improved ORB Feature Extraction Algorithm Based on Enhanced Image and Truncated Adaptive Threshold. *IEEE Access*, *11*(April), 32073–32081. https://doi.org/10.1109/ACCESS.2023.3261665

De Alwis, A. P. G., Dehikumbura, C., Konthawardana, M., Lalitharatne, T. D., & Dassanayake, V. P. C. (2020). Design and Development of a Badminton Shuttlecock Feeding Machine to Reproduce Actual Badminton Shots. *2020 5th International Conference on Control and Robotics Engineering, ICCRE 2020*, 73–77. https://doi.org/10.1109/ICCRE49379.2020.9096444

Forero, M. G., Mambuscay, C. L., Monroy, M. F., Miranda, S. L., Méndez, D., Valencia, M. O., & Selvaraj, M. G. (2021). Comparative Analysis of Detectors and Feature Descriptors for Multispectral Image Matching in Rice Crops. *Plants*, *10*(9), 1–24. https://doi.org/10.3390/plants10091791

Halla, M. A., Djahi, H. J., Tulit Ina, W., & Tena, S. (2025). Sistem Deteksi Nominal Uang Kertas Untuk Penyandang Tunanetra Berbasis Kamera Dengan Output Suara. *Jurnal SINTA: Sistem Informasi Dan Teknologi Komputasi*, *2*(1), 23–30. https://doi.org/10.61124/sinta.v2i1.36

Hapsari, D. A. P., Nofa, W. K., & Santoso, S. (2022). Analisis Performa Deteksi Objek Bergerak pada Algoritma Background Subtraction dan Algoritma Frame Difference. *ICIT Journal*, *8*(1), 98–107. https://doi.org/10.33050/icit.v8i1.2177

Hasnain, M., Pasha, M. F., Ghani, I., Imran, M., Alzahrani, M. Y., & Budiarto, R. (2020). Evaluating Trust Prediction and Confusion Matrix Measures for Web Services Ranking. *IEEE Access*, *8*, 90847–90861. https://doi.org/10.1109/ACCESS.2020.2994222

Heydarian, M., Doyle, T. E., & Samavi, R. (2022). MLCM: Multi-Label Confusion Matrix. *IEEE Access*, *10*, 19083–19095. https://doi.org/10.1109/ACCESS.2022.3151048

Ibrahim, N. M., ElFarag, A. A., & Kadry, R. (2021). Gaussian Blur through Parallel Computing. *Proceedings of the International Conference on Image Processing and Vision Engineering, IMPROVE 2021*, *Improve*, 175–179. https://doi.org/10.5220/0010513301750179

Imsaengsuk, T., & Pumrin, S. (2021). Feature Detection and Description based on ORB Algorithm for FPGA-based Image Processing. *Proceeding of the 2021 9th International Electrical Engineering Congress, IEECON 2021*, 420–423. https://doi.org/10.1109/iEECON51072.2021.9440232

Khotimah, I. anggraeni K., & Rahmandika, M. B. (2020). Identifikasi Potensi Bahaya K3 Menggunakan Metode Failure Mode Effect Analysis Dan Usulan Pencegahan Di Ukm Power Shuttlecock. *Journal of Industrial View*, *2*(2), 12–19. https://doi.org/10.26905/4937

Khudhair, Z. N., Khdiar, A. N., El Abbadi, N. K., Mohamed, F., Saba, T., Alamri, F. S., & Rehman, A. (2023). Color to Grayscale Image Conversion Based on Singular Value Decomposition. *IEEE Access*, *11*(June), 54629–54638. https://doi.org/10.1109/ACCESS.2023.3279734

Liu, H., Tan, T. H., & Kuo, T. Y. (2020). A Novel Shot Detection Approach Based on ORB Fused with Structural Similarity. *IEEE Access*, *8*, 2472–2481. https://doi.org/10.1109/ACCESS.2019.2962328

Liu, X., Zhang, Y., & Wang, Q. (2025). Posture Detection of Dual-Hemisphere Capsule Robot Based on Magnetic Tracking Effects and ORB-AEKF Algorithm. *Micromachines*, *16*(4). https://doi.org/10.3390/mi16040485

Lowe, D. G. (2004). *Distinctive Image Features from Scale-Invariant Keypoints*. 1–28.

Ma, C., Hu, X., Xiao, J., & Zhang, G. (2021). Homogenized ORB Algorithm Using Dynamic Threshold and Improved Quadtree. *Mathematical Problems*

in Engineering, *2021*. https://doi.org/10.1155/2021/6693627

Mahmudin, Saraswati, N. M., & Sorikhi. (2023). Object Tracking Berbasis Keypoint Menggunakan Algoritma Orb (Oriented Fast and Rotated Brief) Pada Raspberry Pi. *Jurnal Informatika Dan Riset*, *1*(2), 1–8. https://doi.org/10.36308/iris.v1i2.505

Md. Shabbir, K. S., Ahmed, M. I., & Marzan Alam. (2021). Detection of Glaucoma using ORB (Oriented FAST and Rotated BRIEF) Feature Extraction. *Journal of Engineering Advancements*, *02*(03), 153–158. https://doi.org/10.38032/jea.2021.03.005

Oliveira, A. J., Ferreira, B. M., & Cruz, N. A. (2021). A Performance Analysis of Feature Extraction Algorithms for Acoustic Image-Based Underwater Navigation. *Journal of Marine Science and Engineering*, *9*(4). https://doi.org/10.3390/jmse9040361

Palupi, L., Ihsanto, E., & Nugroho, F. (2023). Analisis Validasi dan Evaluasi Model Deteksi Objek Varian Jahe Menggunakan Algoritma Yolov5. *Journal of Information System Research (JOSH)*, *5*(1), 234–241. https://doi.org/10.47065/josh.v5i1.4380

Pardiwala, D. N., Subbiah, K., Rao, N., & Modi, R. (2020). Badminton Injuries in Elite Athletes: A Review of Epidemiology and Biomechanics. *Indian Journal of Orthopaedics*, *54*(3), 237–245. https://doi.org/10.1007/s43465-020-00054-1

Ramadhan, R., & Muzaffar, A. (2023). Pengembangan Alat Bantu Latihan Ketepatan Smash Bulutangkis Mahasiswa JPOK Universitas Jambi. *Jurnal Pion*, *3*(1), 27–33.

Rublee, E., & Bradski, G. (2014). *ORB : an efficient alternative to SIFT or SURF ORB : an efficient alternative to SIFT or SURF*. *May*. https://doi.org/10.1109/ICCV.2011.6126544

Singh, G., Goyal, D., & Kumar, V. (2025). Mobile Robot Localization Using Visual Odometry in Indoor Environments with TurtleBot4. *IAES International Journal of Artificial Intelligence*, *14*(1), 760–768. https://doi.org/10.11591/ijai.v14.i1.pp760-768

Sun, R., Qian, J., Jose, R. H., Gong, Z., Miao, R., Xue, W., & Liu, P. (2020). A Flexible and Efficient Real-Time ORB-Based Full-HD Image Feature Extraction Accelerator. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *28*(2), 565–575. https://doi.org/10.1109/TVLSI.2019.2945982

Susim, T., & Darujati, C. (2021). Pengolahan Citra untuk Pengenalan Wajah

(Face Recognition) Menggunakan OpenCV. *Jurnal Syntax Admiration*, *2*(3), 534–545. https://doi.org/10.46799/jsa.v2i3.202

Tahlili, L. P. M. (2011). Al-Qur′an dan Tafsirnya Jilid 2 (Juz 4 - 6). In *Widya Cahaya, Jakarta*.

Tilaksono, H., Kindhi, B., & Istiqomah, F. (2022). Deteksi Citra dan Posisi Barcode Menggunakan Metode Oriented Fast and Rotated Brief (ORB) dan Maximally Stable Extremal Regions (MSER). *Jurnal Teknik ITS*, *11*(2). https://doi.org/10.12962/j23373539.v11i2.85214

Wu, K. (2023). Creating Panoramic Images Using ORB Feature Detection and RANSAC-based Image Alignment. *Advances in Computer and Communication*, *4*(4), 220–224. https://doi.org/10.26855/acc.2023.08.002

Yan, H., Wang, J., & Zhang, P. (2023). Application of Optimized ORB Algorithm in Design AR Augmented Reality Technology Based on Visualization. *Mathematics*, *11*(6). https://doi.org/10.3390/math11061278

Yao, C., Zhang, H., Zhu, J., Fan, D., Fang, Y., & Tang, L. (2023). ORB Feature Matching Algorithm Based on Multi-Scale Feature Description Fusion and Feature Point Mapping Error Correction. *IEEE Access*, *11*(August), 63808–63820. https://doi.org/10.1109/ACCESS.2023.3288594