



UNIVERSITY
IYUNIVESITHI
UNIVERSITEIT

forward together
sonke siya phambili
saam vorentoe

MECHATRONIC PROJECT 478 FINAL REPORT

Combining object tracking algorithms with probabilistic graphical models for accurate ball localization

Author:
Alex Thorpe

Student Number:
25934171

October 6, 2025

Supervisor: Dr. C. E. Van Daalen

Plagiarism Declaration

I have read and understand the Stellenbosch University Policy on Plagiarism and the definitions of plagiarism and self-plagiarism contained in the Policy [Plagiarism: The use of the ideas or material of others without acknowledgement, or the re-use of one's own previously evaluated or published material without acknowledgement or indication thereof (self-plagiarism or text-recycling)].

I also understand that direct translations are plagiarism, unless accompanied by an appropriate acknowledgement of the source. I also know that verbatim copy that has not been explicitly indicated as such, is plagiarism.

I know that plagiarism is a punishable offence and may be referred to the University's Central Disciplinary Committee (CDC) who has the authority to expel me for such an offence.

I know that plagiarism is harmful for the academic environment and that it has a negative impact on any profession.

Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully (acknowledged); further, all verbatim copies have been expressly indicated as such (e.g. through quotation marks) and the sources are cited fully.

I declare that, except where a source has been cited, the work contained in this assignment is my own work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment. I declare that have not allowed, and will not allow, anyone to use my work (in paper, graphics, electronic, verbal or any other format) with the intention of passing it off as his/her own work.

I know that a mark of zero may be awarded to assignments with plagiarism and also that no opportunity be given to submit an improved assignment.



Signature

Alex Thorpe

Initials and surname

25934171

Student number

October 6, 2025

Date

Executive Summary

Title of Project
Combining Object Tracking Algorithms with Probabilistic Graphical Models for Accurate Ball Localization
Objectives
This project aims to estimate a ball's position in real time using standard video footage. Multiple object tracking algorithms will be applied, and their outputs fused using a probabilistic graphical model (PGM) to enhance accuracy, robustness, and responsiveness. The final goal is to evaluate the PGM-based fusion model against individual trackers.
What is current practice and what are its limitations?
Hawk-Eye and Tracab are industry leaders in ball tracking and sports officiating. Hawk-Eye supports over 20 sports with real-time tracking and visualization, while Tracab focuses on football, analyzing player movement and ball trajectory (Hawk-Eye Innovations 2024; Tracab 2024). Despite their accuracy, both systems require complex, expensive multi-camera setups and proprietary infrastructure, limiting their use in smaller-scale, research, or open-source applications.
What is new in this project?
This project introduces an accurate, low-complexity ball tracking system that requires minimal camera infrastructure, making it accessible for research, amateur sports, and low-budget environments.
If the project is successful, how will it make a difference?
If successful, this project could enhance decision-making across various sports by accurately determining ball location in real time — such as confirming goals or out-of-bounds events — making advanced tracking more accessible beyond elite-level systems.
What are the risks to the project being a success? Why is it expected to be successful?
A key risk is that the system may not match the efficiency or accuracy of existing commercial trackers, limiting its ability to fully replace them. However, by studying and building on established tracking methods, this project is expected to deliver a more lightweight and accessible alternative — one that balances performance with lower complexity and cost.
What contributions have/will other students made/make?
N/A
Which aspects of the project will carry on after completion and why?
After completion, the project could be extended by incorporating sport-specific rules, enabling the system to automatically detect rule violations such as offside positions, fouls, or out-of-bounds events. This would build on the core tracking functionality and move the system toward real-time decision support and automated officiating.

What arrangements have been/will be made to expedite continuation?
All components of the project will be thoroughly documented, with clear definitions of the system architecture, algorithms, and data handling. This will ensure the project can be easily continued by future students or refined further for potential commercialization.

Evaluation of ECSA Exit Level Outcomes

ECSA Outcomes Assessed in this Module	
ECSA Outcome	Addressed
GA 1. Problem solving: Demonstrate competence to identify, assess, formulate and solve convergent and divergent engineering problems creatively and innovatively.	Throughout the literature review process and figuring out the algorithms.
GA 2. Application of scientific and engineering knowledge: Demonstrate competence to apply knowledge of mathematics, basic science and engineering sciences from first principles to solve engineering problems.	During the experimentation and testing portion of the project and when applying the algorithms.
GA 3. Engineering Design: Demonstrate competence to perform creative, procedural and non-procedural design and synthesis of components, systems, engineering works, products or processes.	During the concept design and algorithm decision.
GA 5. Engineering methods, skills and tools, including Information Technology: Demonstrate competence to use appropriate engineering methods, skills and tools, including those based on information technology.	Throughout the project the use of different coding methods will be used and learning new skills such as PGMs.
GA 6. Professional and technical communication: Demonstrate competence to communicate effectively, both orally and in writing, with engineering audiences and the community at large.	Project Proposal; Progress Report; Preliminary Draft; Final Report; Oral Presentation; Project Poster.

<p>GA 8. Individual, Team and Multidisciplinary Working: Demonstrate competence to work effectively as an individual, in teams and in multidisciplinary environments.</p>	<p>This project shows individual work very clearly.</p>
<p>GA 9. Independent Learning Ability: Demonstrate competence to engage in independent learning through well-developed learning skills.</p>	<p>Will have to self-learn many different complex topics.</p>

Contents

Plagiarism Declaration	i
Executive Summary	ii
Evaluation of ECSA Exit Level Outcomes	iv
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Report Overview	3
2 Background and Literature Review	4
2.1 Existing Commercial Tracking systems	4
2.2 Computer Vision techniques for tracking	5
2.2.1 Traditional Tracking and Detecting methods	5
2.2.2 Modern Deep Learning-Based Tracking	6
2.3 Image Processing Algorithms	6
2.3.1 Low-Level Vision	7
2.3.2 Intermediate Level Vision	7
2.3.3 OpenCV	7
2.3.4 OpenCV Functions for Low- and Intermediate-Level Vision	8
2.4 Probabilistic Graphical Models	11
2.4.1 Modeling	12
2.4.2 Inference	13
2.4.3 Gaussian Factors	15
2.4.4 Hybrid Factors	16
2.4.5 Expectation Propagation (EP)	18
2.5 Gaps	19
3 Model Design	20
3.1 Video Footage Description	20
3.2 Image Processing Algorithms	21

3.2.1	Requirements	21
3.2.2	Algorithm Design	22
3.2.3	Candidate Algorithms	23
3.3	Probabilistic Graphical Model	26
3.3.1	Design Requirements	27
3.3.2	Model Construction	28
3.3.3	PGM Design	31
4	Implementation	33
5	Experimentation	34
6	Conclusion	35

List of Figures

1	Examples of graphical models	12
2	Conversion of Bayesian Network to Markov Network	14
3	Example of a Conditional Linear Gaussian network showing discrete RV X and continuous RVs Y and Z	17
4	Overview of the model design	20
5	Flow diagram showing the flow of logic each of the IPA will follow	23
6	Relationship between $Pos_{x,y}$, Alg_n and a_n for single algorithm and time step	29
7	Bayesian Network showing the relationships between $Pos_{x,y}$ and the various algorithms used	30
8	Bayesian Network showing the relationships between $Pos_{x,y}$, the various IPA used and the discrete Outlier Indicators at the different time steps	30
9	Flow diagram showing the flow of logic the PGM will follow	32

List of Tables

1	Low-Level Vision Functions from OpenCV	8
2	Intermediate-Level Vision Functions from OpenCV	10
3	Requirements for Image Processing Algorithms	21
4	Candidate Image Processing Algorithms	24
5	Requirements for the Probabilistic Graphical Model	27
6	Random Variables for the Probabilistic Graphical Model	28

List of Abbreviations

ADF Assumed Density Filtering

BN Bayesian Network

BP Belief Propagation

BU Belief Update

CLG Conditional Linear Gaussian

CNN Convolution Neural Network

CPD Conditional Probability Distribution

CV Computer Vision

DL Deep Learning

EP Expectation Propagation

FPS Frames Per Second

ML Machine Learning

MN Markov Network

PGM Probabilistic Graphical Model

RV Random Variable

YOLO You Only Look Once

List of Symbols

$P(A)$ Probability of event A occurring

$P(A \mid B)$ Conditional probability of event A given event B

$P(X_1, \dots, X_n)$ Joint probability distribution over variables X_1 through X_n

Σ Summation operation

1 Introduction

The growing importance of data-driven decision-making in sports has accelerated the adoption of advanced technologies such as Machine Learning (ML), computer vision, and real-time analytics. In particular, ball tracking has become a critical component in modern sports for enhancing player performance analysis, enabling data-supported officiating, and facilitating targeted training interventions. High-profile systems such as Hawk-Eye and TRACAB have become industry standards, Hawk-Eye being widely used for adjudication in tennis and cricket (Hawk-Eye Innovations 2024), while TRACAB supports tactical and performance analysis in football and other team sports (Tracab 2024).

Despite their effectiveness, these systems depend on dense camera arrays and high-performance computing infrastructure installed around stadiums, resulting in high operational costs. Consequently, their deployment is largely limited to professional leagues and elite-level clubs. This creates a gap for smaller leagues, grassroots sports organizations, and researchers who require tracking tools but lack access to such costly resources.

This project investigates a low-cost, accessible solution for real-time ball tracking using pre-recorded video footage sourced from online platforms. By implementing and comparing multiple lightweight tracking algorithms—including classical computer vision techniques and pre-trained object detectors such as YOLO—the project aims to develop a robust pipeline capable of estimating the ball’s position. A key innovation in the proposed framework is the fusion of uncertain outputs from multiple Image Processing Algorithms (IPA) using a Probabilistic Graphical Model (PGM), which enables robust inference even in the presence of noise and partial information (Koller and Friedman 2009). This probabilistic fusion strategy helps mitigate the limitations of individual trackers and reduces overall tracking errors, particularly in challenging conditions such as occlusion and motion blur.

The objective is to develop a cost-effective and computationally efficient tracking system that can run on consumer-grade hardware without compromising significantly on accuracy. This contributes to the democratization of sports analytics, opening the door to broader adoption in non-professional environments such as university clubs, school competitions, and community sports settings.

This research is conducted by Mr. A.H. Thorpe as part of the Mechatronic Project 478/488 under the supervision of Dr. Corne Van Daalen. It aligns with the graduate attributes prescribed by the Engineering Council of South Africa (ECSA).

1.1 Motivation

In many sports, decisions made by officials are often prone to error because they rely heavily on human judgment. These errors can accumulate, with some leagues averaging up to five officiating mistakes per match (Daily News 2023). For example, in

the Premier League, referees still rely on manually drawing lines on video footage to determine ball position for offside or goal-line decisions (Premier League 2020). While this method can be effective, it remains susceptible to inaccuracies caused by varying camera angles and inherent human limitations. An accurate, real-time ball tracking system could significantly improve decision-making by providing objective and precise positional data, reducing inconsistencies during match officiating (The Guardian 2025).

Beyond officiating, precise ball tracking also offers substantial benefits in player training and performance analysis. By capturing various aspects of both player movement and ball dynamics, coaches and athletes can gain deeper insights into performance metrics, enabling targeted improvements in technique, decision-making, and strategy. Such data-driven feedback has been shown to enhance skill development and game understanding across multiple sports disciplines (Oliva-Lozano et al. 2022).

In addition to these high-level benefits, many smaller clubs, schools, and amateur leagues lack the resources to implement advanced commercial tracking systems such as Hawk-Eye or TRACAB. These systems require high-performance cameras and computing infrastructure, making them inaccessible to most grassroots or academic environments. There is a clear need for a low-cost, computationally efficient alternative that can run on widely available consumer-grade hardware.

To address this gap, the project proposes a lightweight video-based tracking framework capable of estimating the ball's position in three-dimensional space using multiple complementary detection techniques. By combining the strengths of different methods, the system aims to deliver reliable tracking performance even under challenging conditions such as occlusion or motion blur.

1.2 Objectives

The objective of this project is to develop a model that can accurately and reliably estimate the position of a ball in real time using pre-recorded video footage. This will be achieved through the fusion of multiple IPA, each capable of independently detecting the ball's position. The outputs of these algorithms will be combined using a PGM to improve robustness and reduce the estimation error in position. The following objectives define the scope of this project:

1. Develop and implement multiple IPA to estimate the ball's position from video footage.
2. Design a PGM-based fusion method to combine the outputs of individual algorithms, improving tracking accuracy and reducing errors from isolated failures.

3. Evaluate and optimize the system to use the minimal number of algorithms required while maintaining acceptable performance.
4. Compare the final model's performance against each individual algorithm in terms of accuracy, robustness, and computational efficiency.

1.3 Report Overview

This section outlines each section that will be discussed in this report. The idea gets formed and the objectives set in Chapter 1. Chapter 2 focuses on the background and literature of the project where core concepts get formed from existing work. From there the Model Design takes place in Chapter 3, this is where the requirements of the IPAs and PGMs are set. Chapter 4 will then go into the implementation of the IPAs and PGM and how the final Algorithms are chosen. Chapter 5 then goes into Experimentation and how the various parameters for the IPA and PGM were selected. Finally the last chapter will cover the conclusion of the report.

2 Background and Literature Review

Object tracking is a task that dates back to the earliest days of human civilization, from following animal tracks to observing celestial bodies for navigation. As technology has advanced, computers have become a key tool in tracking objects with high precision. One of the most common examples in modern life is the Global Positioning System (GPS), which allows users to determine their position anywhere on Earth (Challa 2011).

A major industry that relies heavily on object tracking is sports. Companies such as Hawk-Eye and TRACAB are global leaders in this domain, with Hawk-Eye specializing in officiating decisions and TRACAB focusing on performance analysis and training (Tracab 2024; Hawk-Eye Innovations 2024). Accurate tracking in sports benefits a wide range of stakeholders — from fans and broadcasters to coaches, players, and officials (Labayen et al. 2014). Given that sports are governed by clear rules and boundaries, it is natural that computer-aided systems are increasingly being used to enhance decision-making on and off the field.

Despite its advantages, ball tracking in sports presents numerous challenges. Balls tend to be small, fast-moving objects, often affected by motion blur and frequent occlusion in video footage. These issues can severely limit the performance of traditional image processing techniques. Furthermore, the nature of the tracking problem can vary significantly between sports. For instance, in table tennis, the ball exhibits highly nonlinear and non-periodic motion, making it particularly difficult to track reliably (Kamble, Keskar, and Bhurchandi 2019).

Because of these challenges, no single tracking algorithm can perform robustly in all scenarios. Each technique has its own weaknesses depending on the scene, motion, and visual obstructions. This motivates the use of Probabilistic Graphical Models, which provide a principled framework for combining uncertain outputs from multiple algorithms (Koller and Friedman 2009). PGMs allow relationships between multiple random variables to be explicitly modeled, making them particularly useful for data fusion and robust inference under uncertainty.

This section provides a review of the core technical concepts and prior work relevant to this project. It includes discussions on existing commercial tracking systems, Computer Vision techniques for tracking, IPA, PGMs and, identifies the current gaps in the literature that this project aims to address.

2.1 Existing Commercial Tracking systems

As mentioned in the introduction, two of the leading commercial solutions for ball tracking in sports are Hawk-Eye and TRACAB. Hawk-Eye operates using an array of high-speed cameras positioned around the stadium to capture the ball’s trajectory in real time with sub-millimetre accuracy (TennisNerd 2024). This system, while highly accurate, is prohibitively expensive for widespread use. Instal-

lation costs range between R1 million and R1.25 million per court (approximately \$55,000–\$70,000 USD), making it inaccessible to amateur clubs and individuals (Wong 2016). In addition to its cost, Hawk-Eye requires complex infrastructure, including camera placements two to three stories above the field, which adds to the installation difficulty and venue constraints.

TRACAB, another global leader, combines camera-based tracking with sensor-based solutions. The system includes multiple high-speed cameras—often sourced from Hawk-Eye—alongside a sensor chip embedded in the ball, developed by the German company Cairos Technology AG (World Intellectual Property Organization 2023). TRACAB is commonly used for both training applications and goal-line decision-making. However, its cost is even higher, with installation reported at around R5 million (approx. \$270,000 USD) and an additional R70,000 (approx. \$3,800 USD) per game to operate the system (Wiebe 2013).

While these high-end systems are effective, they are far beyond the reach of most local sports clubs, schools, and individual athletes. More affordable alternatives are emerging, such as XBotGo, which offers a ball-tracking system for a once-off cost of approximately R6,300 (about \$340 USD) (XbotGo 2024). Although this represents a more accessible option, it still requires the user to provide their own compatible camera equipment, which contributes to the total cost. Additionally, shipping limitations may restrict access to certain regions, particularly in developing countries.

2.2 Computer Vision techniques for tracking

Computer Vision is used when significant information is needed to be extracted from images (Stockman and Shapiro 2001). In terms of tracking objects this is vital, as this opens the door to be able to extract vital information of where the object is located. The foundation of object tracking is object detection, which is identifying an object of interest in an image. Wheres object tracking evolved into is being able to "see" the motion of the object as it moves through the frames of a video(Zhao et al. 2015). Object tracking doesn't come without its challenges with the most common being occlusion, illumination variation, and fast motion(Soleimanitaleb, Keyvanrad, and Jafari 2019).

2.2.1 Traditional Tracking and Detecting methods

There are many different methods that lead to how CV was used for tracking but most of them are based on foundational mathematics and statistics.

One such example uses recursive Bayesian logic formulated and derived from the Bayesian probabilistic framework. Examples of algorithms traditionally found are the Kalman filter, extended Kalman filter, unscented Kalman filter, point mass filter, and particle filter. These algorithms were developed to find a resolution to generic

estimation and filtering problems (Challa 2011).

An example based on using mathematical models for detection and tracking is from a study done in 2013 that proposed an algorithm based on integrating the mechanisms of the Human Visual System (HVS) where it detected and tracked infrared dim and a small target. The method is based on 4 steps, firstly a multi-scale Difference of Gaussians, which is a method of detecting features of different scales, which is then placed in feature maps at different scales, the next step was to add a Gaussian window to this map at a location near the object, the penultimate step was to normalize the image and decide on the location of the object in this frame and if this is the final step the algorithm stops. However if there is another frame it estimates the location on the next frame via a Proportional-Integral-Derivative algorithm and goes back to step 1(Mirzaei et al. 2023; Dong et al. 2014).

These both provide an idea of what is traditionally done in terms of tracking and detecting objects through CV.

2.2.2 Modern Deep Learning-Based Tracking

As technology in software and hardware advanced so did the methods in which tracking and detecting was done, Convolution Neural Network based on Deep Learning technology(Pal et al. 2021), which is a relatively recent breakthrough in CV and how tracking is done from video footage.

CNNs are used to identify features in which they are used to localize and classify the objects within the frames(Pal et al. 2021), these make it a very powerful tool for tracking and detecting objects. There are two methods, being discriminative and generative. The discriminative method focuses on binary classification, often called CNN-C, which distinguishes an object from its background. The Generative method, often called CNN-M focuses on learning a robust similarity function, which aims to find a match to the object template from a specified region(Li et al. 2018).

There are a few common modern approaches to object detection, one being the use of CNN as described in the previous paragraph, Transformer based approaches, Vision Language models and lastly Hybrid models such as RetinaMask and EfficientDet(Sapkota et al. 2025). A benefit of YOLO is that it has been trained on the COCO dataset to be able to detect sports balls as well as that YOLO has a small model size and fast calculation speed.

2.3 Image Processing Algorithms

This section discusses how IPA can be used to extract relevant raw data for the PGMs. Image Processing Algorithms (IPA) is a term that can be used in equivalence to CV as in many cases IPA simulates human vision (Pitas 2000).

IPA can be broken down into 3 distinct categories, Low-Level, Intermediate level, and High-level vision algorithms. The low-level vision focuses on the most basic operations applied to raw pixels, the goal is to improve the image quality or extract very simple features. The intermediate level vision takes this cleaned low-level vision image and organizes these pixels into meaningful structures, this helps bridge the gap between pixel-level data and understanding of the objects in the image. Finally the high-level vision, is where Machine Learning (ML) comes into play and allows for object recognition (Pitas 2000).

2.3.1 Low-Level Vision

As stated before, low-level vision refers to the basic pixel-level processing that prepares raw images for higher-level interpretations (Ji 2013). In order to accomplish this various techniques can be applied, these enhance the image without altering underlying information within the image. The techniques that can be used are Edge Detection, this is where the operation identifies points on the image where brightness changes sharply, Filtering, which enhances certain aspects of an image, and finally Thresholding, which converts grayscale images into binary images based on a threshold value (MetaEye 2025) by mapping intensity values above or below a threshold into distinct classes (such as the background and foreground). In practice this is commonly used in pre-processing video frames to enable more robust object detection and tracking.

2.3.2 Intermediate Level Vision

This level builds on the processing performed in low-level vision by assembling coherent objects and motion patterns. As discussed above this level bridges the gap between low-level and high-level visions (MetaEye 2025). Similar to Low-level vision, several techniques are commonly used at the intermediate level. These include segmentation, which divides the image into various segments or regions that share similar attributes (For example the watershed algorithms or k-means clustering), Object detection which identifies and localizes objects within each frame and finally Optical flow which estimates the motion between two consecutive frames (MetaEye 2025). Together all of these help to have a basic level of detection within the videos, such as tracking of a ball on the field. The final level being High-level vision is when classification of the objects come in, as this isn't in the scope of this project a further discussion is not needed.

2.3.3 OpenCV

OpenCV is an Open Source Computer Vision library for image and video analysis and was developed by Intel over two decades ago. Originally written in C++/C but now has wrappers for Python, Java or Matlab (Culjak et al. 2012). The library

possesses various techniques for Low-level vision, intermediate level vision and High-level vision (Mohamad et al. 2015). Key functioning included are Threshold, Filtering, Finding edges, segmentation, background subtraction, and various other functions that can be applied to various different IPA (Marengoni and Stringhini 2011).

2.3.4 OpenCV Functions for Low- and Intermediate-Level Vision

This subsection provides a detailed breakdown of the low-level and intermediate-level vision functions available in OpenCV, which are relevant for image processing algorithms used in ball tracking. These functions are implemented in the library and can be accessed via Python bindings, as utilized in this project.

Low-Level Vision

As discussed previously low-level vision focuses on basic pixel-level processing that prepares videos or images for higher-level interpretations. The relevant functions available in OpenCV are summarized in Table 1.

Table 1: Low-Level Vision Functions from OpenCV

Operation	Description
Thresholding	Converts grayscale images into binary images based on a threshold value, this is useful for separating the object from the background. Can use functions like <code>cv2.threshold()</code> (applies a fixed threshold to pixel intensities for binary segmentation) (OpenCV Team 2023j) and <code>cv2.inRange()</code> (checks if the pixel values fall within a specified ranges for color-based masking) (OpenCV Team 2023k).

Operation	Description
Filtering	Enhances certain aspects of an image, such as smoothing or sharpening, which in turn helps improve ball visibility. Can use functions like <code>cv2.medianBlur()</code> (replaces each pixel with the median intensity of neighboring pixels to reduce noise) (OpenCV Team 2023q) and <code>cv2.GaussianBlur()</code> (applies Gaussian smoothing, which blurs the image by averaging nearby pixels with weights that decrease with distance from the center, effectively reducing high-frequency noise while preserving edges better than uniform averaging) (OpenCV Team 2023r).
Frame differencing	Computes the absolute difference between consecutive frames to detect motion. Can use <code>cv2.absdiff()</code> (calculates pixel-wise absolute difference for motion detection) (Szeliski 2010).
Finding edges	Identifies points where brightness changes sharply, which can help outline the ball against the court or field. Can use functions like <code>cv2.Canny()</code> (multi-stage edge detection which uses gradients and hysteresis thresholding) (OpenCV Team 2023b) and <code>cv2.Sobel()</code> (computes image gradients for edge detection in horizontal or vertical directions) (OpenCV Team 2023c).
Color space conversion	Transforms images between different color spaces, which can enhance color-based segmentation of the ball. Can use <code>cv2.cvtColor()</code> (changes color representation, for example going from BGR to HSV for better color separation) (OpenCV Team 2023d).

Operation	Description
Morphological operations	Techniques like dilation and erosion that refine binary images by removing noise and closing gaps in detected objects. Can use functions like <code>cv2.dilate()</code> (expands bright regions in binary images using a structuring element) (OpenCV Team 2023l), <code>cv2.erode()</code> (shrinks bright regions to remove small noise) (OpenCV Team 2023m), and <code>cv2.morphologyEx()</code> (applies advanced operations like opening or closing for shape refinement) (OpenCV Team 2023n).

Intermediate-Level Vision

Building on low-level processing, intermediate-level vision assembles coherent objects and motion patterns. The relevant functions available in OpenCV are summarized in Table 2.

Table 2: Intermediate-Level Vision Functions from OpenCV

Operation	Description
Segmentation	Divides the image into segments or regions that share similar attributes, this helps isolate the ball. Can use functions like <code>cv2.findContours()</code> (detects contours in binary images for shape analysis) (OpenCV Team 2023h), <code>cv2.connectedComponents()</code> (labels connected regions in binary images for segmentation) (OpenCV Team 2023e), and <code>cv2.HoughCircles()</code> (detects circular shapes for ball-like object segmentation) (OpenCV Team 2023i).

Operation	Description
Object detection	Identifies and localizes objects within each frame, this is crucial for tracking the ball. Can use shape-based filtering with <code>cv2.findContours()</code> (detects contours for shape analysis) (OpenCV Team 2023h), <code>cv2.convexHull()</code> (computes convex hull for solidity) (OpenCV Team 2023f), and <code>cv2.arcLength()</code> (calculates perimeter for circularity) (OpenCV Team 2023g) to filter ball-like objects based on properties like aspect ratio, circularity, and solidity.
Background subtraction	Separates moving foreground objects from the static background, useful for isolating the ball in video frames. Can use <code>cv2.createBackgroundSubtractorMOG2()</code> (creates a MOG2 background subtractor for motion detection) (OpenCV Team 2023a).
Optical flow	Estimates motion between two consecutive frames, useful for predicting ball movement. Can use functions like <code>cv2.calcOpticalFlowFarneback()</code> (dense optical flow estimation using polynomial expansion) (OpenCV Team 2023o) and <code>cv2.calcOpticalFlowPyrLK()</code> (sparse optical flow using the Lucas-Kanade method) (OpenCV Team 2023p).

2.4 Probabilistic Graphical Models

Probabilistic Graphical Models (PGMs) offer a powerful framework for representing and reasoning about probabilistic relationships, allowing computers to answer complex questions under uncertainty. They are widely used across various applications, including speech recognition, image segmentation, disease outbreak modeling, and CV, which is the primary focus of this project (Koller and Friedman 2009). PGMs combine probability theory and graph theory to create structured and interpretable models of complex distributions, and they scale effectively to high-dimensional problems. The structure and concepts throughout these following sub sections are based primarily on the foundational work presented by Koller and

Friedman(Koller and Friedman 2009) as well as the courses presented by Daphne Koller on Coursera (Koller and Stanford 2025b; Koller and Stanford 2025a).

This section explores the key concepts of PGMs that are directly relevant to the object tracking approach developed in this project. It begins with a discussion on modeling, introducing the way graphical models are constructed and how probabilistic dependencies are encoded within them. Following that, the section discusses inference, describing how the various nodes in the model interact and how information is propagated throughout the graph to produce a coherent estimate. It then examines Gaussian factors, explaining how Gaussian distributions are used to represent uncertainty and how these distributions are applied to the outputs of image processing algorithms. Finally, it concludes by demonstrating how this probabilistic output is used for object tracking, and how it contributes to a more reliable and interpretable tracking system.

2.4.1 Modeling

Models are declarative representations of our understanding of the world. They take in input data, apply structured reasoning, often through algorithms, and produce output that is relevant to the user’s task. In the context of PGMs, the modeling is done through probabilistic reasoning, which is essential for managing uncertainty.

When creating a model in the context of PGMs, three main steps are typically followed. First, the Random Variables (RV) involved in the process are identified.

Next, a graphical model is constructed by selecting a representation suited to the problem. There are two main types of graphical models used in PGMs: Bayesian Networks (BNs), which are directed, and Markov Networks (MNs), which are undirected. Examples of both are shown in Figure 1.

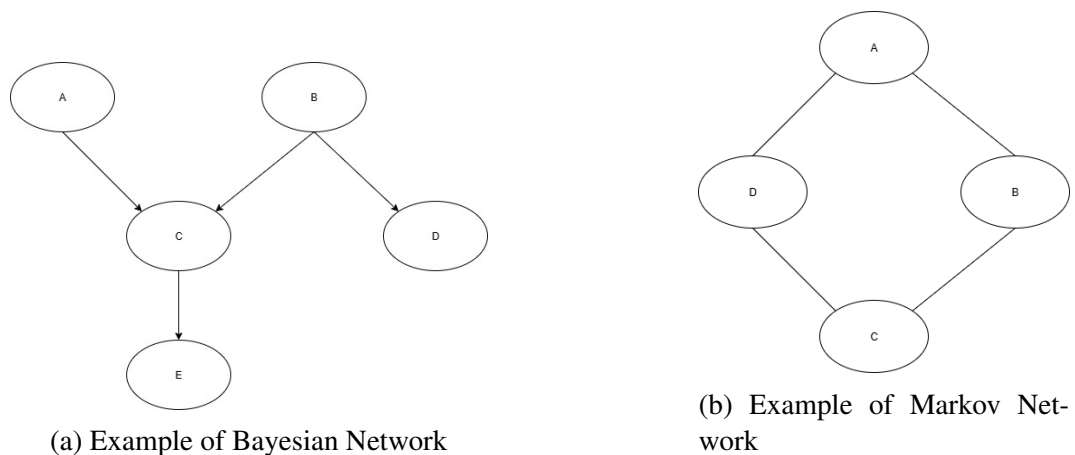


Figure 1: Examples of graphical models

In Figure 1a, the BN is represented as a directed graph, where edges have arrows indicating conditional dependencies between random variables. A node with out-

going arrows is referred to as a parent, while nodes receiving those arrows are its children. For example, the arrow from node A to node C signifies that “C depends directly on A,” or equivalently, “A influences C.”

By contrast, the MN in Figure 1b is undirected, with edges lacking any direction. This reflects symmetric dependencies: neither node is assumed to influence the other in a causal sense. In this case, the undirected edge between A and B indicates that “A and B are directly dependent,” without implying a specific direction of influence.

Finally, the last step is to define the factors required by the graph. For a BN, each factor corresponds to a Conditional Probability Distribution (CPD), a conditional probability ($P(A | B)$) distribution for a variable given its parents. If a node has no parents, its factor is simply a marginal distribution $P(A)$. For example, in the BN from Figure 1a, the associated distributions are:

$$P(A), P(B), P(C|A, B), P(D|B), P(E|C) \quad (1)$$

These are combined into one joint distribution ($P(X_1, \dots, X_n)$) using the chain rule:

$$P(A, B, C, D, E) = P(A).P(B).P(C|A, B).P(D|B).P(E|C) \quad (2)$$

This joint distribution can then be used like any other multivariate distribution.

While this approach is manageable for small networks, it becomes computationally expensive for larger ones. This is where inference comes in. The next section discusses how inference methods, particularly belief propagation, enable efficient probabilistic queries without explicitly computing full joint distributions.

2.4.2 Inference

Inference is the process of calculating probabilities of RVs of interest given partial information. In practice, this means calculating the probability distribution of one RV conditioned on observed evidence from other RVs. This is very useful as in the real world not every RV is observed, therefore the use of inference provides reason for unobserved RVs.

All further explanations will be explained for BN because they can be encoded into a MN. This conversion from BN to MN involves removing the directional edges as well as making the inference more efficient. The conversion is achieved through moralization which is done in two steps:

- 1) Connect all the parents of a common child - This means if two nodes have the same child an edge gets placed between them, also known as marrying the parents.

- 2) All directional edges are dropped and replaced with undirected edges.

This method is shown as an example in Figure 2, where it can be seen that the directed edges are converted to undirected edges and all parents who share children are joined.

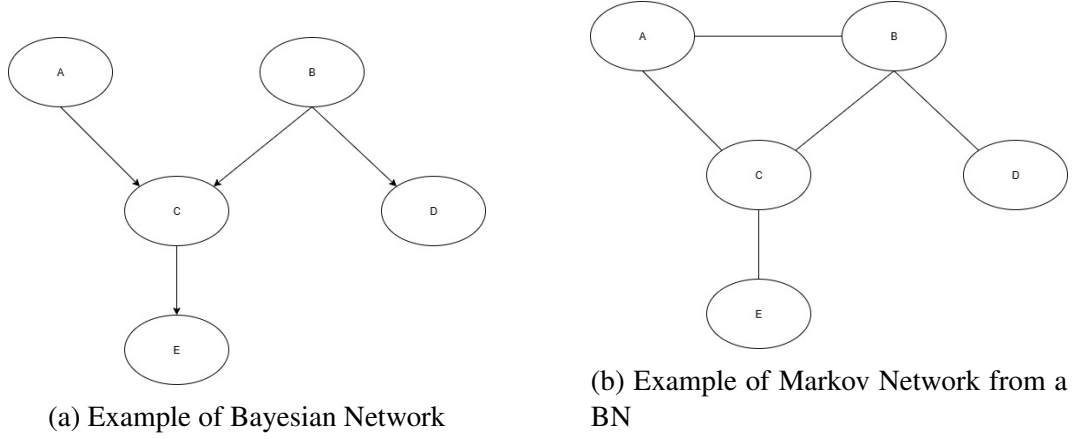


Figure 2: Conversion of Bayesian Network to Markov Network

Now that all networks are MN, the focus shifts to the two main methods of applying inference, Belief Propagation (BP) and Belief Update (BU). These two methods are very similar, with a few differences.

BU is typically applied to a single variable or a small set of variables, using Bayes' Theorem directly:

$$\text{Posterior belief} = \frac{\text{Likelihood} \times \text{Prior belief}}{\text{Normalization factor}} \quad (3)$$

The purpose of BU is to refine the belief for a specific variable when new evidence arrives. This update is applied once per new observation, sequentially.

An example for this is if we look at Figure 2b. If evidence at D is observed, Bayesian Updating can be applied to compute the belief for B first, as B is directly connected to D, then the new belief of B can be used to update the beliefs of A and C, which can be extended as far as needed.

In terms of mathematically explaining this, let evidence be observed at node D , denoted $D = d$. Using Bayesian Updating, we first update the belief for the neighboring node B :

$$P(B \mid D = d) = \frac{P(D = d \mid B) P(B)}{P(D = d)} \quad (4)$$

where the normalization factor is

$$P(D = d) = \sum_B P(D = d \mid B) P(B). \quad (5)$$

Next, we propagate the updated belief to the neighbors of B , namely A and C :

$$P(A \mid D = d) = \sum_B P(A \mid B) P(B \mid D = d), \quad (6)$$

$$P(C \mid D = d) = \sum_B P(C \mid B) P(B \mid D = d). \quad (7)$$

Finally, the belief at E can be updated via Σ over C :

$$P(E \mid D = d) = \sum_C P(E \mid C) P(C \mid D = d). \quad (8)$$

This demonstrates sequential Bayesian Updating: starting from the observed evidence and propagating through the network to refine the beliefs of all connected nodes.

The other method is BP, which is a message passing algorithm for inference in graphical models. This means that each node sends "messages" between each other about variables they have in common. The nodes then combine these messages into beliefs.

2.4.3 Gaussian Factors

For object tracking tasks such as in this project, Gaussian factors are especially useful as they can naturally represent noisy measurements of continuous variables, such as the ball's position across the video frames. Gaussian factors often refer to factors or potential functions defined by a multivariate Gaussian distribution over a set of RV. There are two main ways of describing Gaussian (Normal) distributions

1. Mean-Covariance form:

$$\mathcal{N}(x; \mu, \Sigma) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left\{ -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right\} \quad (9)$$

where:

- x : Vector of RVs
- n : Length of x
- Mean (μ): This is the "center" or average of the data.

- Covariance (Σ): This represents how much the values vary and how the different dimensions move together. A higher covariance on the main diagonal represents a lower confidence and large variance from the mean.

2. Canonical (Information) form:

$$C(x; K, h, g) = \exp \left\{ -\frac{1}{2} x^T K x + h^T x + g \right\} \quad (10)$$

where:

- Information (Precision) Matrix K : Inverse of covariance, therefore a higher precision means lower uncertainty ($K = \Sigma^{-1}$)
- Information Vector h : Related to the mean ($h = K\mu$)
- Constant g : Adjusts the overall height but isn't always required.

The reason the canonical form is important for using Gaussian factors in a PGM is that it allows for more efficient inference. Inference for Gaussian factors in PGMs is performed via message passing on a cluster graph, following these steps:

1. Create Cluster graph: Group all the factors together into "Clusters" connected by shared variables.
2. Assign Factors to Clusters: Each cluster's "Potential" is a product of all the Factors assigned to it.
3. Send messages between Clusters: Each message marginalizes out the cluster's private variables, passing only the shared variables into its neighbor.
4. Compute Beliefs: Multiply incoming messages at each Cluster to get the final belief over its variables.

2.4.4 Hybrid Factors

Hybrid networks are PGMs that include both discrete and continuous RVs within the same modeling framework. This needs to be discussed because pure discrete networks aren't able to handle continuous measurements, and pure continuous networks can't represent discrete measurements. This section discusses what happens when a network or model has both discrete and continuous RVs.

Hybrid factors usually depicted mathematically as:

$$\phi(x, y) \quad (11)$$

where:

- x : Discrete RVs
- y : Continuous RVs

What this does is it handles the interaction between categorical decisions and real-valued outcomes. The joint distribution then factorizes as:

$$p(x, y) = \prod_{i=1}^n \phi_i(X_i, Y_i) \quad (12)$$

where each factor ϕ_i can be either purely discrete, purely continuous or truly hybrid. This factorization is important as it enables inference and represents dependencies.

Conditional Linear Gaussian (CLG) Networks

The most commonly used class of Hybrid Networks is a CLG network. A CLG network provides the structure to handle mixed discrete-continuous variables while keeping computational feasibility. A CLG network is generally preferred over general hybrid models as it is more tractable for inference, allowing for exact probabilistic computation rather than requiring approximation methods.

The CLG network is a hybrid network where:

- Discrete variables can have any parents. (Discrete or Continuous)
- Continuous Variables with discrete parents follow a Gaussian distribution whose parameters depend on the parents.
- Continuous Variables with continuous parents must follow a linear Gaussian model.

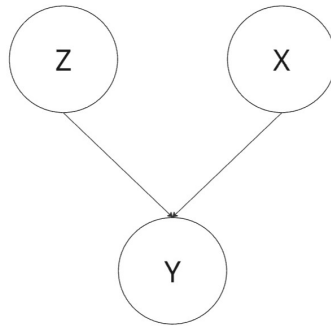


Figure 3: Example of a Conditional Linear Gaussian network showing discrete RV X and continuous RVs Y and Z

A general setup of a CLG network is shown in Figure 3, where Y is a continuous RV with discrete parent X and continuous parent Z , the conditional distribution is given by:

$$P(Y|Z, X = x) = N(Y; \mu_x(Z), \Sigma_x) \quad (13)$$

where:

- $\mu_x(Z) = a_x + b_x Z$ is a linear function of Z with parameters depending on the discrete state x .
- Σ_x is the covariance matrix also depending on x .

With all of this in mind, inference in CLG networks becomes more complex due to the mixture of discrete and continuous RVs. This is where Expectation Propagation (EP) allows for easier inference.

2.4.5 Expectation Propagation (EP)

EP is a deterministic approximation technique for BNs (T. P. Minka 2013). EP unifies two common approaches to inference techniques, the Kalman filter and Loopy BP, which is an extension of previously mentioned BP (T. P. Minka 2013). The method of EP is useful for Hybrid networks, as it approximates the belief states by retaining expectations such as means and variances (T. P. Minka 2013).

EP operates by approximating the posterior distribution through iterative moment matching between 'cavity' distributions (approximations excluding one factor's influence, denoted $q_{-i}(x)$) and 'tilted' distributions (cavity multiplied by the excluded factor, $q_{-i}(x) \cdot f_i(x)$). This process minimizes the Kullback-Leibler divergence (Thomas P Minka 2001; Heskes 2006).

EP uses an iterative approach which in many cases is computationally infeasible. A special case of EP is Assumed Density Filtering (ADF) or commonly called Moment Matching. This method is done by initializing all the approximating factors except the first one to be uniform and then only allowing a single pass (Bishop 2006, p. 510). This is necessary in the cases when data points arrive sequentially and the model must learn from each point and then discard the data before moving on to the next point (Bishop 2006, p. 510).

Key steps for the ADF algorithm are as follows:

1. **Initialization:** Start with simple approximations for all factors, assuming minimal prior knowledge.
2. **Sequential processing:** Calculate the 'cavity' and 'tilted' distributions and then match moments of the 'tilted' distribution to the Gaussian distribution, updating the approximation immediately.

3. **Single Pass:** Unlike the full EP's iterations, the ADF processes each data point once, discards it and then moves on to the next data point.

2.5 Gaps

Despite advancements in commercial ball tracking systems and computer vision techniques, significant gaps remain in accessible, real-time solutions for sports analytics. High costs and infrastructure requirements of systems like Hawk-Eye limit their use beyond professional leagues, while open-source alternatives often lack the precision needed for reliable ball trajectory estimation. Additionally, integrating computer vision outputs with probabilistic graphical models for inference in hybrid networks presents challenges, particularly in handling occlusions, fast motion, and sequential data processing. Approximation methods like Expectation Propagation offer potential, but their computational demands and accuracy trade-offs in dynamic environments require further optimization for low-cost, real-time applications.

3 Model Design

This section highlights the modular architecture, video footage, the role of PGMs, and the use of image processing algorithms. Figure 4 illustrates the overall pipeline of the system. The process begins with a sports video containing a ball. This video is passed through multiple image processing algorithms, each producing an independent estimate of the ball's position. These outputs are then fused using a PGM, which produces a final, unified estimate of the ball's position.

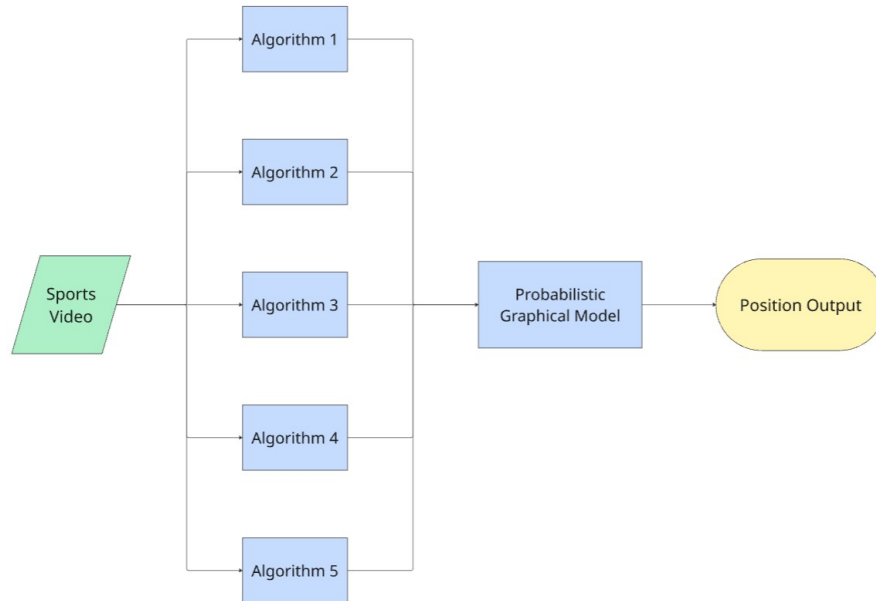


Figure 4: Overview of the model design

3.1 Video Footage Description

The sports video footage used in this project is sourced from a youtube video that displays a Court Level Tennis practice between Zverev and Djokovic (Apilado 2024). The video is 43 minutes long and downloaded in MP4 at 1440p60 quality. The footage is captured at court level, with the camera positioned behind one of the players, where the ball is mostly visible and clearly identifiable. The video will be cut down into shorter video clips where the ball is in play, this is done to ensure manageable processing time, and will be using 30 Frames Per Second (FPS). The video will be pre-processed to ensure consistent lighting and color balance, which helps improve the performance of the image processing algorithms.

3.2 Image Processing Algorithms

This subsection builds on the discussion of IPA in the literature review (Section 2.3), where IPAs are categorized into various levels. The purpose of IPA in this project is to extract the raw data of the ball's position in each frame of the video and the reason for having multiple IPA is to ensure that if one algorithm fails to detect the ball in a frame, the other algorithms and the PGM can account for this failure and still provide reliable tracking of the ball. This subsection outlines the requirements needed of the IPA, a discussion and design of the various candidate algorithms, and the differences in the algorithms.

3.2.1 Requirements

To ensure that the IPAs effectively support ball tracking, they must meet specific requirements. These requirements are derived from the project's needs for real-time performance and robustness to dynamic sports environments, building on the challenges discussed in the literature review. The key requirements for the IPAs are outlined in Table 3.

Table 3: Requirements for Image Processing Algorithms

Requirement	Description
Robustness to noise and distractions	The algorithm must reliably track the ball without getting "distracted" by background noise, such as players, lines on the fields or courts, or spectators. This ensures consistent detection in cluttered sports footage.
Accuracy in ball center localization	The algorithm should precisely estimate the ball's center position, as well as minimizing errors that could affect trajectory prediction.
Process video footage at 30 FPS	The algorithm must process frames at 30 FPS to support close to real-time tracking, maintaining efficiency for video analysis without significant time delays.

Requirement	Description
Expand Scalability	The algorithm should handle varying video lengths, resolutions, and input formats with minimal performance degradation, allowing easy adaptation to diverse footage.

3.2.2 Algorithm Design

This subsection discusses the general design of the IPAs, to ensure that the requirements in Section 3.2.1 are met. The design is presented as a Flow Diagram as per Figure 5, where the IPA needs to be able to take in sports video footage, process each frame, and output an estimated position. The per-frame pixel coordinate (x, y) is normalized by the frame width W and height H as follows:

$$x_{\text{norm}} = \frac{x}{W}, \quad y_{\text{norm}} = \frac{y}{H} \quad (14)$$

which maps positions into the unit square $[0, 1] \times [0, 1]$. The idea behind using $[-1, -1]$ is that the later PGM will be able to identify easily that this input is not ideal and will ignore it since that specific algorithm didn't detect anything in that frame. These outputs then need to be outputted into a text file that can be used in the PGM, where the naming of the text file will be `alg_n.txt`, where $n \in \{1, \dots, N\}$ and N is the number of algorithms used. Designing the general IPA to work in this way satisfies several of the requirements, being that it can process each frame at 30 FPS, it can easily be scaled to various video lengths, and it can be adapted to different video formats. The rest of the requirements will be met according to the specific Low-Level and Intermediate-Level operations used in each algorithm, the position of which can be seen in Figure 5.

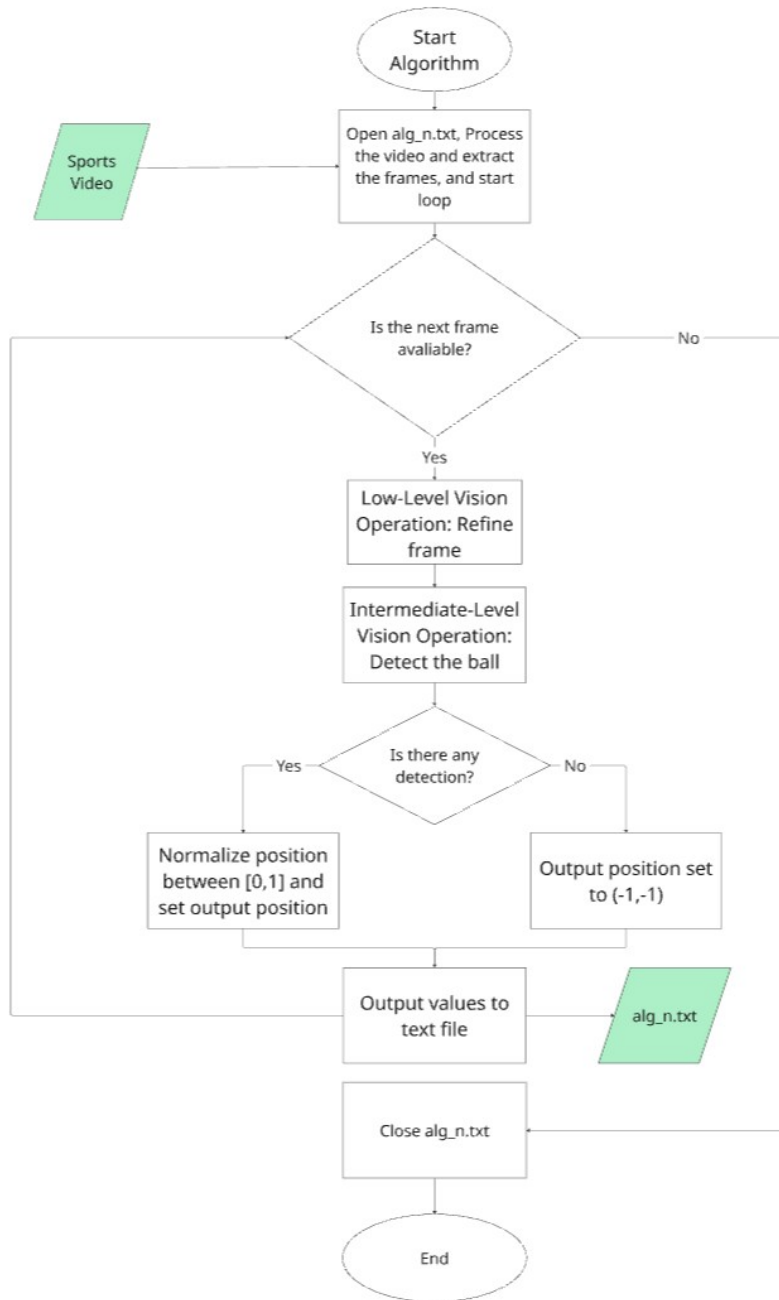


Figure 5: Flow diagram showing the flow of logic each of the IPA will follow

3.2.3 Candidate Algorithms

This subsection discusses and guides the design of the various candidate algorithms that can be considered for this project. The functions and operations of these algorithms are based on the OpenCV library, as discussed in Section 2.3.4. The aim of this subsection is to provide various combinations of operations that can be used to

create different algorithms, which will then be tested and evaluated to determine the best performing algorithms for this project. The candidate algorithms are summarized in Table 4 and described further below the table.

Table 4: Candidate Image Processing Algorithms

Algorithm Name	Low-Level Vision Operation Used	Intermediate-Level Vision Operation Used
HSV Color Thresholding	Median blur and morphological operations	Contour detection and shape analysis
Shape-Aware Motion Tracker	Frame differencing, thresholding, and morphological operations	Contour detection, ellipse fitting, and shape analysis
Brightness Streak Detector	Frame differencing, thresholding, and dilation	Contour detection and shape analysis
Motion Blur Hybrid	Frame differencing, thresholding, and morphological operations	Contour detection and shape analysis
Background Subtraction Tracker	Morphological operations	Background subtraction (MOG2), contour detection and shape analysis
Canny Edge and Hough Circles	Canny edge detection	Hough circle transform

Algorithm Name	Low-Level Vision Operation Used	Intermediate-Level Vision Operation Used
Lucas-Kanade Optical Flow	N/A	Optical flow computation and feature tracking
Template Matching	Template correlation	Localization

The descriptions for each algorithm are as follows:

- **HSV Color Thresholding:** This algorithm uses color-based segmentation by converting the frame to a HSV color space. It applies median blur to reduce noise while preserving edges, followed by morphological operations like erosion and dilation to refine the binary mask. Contours are then detected, and shape analysis filters for circular shapes based on metrics like circularity and aspect ratio, making it effective for detecting the ball against varying backgrounds, however it may potentially be sensitive to lighting changes affecting color perception.
- **Shape-Aware Motion Tracker:** This algorithm focuses on motion detection. It computes frame differencing to highlight moving objects, applies thresholding to create a binary image, and uses morphological operations to clean up noise. It detects contours and fits ellipses to candidate shapes, then applies shape analysis to filter based on elongation and size, which is particularly useful for tracking fast-moving balls but may struggle with slow or stationary objects.
- **Brightness Streak Detector:** This algorithm is designed to detect bright streaks caused by ball motion blur. It performs frame differencing to capture motion, thresholds the result, and applies dilation to connect fragmented streaks. Contour detection identifies potential ball paths, with shape analysis filtering for elongated shapes typical of motion-blurred balls, offering robustness to high-speed motion but potentially missing slow-moving or non-blurred balls.
- **Motion Blur Hybrid:** This algorithm combines motion detection with shape analysis for blurred trajectories. It uses frame differencing and thresholding

to isolate moving regions, applies morphological operations for refinement, detects contours, and analyzes shapes for elongated forms regardless of motion blur. This approach should excel when there is significant motion blur but may require tuning for different blur intensities.

- **Background Subtraction Tracker:** This algorithm utilizes background modeling to separate foreground from background. It applies morphological operations to preprocess frames, uses MOG2 background subtraction to generate foreground masks, detects contours in the mask, and applies shape analysis to identify ball-like objects. It should be effective for static camera scenarios but may be affected by changing backgrounds or lighting variations.
- **Canny Edge + Hough Circles:** This algorithm detects edges using the Canny edge detector to identify object boundaries. It then applies the Hough circle transform to find circular shapes. It should be particularly suited for detecting the ball when it's well-defined and circular, offering geometric precision but potentially struggling with deformed or partially occluded balls.
- **Lucas-Kanade Optical Flow:** This algorithm implements sparse optical flow using the Lucas-Kanade method. It tracks feature points across frames to estimate motion trajectories. It should be ideal for following the ball's path over time, providing temporal continuity, but requires good feature initialization and may drift over long sequences.
- **Template Matching:** This algorithm uses correlation-based template matching to locate the ball by comparing a reference template against the frame. It should be straightforward and effective when the ball appearance is consistent, but can be computationally intensive and sensitive to scale, rotation, or appearance changes.

These algorithms all have been designed and tested to ensure the algorithms meet the requirements outlined in Section 3.2.1. All of these algorithms use low-level vision to remove noise and make the algorithms more robust and free from distractions. The intermediate-level vision operations are utilized to accurately determine the center of the ball's position. Therefore based on Figure 5 and the descriptions above, these algorithms meet all of the requirements set.

3.3 Probabilistic Graphical Model

The purpose of incorporating PGMs in this project is to combine the outputs of multiple image processing algorithms in a principled and probabilistic manner. These algorithms, while effective individually, are not always reliable, they may occasionally fail to detect the ball or produce inaccurate estimations due to occlusion, motion blur, illumination variation or other visual disturbances.

A PGM allows for a robust fusion of these imperfect outputs by modeling both observed and latent variables, capturing the uncertainty inherent in each tracker’s prediction. This subsection presents the design and construction of the PGM model used in the project, including its design requirements, structure, variable definitions, and how inference will be performed to estimate the ball’s true position.

3.3.1 Design Requirements

In order to effectively design and implement the PGM for this project, a clear set of requirements must be defined. These requirements provide a foundation for understanding how the model should be structured, what functionality it must support, and how the final output should behave. In particular, the requirements address aspects such as uncertainty handling, output quality, and model scalability. Table 5 outlines the key requirements that the proposed PGM must satisfy.

Table 5: Requirements for the Probabilistic Graphical Model

Requirement	Description
Handle uncertainty	The PGM must account for noise or missing data in algorithm outputs by modeling them probabilistically.
Fuse multiple IPAs	Must combine several algorithm outputs at each frame into a multivariate normal distribution.
Temporal consistency	The model should account for the smooth, continuous nature of the ball’s motion by modeling how its position evolves over time, ensuring that current estimates depend on previous positions.
Scalable inference	The model must support efficient inference algorithms that remain computationally feasible as the video length increases or as additional tracking algorithms are integrated.

Requirement	Description
Modular design	The structure should allow easy integration or removal of trackers without changing the core model.
Factor structure	The relationships between variables must be clearly defined using Gaussian Distributions.

3.3.2 Model Construction

A BN was selected for this project as its strengths lie in its ability to explicitly model the dependencies between variables through the use of directed edges. This is BNs strength when compared with MN for this project, while both can represent complex dependencies and perform inference under uncertainty (Koller and Friedman 2009), the causal reasoning of the BN works well for this project as the image processing algorithms depend on the actual position of the ball.

Random Variables

There are 3 RVs per time step, these are presented in Table 6.

Table 6: Random Variables for the Probabilistic Graphical Model

RV	Symbol	Description	Domain
Ball position	$Pos_{x,y}$	Actual position of the ball on the screen in normalized coordinates (x and y values)	$[0, 1] \times [0, 1]$
Algorithm outputs	Alg_n (for $n = 1, \dots, N$)	Output from the n -th image processing algorithm	$[0, 1] \times [0, 1]$
Outlier Indicator	a_n (for $n = 1, \dots, N$)	Discrete RV indicating if Alg_n is an outlier	$\{0 = \text{outlier}; 1 = \text{inlier}\}$

Since each image processing algorithm typically outputs a single (x,y) position estimate, this alone does not provide a continuous distribution over all possible ball locations. To meet the requirement of a multivariate normal distribution in the model, a Gaussian factor is applied to each algorithm's output. This is done by taking the algorithm's output as the mean μ_n of a multivariate Gaussian distribution and associating it with a covariance matrix Σ_n that captures the algorithm's expected positional uncertainty. The specific values for these covariance matrices, along with temporal motion parameters and outlier probability priors, will be determined through systematic empirical evaluation in Chapter 4.

This transforms the single point estimate into a probabilistic relationship, centered around the algorithm's prediction with an associated covariance that represents its uncertainty. The resulting domain for each algorithm's output becomes $[0, 1] \times [0, 1]$, with a continuous distribution that reflects the inherent error or noise in the algorithm's estimate.

Sometimes the algorithms may produce irregular or wrong outputs, for example if the ball is occluded or there are distractions within the frame then one of the IPA may output a position that is far from the actual ball position. To account for this the PGM includes a discrete RV called a_n for each individual IPA. This RV indicates if the output is an outlier or inlier, this therefore produces a Hybrid network as there are now both discrete and continuous RVs.

Relationships

Now that the RVs have been defined, the next step is to define the relationships between them. It is evident that the output of each algorithm directly depends on the actual position of the ball as well as whether the output is an outlier. Therefore, there is a causal relationship between $Pos_{x,y}$ and Alg_n and between a_n and Alg_n , as illustrated in Figure 6. This implies that the algorithm outputs are conditionally dependent on both the true ball position and the Outlier Indicator.

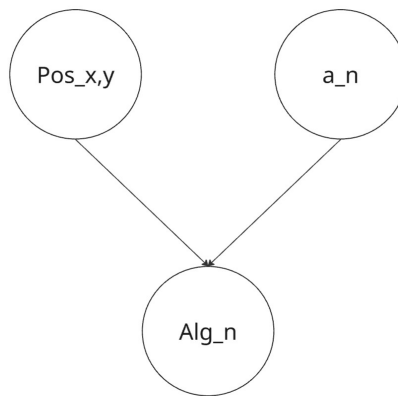


Figure 6: Relationship between $Pos_{x,y}$, Alg_n and a_n for single algorithm and time step

With this in mind, the full model can be defined as shown in Figure 7. This represents the final BN design used in this project for a single time step. (While the example illustrates five algorithms, the final number is still to be confirmed.)

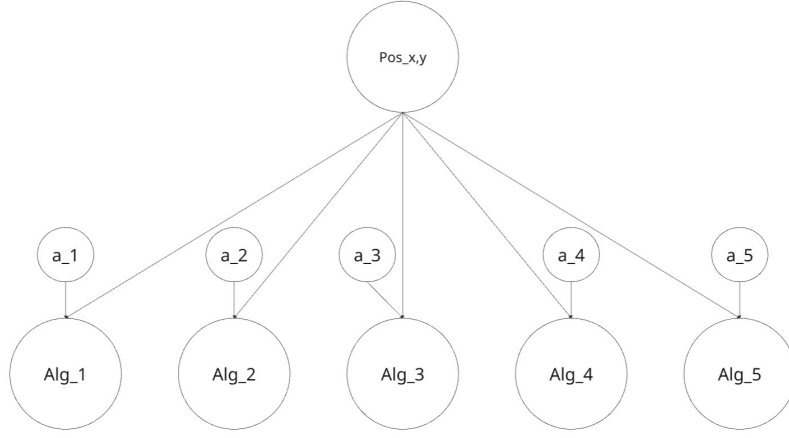


Figure 7: Bayesian Network showing the relationships between $Pos_{x,y}$ and the various algorithms used

Time-step connections

At this point in order for the ball to be tracked smoothly and efficiently the other frames need to be taken into account. This will be done with $Pos_{x,y}$ having a direct influence on the same RV in the next time step. From here the use of $Pos_{x,y}1, Pos_{x,y}2, \dots, Pos_{x,y}n$ will be used per time step. Figure 8 shows the relationships between each time step. As you can see, $Pos_{x,y}1$ has direct influence on $Pos_{x,y}2$, this means that the position of the ball in the next frame is dependent on the position of the ball in the current frame. This is important as it ensures that the ball's motion is modeled smoothly over time, reflecting its continuous trajectory across frames.

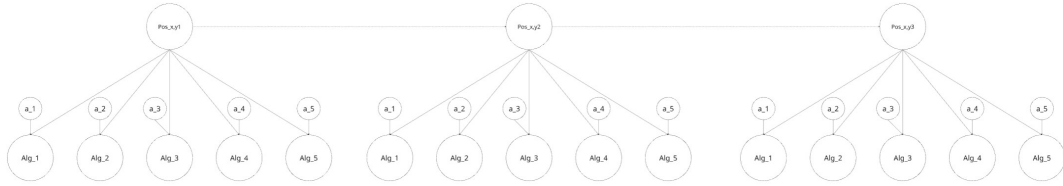


Figure 8: Bayesian Network showing the relationships between $Pos_{x,y}$, the various IPA used and the discrete Outlier Indicators at the different time steps

Factor Definitions

The probabilistic relationships in the BN are formalized through the following factor definitions:

- Temporal dynamics factor modeling the ball's motion between consecutive

frames

$$P(Pos_{x,y}^t | Pos_{x,y}^{t-1}) \quad (15)$$

- Observation model factor relating algorithm outputs to true position and outlier status

$$P(Alg_n | Pos_{x,y}, a_n) \quad (16)$$

- Outlier prior factor representing the probability that algorithm n produces an outlier

$$P(a_n) \quad (17)$$

These factors define the conditional dependencies within the hybrid network.

Mathematical Specification

These factor definitions now need to be formalized as Gaussian distributions. The mathematical specifications for the factors are as follows:

- Temporal dynamics factor:

$$P(Pos_{x,y}^t | Pos_{x,y}^{t-1}) = \mathcal{N}(Pos_{x,y}^t; Pos_{x,y}^{t-1}, \Sigma_{motion}) \quad (18)$$

- Observation model:

$$P(Alg_n | Pos_{x,y}, a_n) = \begin{cases} \mathcal{N}(Alg_n; Pos_{x,y}, \Sigma_n) & \text{if } a_n = 1 \text{ [inlier]} \\ \mathcal{N}(Alg_n; c, \Sigma_o) & \text{if } a_n = 0 \text{ [outlier]} \end{cases} \quad (19)$$

where:

- c : A constant mean.
- Σ_o : A large covariance matrix representing high uncertainty for outliers.

3.3.3 PGM Design

This subsection discusses the design flow of the PGM, as shown in Figure 9. The model should begin by reading the IPA outputs from their text files. For each time step (Frame), the model needs to set all of the initial means and covariance matrices. From there each the model should loop through each algorithm and perform EP to update the belief about the ball's position based on each algorithm's output and its outlier status. EP needs to be done here as this model is a Hybrid network and regular BP will not work for this model. The beliefs or messages are then fused from the last algorithm via Moment Matching as the inference method used is EP therefore there will be multiple difference guassians fusing messages together. Finally the outputs need to be converted to Mean-Covariance form and outputted to a text file that can be used for evaluation and to be overlayed with the original video footage to show the tracking.

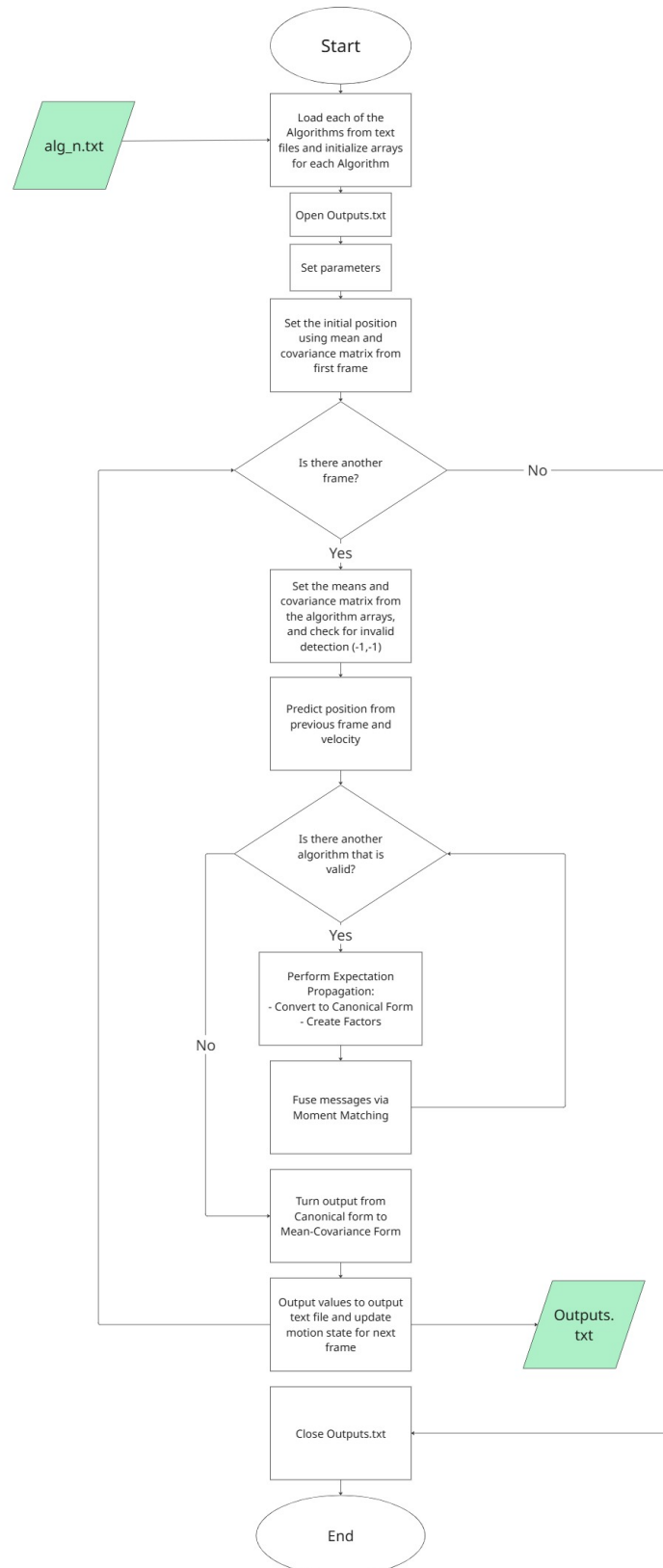


Figure 9: Flow diagram showing the flow of logic the PGM will follow

4 Implementation

This chapter explains how the model will be implemented, detailing both its conceptual workings and the practical coding approach. It will first present the underlying mathematics from first principles, followed by a description of the implementation using the EMDW library developed at Stellenbosch University. (this still needs to be completed, will emphasize more on what was said in the opening paragraph).

5 Experimentation

This chapter outlines the experimental setup for evaluating the performance of the ball tracking system. It describes the datasets used, the evaluation metrics, and the experimental procedures. The goal is to systematically assess how well the integrated image processing algorithms and probabilistic graphical model work together to accurately track the ball's position in various sports video scenarios. (this still needs to be completed, will emphasize more on what was said in the opening paragraph).

6 Conclusion

This chapter summarizes the key findings from the project, reflecting on the effectiveness of the proposed ball tracking system. It discusses the strengths and limitations of the approach, as well as potential areas for future work and improvements. The conclusion aims to encapsulate the overall contributions of the project to the field of sports analytics and computer vision. (this still needs to be completed, will emphasize more on what was said in the opening paragraph).

References

- Apilado, COURT LEVEL TENNIS - Liam (2024). *Djokovic and Zverev Full Court Level Practice | Points, Split-Screen, Groundstrokes etc. 2024 4K 60FPS*. Accessed: 2025-08-02. URL: <https://www.youtube.com/watch?v=n4TTjRjKB8I>.
- Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning*. Accessed: 2025-09-29. Springer. ISBN: 978-0387310732.
- Challa, Subhash (2011). *Fundamentals of Object Tracking*. Cambridge: Cambridge University Press. ISBN: 9780521192475.
- Culjak, Ivana et al. (May 2012). “A brief introduction to OpenCV”. In: *2012 Proceedings of the 35th International Convention MIPRO*. IEEE, pp. 1725–1730.
- Daily News (2023). *Referee accuracy: VAR still debatable*. Accessed: 2025-07-15. URL: <https://dailynews.co.tz/referee-accuracy-var-still-debatable/>.
- Dong, Xiaofei et al. (2014). “Infrared Dim and Small Target Detecting and Tracking Method Inspired by Human Visual System”. In: *Infrared Physics & Technology* 62, pp. 100–109. DOI: 10.1016/j.infrared.2013.10.010.
- Hawk-Eye Innovations (2024). *Our Technology*. Accessed: 2025-07-13. URL: <https://www.hawkeyeinnovations.com/technology>.
- Heskes, Tom (2006). “Approximate inference: EP and loopy belief propagation”. In: *Probabilistic graphical models*, pp. 205–210.
- Ji, Qiang (2013). *Introduction to 3D Vision*. Accessed: 2025-09-29. URL: https://sites.ecse.rpi.edu/~qji/CV/3dvision_introx.pdf.
- Kamble, Pravin R., Avinash G. Keskar, and Kishor M. Bhurchandi (2019). “Ball Tracking in Sports: A Survey”. In: *Artificial Intelligence Review* 52.3, pp. 1655–1705. DOI: 10.1007/s10462-018-9629-5.
- Koller, Daphne and Nir Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press. ISBN: 9780262013192.
- Koller, Daphne and University Stanford (2025a). *Probabilistic Graphical Models 2: Inference*. Coursera. Accessed: 2025-09-29. URL: <https://www.coursera.org/learn/probabilistic-graphical-models-2-inference/home/module/1>.
- (2025b). *Probabilistic Graphical Models: Representation*. Coursera. Accessed: 2025-09-29. URL: <https://www.coursera.org/learn/probabilistic-graphical-models/home/module/1>.
- Labayen, Mikel et al. (2014). “Accurate Ball Trajectory Tracking and 3D Visualization for Computer-Assisted Sports Broadcast”. In: *Multimedia Tools and Applications* 73.3, pp. 1819–1842. DOI: 10.1007/s11042-013-1667-7.
- Li, Peixia et al. (2018). “Deep Visual Tracking: Review and Experimental Comparison”. In: *Pattern Recognition* 76, pp. 323–338. DOI: 10.1016/j.patcog.2017.11.024.
- Marengoni, M. and D. Stringhini (Aug. 2011). “High level computer vision using OpenCV”. In: *2011 24th SIBGRAPI Conference on Graphics, Patterns, and Images Tutorials*. IEEE, pp. 11–24.

- MetaEye (2025). *Three Levels of Computer Vision*. Accessed: 2025-09-29. URL: https://www.metaeye.co.uk/three-levels-of-computer-vision?utm_source=chatgpt.com.
- Minka, T. P. (2013). “Expectation propagation for approximate Bayesian inference”. In: *arXiv preprint arXiv:1301.2294*. Accessed: 2025-09-29. URL: <https://arxiv.org/abs/1301.2294>.
- Minka, Thomas P (2001). “Expectation propagation for approximate Bayesian inference”. In: *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann, pp. 362–369.
- Mirzaei, Behzad et al. (2023). “Small Object Detection and Tracking: A Comprehensive Review”. In: *Sensors* 23.15. ISSN: 1424-8220. DOI: 10.3390/s23156887. URL: <https://www.mdpi.com/1424-8220/23/15/6887>.
- Mohamad, Mohd et al. (2015). “A review on OpenCV”. In: *Universitas Malaysia Terengganu* 3, p. 1.
- Oliva-Lozano, José María et al. (2022). *Performance Tracking in Professional Football*. Accessed: 2025-07-15. URL: <https://www.isspf.com/articles/performance-tracking-in-professional-football/>.
- OpenCV Team (2023a). *Background Subtraction*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d1/dc5/tutorial_background_subtraction.html.
- (2023b). *Canny Edge Detection*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.
- (2023c). *Canny Edge Detection*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html.
- (2023d). *Changing Colorspaces*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html.
- (2023e). *Connected Components*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html.
- (2023f). *Contour Features*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html.
- (2023g). *Contour Features*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html.
- (2023h). *Contours: Getting Started*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html.
- (2023i). *Hough Circle Transform*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html.
- (2023j). *Image Thresholding*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.
- (2023k). *Image Thresholding*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html.
- (2023l). *Morphological Transformations*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html.

- OpenCV Team (2023m). *Morphological Transformations*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html.
- (2023n). *Morphological Transformations*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html.
- (2023o). *Optical Flow*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html.
- (2023p). *Optical Flow*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/d4/dee/tutorial_optical_flow.html.
- (2023q). *Smoothing Images*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/dc/dd3/tutorial_py_filtering.html.
- (2023r). *Smoothing Images*. Accessed: 2025-10-04. URL: https://docs.opencv.org/4.x/dc/dd3/tutorial_py_filtering.html.
- Pal, Subhadip Kumar et al. (2021). “Deep Learning in Multi-object Detection and Tracking: State of the Art”. In: *Applied Intelligence* 51.9, pp. 6400–6429. DOI: 10.1007/s10489-021-02283-1.
- Pitas, Ioannis (2000). *Digital Image Processing Algorithms and Applications*. John Wiley & Sons. ISBN: 978-0471377391.
- Premier League (2020). *How offsides are determined by VAR*. Accessed: 2025-07-15. URL: <https://www.premierleague.com/en/news/1488423>.
- Sapkota, Rachana et al. (2025). *RF-DETR Object Detection vs YOLOv12: A Study of Transformer-based and CNN-based Architectures for Single-Class and Multi-Class Greenfruit Detection in Complex Orchard Environments Under Label Ambiguity*. arXiv preprint. arXiv: 2504.13099 [cs.CV]. URL: <https://arxiv.org/abs/2504.13099>.
- Soleimanitaleb, Zahra, Mohammad Amin Keyvanrad, and Amir Jafari (Oct. 2019). “Object Tracking Methods: A Review”. In: *2019 9th International Conference on Computer and Knowledge Engineering (ICCCKE)*. IEEE, pp. 282–288. DOI: 10.1109/ICCCKE48569.2019.8964846.
- Stockman, George and Linda G. Shapiro (2001). *Computer Vision*. 1st. USA: Prentice Hall PTR. ISBN: 0130307963.
- Szeliski, Richard (2010). *Computer vision: algorithms and applications*. Springer Science & Business Media.
- TennisNerd (Aug. 2024). *Electronic Line Calling: Hawk-Eye Technology Explained*. Accessed 2025-07-15. URL: <https://www.tennisnerd.net/articles/electronic-line-calling-hawk-eye-technology-explained/39915>.
- The Guardian (2025). *Rise of the machines: amid AI outrage, technology can be a force for good in sport*. Accessed: 2025-07-15. URL: <https://www.theguardian.com/sport/2025/jul/15/rise-of-the-machines-ai-outrage-technology-tennis-sport>.
- Tracab (2024). *TRACAB: The Worlds Leading Sports Tracking System*. Accessed: 2025-07-13. URL: <https://tracab.com>.

- Wiebe, Andrew (Apr. 2013). *MLS Commissioner Don Garber says league won't adopt goal-line technology by 2014*. Archived via the Wayback Machine; accessed 2025-07-15. URL: <https://web.archive.org/web/20150919114824/http://www.mlssoccer.com/news/article/2013/04/25/mls-commissioner-don-garber-says-league-wont-adopt-goal-line-technology-2014>.
- Wong, Yu-Po (Ken) (2016). *Low-cost Tennis Line Call System with Four Webcams*. Tech. rep. Accessed 2025-07-15. Stanford, CA 94305: Department of Applied Physics, Stanford University. URL: https://web.stanford.edu/class/cs231a/prev_projects_2016/final_report_v2.pdf.
- World Intellectual Property Organization (2023). *Goal-line technology – Getting it right*. Accessed 2025-07-15. URL: <https://www.wipo.int/en/web/wipo-magazine/articles/goal-line-technology-getting-it-right-37357>.
- XbotGo (2024). *XbotGo Chameleon – AI-Powered Sports Tracking Camera*. Accessed 2025-07-15. URL: <https://xbotgo.com/pages/xbotgo-chameleon>.
- Zhao, Yuwei et al. (Aug. 2015). “An Overview of Object Detection and Tracking”. In: *2015 IEEE International Conference on Information and Automation (ICIA)*. IEEE, pp. 280–286. DOI: 10.1109/ICInfA.2015.7279340.