Virtual GPS for UAV Navigation in GPS-Denied Environments: Sensor Fusion of Inertial Navigation System and Google Maps-Based Virtual GPS Using Kalman Filtering

GPS Erişimi Kısıtlı Bölgelerde İHA Navigasyonu için Sanal GPS: Kalman Filtrelemesi Kullanılarak Ataletsel Navigasyon Sistemi ve Google Haritalar Tabanlı Sanal GPS'in Sensör Füzyonu

EMRE KESKİ

PROF. DR. CAN ULAŞ DOĞRUER
Supervisor

Submitted to
Graduate School of Science and Engineering of Hacettepe University
as a Partial Fulfillment to the Requirements
for the Award of Degree of Masters
in Mechanical Engineering

2025

ABSTRACT

VIRTUAL GPS FOR UAV NAVIGATION IN GPS-DENIED ENVIRONMENTS: SENSOR FUSION OF INERTIAL NAVIGATION SYSTEM AND GOOGLE MAPS-BASED VIRTUAL GPS USING KALMAN FILTERING

Emre KESKİ

Degree of Masters, Mechanical Engineering
Supervisor: Prof. Dr. Can Ulaş DOĞRUER
September 2025, 123 pages

Modern unmanned aerial vehicles (UAV) have become essential tools across many fields yet they share one major weakness: a complete dependency on Global Positioning System (GPS) signals for accurate positioning. This dependency often makes UAVs invulnerable because GPS signals can become unavailable due to satellite positions, jamming, or failures etc. In such cases, UAVs can lose their accurate navigation capability, which often results in vehicle loss. This research concentrates on a solution to overcome these limitations by using Virtual GPS approach. The proposed system provides accurate global positioning capabilities without requiring satellite-based signals. The approach replaces GPS integration with an innovative visual-inertial localization system that uses publicly available satellite imagery from Google Maps as a global reference dataset.

The simulation is built around three integrated components. First, a virtual UAV camera view is generated from Google Maps satellite imagery using geometric transformations. Second, features are extracted from both the virtual UAV camera and satellite imagery using the Speeded-Up Robust Features (SURF) algorithm. Feature matching between the two image sources provides input to the Random Sample Consensus (RANSAC) algorithm to establish reliable correspondences while systematically rejecting false

matches. The validated correspondences enable the computation of similarity transformations, which directly yield estimates for the UAV's global position and heading. The third component is sensor fusion, which provides a continuous localization solution by integrating the low-frequency visual position estimates with high-frequency inertial measurements through an Extended Kalman Filter (EKF). The EKF maintains a six-dimensional state vector, and visual measurements are validated using Mahalanobis distance calculations to reject outliers. The control component uses Nonlinear Model Predictive Control (NMPC) for trajectory tracking. The NMPC algorithm computes optimal control inputs by minimizing cost functions that penalize tracking errors, and it adaptively manipulates control weights based on position and heading errors to maintain optimal performance.

Simulations using various trajectories demonstrate that the proposed system is effective and reliable. The results indicate that the system provides stable and accurate localization even when subjected to sensor noise, IMU bias, and temporary visual ambiguities. This work represents a major step forward in making UAVs truly independent of GPS by combining computer vision, statistical estimation theory, and optimal control. Therefore, it offers a practical solution that could enhance UAV operations in places where GPS signals are unavailable or unreliable.

Keywords: Sensor Fusion, Global Positioning, Unmanned Aerial Vehicle, Model Predictive Control, Extended Kalman Filter

ÖZET

GPS ERIŞIMI KISITLI BÖLGELERDE İHA NAVIGASYONU IÇIN SANAL GPS: KALMAN FILTRELEMESI KULLANILARAK ATALETSEL NAVIGASYON SISTEMI VE GOOGLE HARITALAR TABANLI SANAL GPS'IN SENSÖR FÜZYONU

Emre KESKİ

Yüksek Lisans, Makina Mühendisliği Bölümü
Tez Danışmanı: Prof. Dr. Can Ulaş DOĞRUER
Eylül 2025, 123 sayfa

Modern insansız hava araçları (İHA), birçok alanda vazgeçilmez araçlar haline gelmiş olsalar da önemli bir zayıflığı paylaşmaktadırlar: doğru konum bilgisi için Küresel Konumlandırma Sistemi (GPS) sinyallerine olan tam bağımlılıkları. Bu bağımlılık, GPS sinyallerinin karıştırma, sistem arızaları veya uydu pozisyonları gibi nedenlerle kullanılamaz hale gelebilmesi sebebiyle İHA'ları savunmasız bırakmaktadır. Bu gibi durumlarda İHA'lar, seyrüsefer kabiliyetlerini yitirerek genellikle aracın kaybedilmesine yol açan sonuçlarla karşılaşabilirler. Bu araştırma, uydu tabanlı sinyallere ihtiyaç duymadan doğru küresel konumlandırma sağlayan bir Sanal GPS yaklaşımı ile bu sınırlamanın üstesinden gelmeye odaklanmaktadır. Sistem, GPS'i, Google Haritalar gibi kamuya açık uydu görüntülerini küresel bir referans olarak kullanan görsel-ataletsel bir yerelleştirme yöntemiyle değiştirmektedir.

Simülasyon, üç entegre bileşen etrafında kurulmuştur. İlk olarak, geometrik dönüşümler kullanılarak Google Haritalar uydu görüntülerinden sanal bir İHA kamera görüntüsü oluşturulur. İkinci olarak, hem sanal İHA kamerasından hem de uydu görüntülerinden, ölçek ve aydınlatma değişimlerine karşı dayanıklılık sunan Hızlandırılmış Sağlam

Özellikler (SURF) algoritması kullanılarak özellikler çıkarılır. İki görüntü kaynağı arasındaki özellik eşleştirmesi, hatalı eşleşmeleri sistematik olarak reddederken güvenilir karşılıklar kurmak için Rastgele Örneklem Mutabakatı (RANSAC) algoritmasına girdi sağlar. Doğrulanan bu karşılıklar, İHA'nın küresel konumu ve yönelimi için doğrudan tahminler üreten bir benzerlik dönüşümünün hesaplanmasını mümkün kılar.

Üçüncü bileşen olan sensör füzyonu, düşük frekanslı görsel konum tahminlerini yüksek frekanslı ataletsel ölçümlerle bir Genişletilmiş Kalman Filtresi (EKF) aracılığıyla entegre ederek sürekli bir yerelleştirme çözümü sunar. EKF, altı boyutlu bir durum vektörünü takip eder ve görsel ölçümler, aykırı değerleri reddetmek için Mahalanobis uzaklığı kullanılarak doğrulanır. Kontrol bileşeni ise yörünge takibi için Doğrusal Olmayan Model Öngörülü Kontrol (NMPC) kullanır. NMPC algoritması, takip hatalarını cezalandıran maliyet fonksiyonlarını en aza indirerek en uygun kontrol girdilerini hesaplar ve en iyi performansı sürdürmek için konum ve yönelim hatalarına göre kontrol ağırlıklarını uyarlanabilir bir şekilde ayarlar.

Çeşitli yörüngeler kullanılarak yapılan simülasyonlar, önerilen sistemin etkin ve güvenilir olduğunu göstermektedir. Sonuçlar, sensör gürültüsü, IMU yanlılığı ve geçici görsel belirsizlikler hesaba katıldığında dahi sistemin kararlı ve doğru bir yerelleştirme sağladığını ortaya koymaktadır. Bu çalışma; görüntü işleme, istatistiksel tahmin teorisi ve en uygun kontrolü birleştirerek İHA'ları GPS'ten gerçek anlamda bağımsız hale getirme yolunda önemli bir adım atmaktadır. Dolayısıyla, GPS sinyallerinin mevcut olmadığı veya güvenilir olmadığı yerlerde İHA operasyonlarını geliştirebilecek pratik bir çözüm sunmaktadır.

Anahtar Kelimeler: Sensör Füzyonu, Küresel Konumlandırma, İnsansız Hava Aracı, Model Öngörülü Kontrol, Genişletilmiş Kalman Filtresi

ACKNOWLEDGMENTS

I would like to express my deepest and most sincere gratitude to my supervisor, Prof. Dr. Can Ulaş DOĞRUER, for his invaluable guidance, unwavering support, and constant encouragement throughout this thesis research. His vision, expertise, and patience were instrumental in shaping this work and bringing it to completion.

I am also grateful to TÜBİTAK SAGE for providing me with the valuable opportunities and supportive environment that made this research possible. I would like to thank my managers and colleagues for their understanding and encouragement during this period.

Lastly, my deepest and most special thanks go to my beloved wife, Cemile KESKİ. Her amazing and endless support, patience, and belief in me have been my greatest source of motivation throughout this challenging journey. This accomplishment would not have been possible without her.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DOF	Degrees-of-Freedom
BRIEF	Binary Robust Independent Elementary Features
FAST	Features from Accelerated Segment Test
EKF	Extended Kalman Filter
EQN	Equation
GNSS	Global Navigation Satellite Systems
GPS	Global Positioning System
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
MAE	Mean Absolute Error
MEMS	Micro-Electro-Mechanical Systems
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
ORB	Oriented FAST and Rotated BRIEF
RANSAC	Random Sample Consensus
RMSE	Root Mean Square Error
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SQP	Sequential Quadratic Programming
SURF	Speeded-Up Robust Features
UAV	Unmanned Aerial Vehicle
UKF	Unscented Kalman Filter
vGPS	Virtual GPS
VINS	Visual-Inertial Navigation System
vSLAM	Visual SLAM
VTOL	Vertical Take-Off and Landing

1. INTRODUCTION

The rapid advancement of unmanned aerial vehicle technology has fundamentally transformed numerous industries and operational paradigms over the past two decades. These platforms have evolved from specialized military assets to easily accessible tools which are supporting diverse applications ranging from package delivery and precision agriculture to search and rescue operations and critical infrastructure inspection (Groves, 2013). However, this widespread adoption has simultaneously revealed a fundamental vulnerability that creates significant risks to mission success and operational safety across all application domains.

Contemporary UAV systems exhibit an almost universal dependence on Global Positioning System signals for their primary navigation and positioning functions. This dependency emerged naturally during the early development phases of autonomous flight systems when GPS represented the most practical and cost-effective solution for accurate global positioning (Kaplan & Hegarty, 2017). The technology offered excellent positioning accuracy, worldwide coverage, and relatively straightforward integration with existing flight control architectures. Consequently, most UAV designs were designed according to the fundamental assumption that GPS signals would remain continuously available throughout normal operations.

This design assumption has proven increasingly problematic as UAV operations have expanded into more challenging environments and mission profiles. Urban environments present difficulties where tall buildings create signal shadows and multipath propagation effects that can severely degrade GPS accuracy or cause complete signal loss for extended periods (Hofmann-Wellenhof et al., 2008). Military and security applications face additional challenges from intentional GPS jamming or spoofing attacks by adversaries seeking to disable or misdirect autonomous systems (Psiaki & Humphreys, 2016).

The consequences of GPS signal loss extend far beyond operational inconvenience to represent fundamental safety and security concerns that can result in catastrophic failures. When GPS signals become unavailable, most UAV platforms automatically revert to basic stabilization modes that maintain attitude control but provide no global positioning capability. While the aircraft can maintain stable flight characteristics, it loses awareness

of its location relative to mission objectives, obstacles, terrain features, or designated safe landing areas.

1.1. Problem Definition

The fundamental challenge considered in this research involves developing a navigation system capable of providing an accurate and reliable positioning likewise GPS, yet without dependence on GPS signals or other external interaction that can be compromised, jammed, spoofed, or rendered unavailable. This requirement covers several technical problems that must be solved through an integrated approach to achieving practical operational capability for real-world UAV applications.

Current approaches to GPS-denied navigation each exhibit significant limitations that prevent widespread adoption for UAV applications requiring sustained accurate positioning. Inertial navigation systems can provide short-term positioning through mathematical integration of accelerometer and gyroscope measurements, but they suffer from cumulative drift that renders them unsuitable for extended missions (Titterton et al., 2004). Since a high-quality inertial measurement unit (IMU) on the market accumulates significant errors as the time passes, this makes the individual usage of IMUs insufficient for sustained and accurate global positioning required applications.

Visual odometry techniques estimate vehicle motion by analyzing the visible movement of distinctive features across sequential camera frames, achieving impressive short-term accuracy that can exceed the performance of inertial sensors (Scaramuzza & Fraundorfer, 2011). However, this approach suffer from completely failing in environments with poor visual texture, challenging illumination conditions, or rapid lighting changes. More critically, visual odometry provides only relative motion estimates without the ability to determine absolute global position because of the lack the external coordinate knowledge.

Simultaneous Localization and Mapping (SLAM) approaches address the relative positioning limitation by constructing detailed maps of unknown environments while simultaneously tracking vehicle position within those maps (Cadena et al., 2016). These applications work effectively for local navigation and reconnaissance missions or returning previous locations in small areas but cannot provide the global positioning which is required for UAV navigation to specific coordinates, operations in large areas

since the mapping computations have become extremely large, or coordination with other vehicles operating in the same operation area.

The technical requirements for a GPS-alternate solution extend beyond basic positioning accuracy to encompass real-time performance constraints, computational efficiency, environmental robustness, and integration with existing flight control systems. The solution must operate effectively on embedded processors with limited computational resources while maintaining update rates sufficient for stable flight control and responsive maneuvering (Savage, 2000). In addition, the system must also be reliable in diverse environmental conditions including weather effects, seasonal changes on the ground which causes difference in UAV/satellite images, and temporary visual obstructions which causes significant visual localization errors.

1.2. Objective and Research Goals

The objective of this research is to develop an extensive navigation system that achieves positioning accuracy compared to GPS while operating independently in case of unreliable satellite signals in GPS-denied environments. Therefore, the objective requires handling several technical challenges through an integrated approach that combines computer vision, sensor fusion, and control theory.

The first research goal focuses on creating a robust visual localization system that can accurately and reliably match the images of the UAV camera and satellite images which is obtained from Google Maps or any publicly available sources. This requires solving the challenging problem of establishing correspondence between images captured from looking down built in UAV camera and the satellite imagery in different environmental conditions (in real-world application) (Patel et al., 2020). Moreover, the system must identify and track distinctive visual landmarks i.e. features in the images that remain recognizable despite substantial variations in scale, perspective, lighting, and seasonal appearance. In the research this problem is approached by creating simulation images that can be placed as virtual UAV imagery to decrease those challenges and focus on further goals.

Another research goal addresses optimal sensor fusion between relatively low frequency visual position estimates and continuous inertial measurements to provide smooth,

accurate navigation solutions suitable for UAV operations. Visual localization typically provides accurate but infrequent updates, often more limited when environmental conditions permit reliable feature matching (Li & Mourikis, 2013). Inertial sensors deliver continuous measurements at much higher frequencies but significantly accumulate drift due to noise and bias effects. The fusion algorithm systematically combines these complementary data sources to provide continuous, accurate navigation estimates while maintaining appropriate uncertainty bounds.

Obtaining the requirements of real-time applications of the system is the third goal of this research, addressing the critical gap between academic algorithm development and practical implementation requirements for operational UAV systems. Many research approaches demonstrate excellent performance in post-processing applications but require computational resources that exceed the capabilities of typical UAV platforms or processing times incompatible with real-time flight control requirements (Mourikis & Roumeliotis, 2007). The system must operate within the constraints imposed by embedded processors, limited memory, and strict power consumption requirements while maintaining the update rates necessary for stable autonomous flight operations.

Robustness and fault tolerance are subjects of the fourth research goal, ensuring the system maintains acceptable performance when subcomponents experience degraded operation or temporary failure conditions. Visual processing may become unreliable due to poor lighting conditions, camera obstruction, motion blur, or low-texture environments that provide insufficient distinctive features for reliable matching (Davison et al., 2007). The system must continue providing navigation solutions using alternative sensors and algorithms until normal visual processing capabilities can be restored.

1.3. Expected Contributions

This research makes several significant contributions to autonomous navigation and localization that advance theoretical understanding for UAV operations that will be held on GPS-denied environments. These contributions address critical gaps in existing approaches while providing practical solutions that can be implemented on UAV navigation systems.

The first major contribution demonstrates that freely available satellite imagery from Google Maps contains sufficient information content to support accurate UAV localization when processed using appropriate computer vision algorithms and geometric transformation techniques. Previous approaches to satellite-based navigation often required expensive, high-resolution imagery, extensive preprocessing of reference datasets, or specialized equipment that limited practical applicability (Ren et al., 2024). This work proves that standard satellite maps available worldwide at no cost can support meter-level positioning accuracy for practical UAV applications.

The second contribution addresses the specific technical challenges associated with cross-view image matching between aerial and satellite perspectives through geometric transformation and feature correspondence techniques. This problem involves substantial differences in viewing geometry, scale, imaging conditions, and temporal variations that cause traditional feature matching algorithms to fail or produce unreliable results (Zhang & Singh, 2014). This research develops and demonstrates innovative approach for obtaining reliable correspondences comparing the virtual UAV imagery and satellite images, including transformation strategies, robust feature detection methods, and statistical outlier rejection techniques.

The third contribution involves designing and implementing a tightly coupled EKF framework that effectively integrates visual localization updates with continuous localization obtaining from high-frequency inertial measurements while maintaining computational efficiency suitable for embedded platforms. While EKF-based sensor fusion represents an established theoretical framework, the specific challenges of combining intermittent visual updates with continuous inertial data require careful attention to uncertainty modeling, measurement validation, and state prediction accuracy (Mourikis & Roumeliotis, 2007).

The fourth contribution includes developing robust statistical validation techniques that enable reliable operation in challenging situations where sensor/visual measurements may be corrupted by outliers, exhibit non-Gaussian characteristics, or become temporarily unavailable due to environmental conditions (Bar-Shalom et al., 2001). Visual navigation systems are inherently susceptible to false feature matches, incorrect

correspondences, and corruption that can severely degrade estimation performance if not properly detected and rejected.

The fifth contribution involves integrating Nonlinear Model Predictive Control with the visual-inertial navigation system to demonstrate closed-loop trajectory tracking capabilities that leverage improved state estimation for enhanced flight performance. The NMPC implementation accounts for UAV dynamic constraints, control input limitations, and trajectory tracking objectives while adapting to real-time navigation uncertainty estimates (Groves, 2013).

1.4. Proposed Solutions

The technical approach represents a fundamental departure from traditional UAV navigation strategies that attempt to improve GPS reception quality or develop more accurate inertial sensors. Instead, this research creates an entirely independent navigation paradigm based on visual landmark recognition, intelligent multi-sensor fusion techniques, and advanced control integration that eliminates dependence on external signals while maintaining operational effectiveness.

The core innovation centers on the Virtual GPS module, which functions as a sophisticated computer vision system that extracts global positioning information through systematic comparison of real-time UAV camera imagery (created virtually in the study) with satellite images from Google Maps. The process begins when the UAV's onboard camera captures imagery of the terrain below during normal flight operations (Patel et al., 2020). This imagery undergoes processing through the SURF feature detection algorithm, which identifies distinctive visual landmarks such as building corners, road intersections, vegetation boundaries, unique terrain patterns, or other features that should remain recognizable across different viewing conditions (Bay et al., 2008).

Simultaneously, the system has assumed with integrated Google Maps satellite imagery covering the UAV's mission area or approximate location based on the most recent position estimation taken from the navigation algorithm. This satellite imagery undergoes predefined geometric transformation process to simulate the expected appearance by considering UAV's current perspective, UAV’s estimated altitude (taken as constant in the study), camera orientation (considered as looking down built-in UAV camera), viewing

geometry, and other factors that influence visual correspondence between aerial and satellite imagery (Ren et al., 2024).

Both the UAV's camera imagery and the satellite imagery are processed individually through an identical SURF feature detection algorithm, to create corresponding sets of features from each image source that can be systematically compared in further sections of the analysis. The system then attempts to establish reliable correspondences between these feature sets using the RANSAC algorithm, which provides a robust statistical framework for identifying valid matches while systematically rejecting false correspondences that could lead to significant localization errors (Fischler & Bolles, 1981).

Once reliable feature correspondences are established between UAV and satellite imagery, the system computes a similarity transformation that describes the mathematical relationship between the UAV's current view and the satellite reference coordinate system. This transformation directly provides estimates of the UAV's global position and heading relative to the coordinate system defined by the satellite maps, effectively creating a GPS-independent positioning solution that maintains global reference frame awareness.

However, visual localization alone proves insufficient for smooth flight control applications because camera-based position estimates typically update at relatively low frequencies and can become temporarily unavailable due to environmental conditions. This limitation necessitates the integration of complementary inertial navigation capabilities that can provide continuous motion estimates during visual outages (Li & Mourikis, 2013).

The Extended Kalman Filter implementation creates a unified state estimation framework that leverages the complementary characteristics of visual and inertial measurements through optimal statistical combination techniques. The EKF maintains a probabilistic model of the UAV's complete state vector including position, velocity, and orientation, along with associated uncertainty estimates that reflect current reliability of state knowledge and guide measurement weighting decisions (Welch & Bishop, 2006).

1.5. Content of the Thesis

This thesis is organized to provide an understandable expression from theoretical foundations through detailed implementation to comprehensive evaluation and validation. Therefore, each section is built systematically upon previous material while providing sufficient technical detail to enable reproduction of research results.

Section 2 presents a comprehensive literature survey that examines the current status of researches in regard of UAV navigation, computer vision, sensor fusion technologies, and related fields that contribute to the theoretical and practical foundation for the proposed system. The study systematically identifies gaps in existing approaches, establishes theoretical context for research contributions, and provides analysis of previous work in localization, navigation, robust estimation techniques, and control system integration.

Section 3 provides detailed descriptions of methodology and implementation of the Virtual GPS system, including complete mathematical formulations, algorithm designs, implementation strategies, and engineering considerations for each component of the integrated navigation framework. This section presents sufficient technical details to enable other researchers to reproduce the work while highlighting key design decisions, engineering trade-offs, performance considerations, and implementation challenges that influence overall system performance and practical applicability.

Section 4 presents comprehensive evaluation results that demonstrate performance characteristics of the developed system under various operational conditions, mission scenarios, environmental challenges, and failure modes representative of real-world deployment conditions. The evaluation encompasses accuracy assessments, robustness analysis, computational performance characterization, and comparative studies across diverse environmental conditions and mission profiles.

Section 5 concludes the thesis by summarizing key research contributions, discussing their implications for the broader field of autonomous navigation, addressing limitations of the current approach, and identifying promising directions for future research and development efforts. The section examines potential impact of this work on practical UAV applications, the broader autonomous systems community, and related fields that could benefit from similar approaches to GPS-independent navigation and positioning.

2. LITERATURE REVIEW

Developing GPS-independent navigation systems for unmanned aerial vehicles requires integration of knowledge from several distinct research domains that have traditionally operated with limited cross-pollination and different performance priorities. Computer vision researchers have developed sophisticated algorithms for object recognition, motion tracking, and scene understanding, but their work often assumes unlimited computational resources and controlled environmental conditions (Lowe, 2004). Robotics researchers focus on real-time performance constraints and practical implementation considerations but may not incorporate the latest advances in computer vision techniques. Aerospace engineers possess a deep understanding of flight dynamics and control systems but may lack familiarity with modern computer vision algorithms or sensor fusion techniques.

2.1. Visual Navigation Techniques for UAV Localization

The concept of using visual information for navigation predates the technology in millennia, with early human navigators relying on celestial observations, landmark recognition, and environmental cues to determine position and direction. The automation of visual navigation through digital image processing represents a natural evolution of these traditional approaches, enhanced by computational capabilities that far exceed human perceptual limitations while maintaining the fundamental principle of using visual landmarks for position determination.

Robust feature detection algorithms marked a critical breakthrough in computer vision that enabled practical visual navigation applications for autonomous systems. The Scale-Invariant Feature Transform (SIFT) introduced by Lowe (2004) demonstrated that distinctive image features could be reliably detected and matched across images which are captured under significantly different viewing conditions, scale variations, rotation changes, and illumination differences. SIFT features bring remarkable robustness to geometric and photometric transformations, making them suitable for tracking visual landmarks over time with establishing correspondences between captured images from different viewpoints.

The major challenge of SIFT is that the SIFT computations often need high computation power. For this reason, real-time applications of SIFT struggle a lot due to its complexity,

especially when running on embedded processors. UAV platforms have already some challenges in this manner. These systems operate under strict constraints for power consumption, weight, and cost. Such limitations automatically restrict the computational resources available for image processing tasks. This problem was recognized by researchers and has begun developing faster alternatives. The goal was to maintain SIFT's robustness while significantly reducing the computational burden.

The SURF algorithm has come forward as one of the most successful attempts to overcome computational limitations of SIFT while maintaining acceptable robustness characteristics for practical applications (Bay et al., 2008). SURF utilizes integral image representations and Hessian-based feature detection to achieve substantial performance improvements compared to SIFT processing. The algorithm's computational efficiency makes it particularly suitable for UAV applications since there are limitations in processor capabilities and real-time calculations.

Recent work by Ren et al. (2024) specifically addresses UAV localization using SURF features matched against Google Maps satellite imagery, demonstrating practical feasibility of using publicly available satellite maps as reference for UAV navigation applications. Their results show that SURF features can establish reliable correspondences between UAV camera and satellite imagery despite significant differences in viewing perspective, scale, and imaging conditions.

Visual odometry techniques estimate vehicle motion by analyzing apparent movement of distinctive features between consecutive camera frames. This analysis enables establishing geometric relationships between feature motion in the image plane and camera motion in three-dimensional space. When sufficient features can be tracked reliably across multiple frames, camera motion can be estimated through geometric calculations that provide relative position and orientation changes between consecutive/predefined time steps (Scaramuzza & Fraundorfer, 2011).

The basic limitations of visual odometry approaches come forward from their reliance on relative measurements that largely accumulated drift over extended time periods. These errors are caused by measurement noise, feature tracking errors, and absence of global reference information etc. While these techniques can achieve remarkable accuracy for

short-term motion estimation, often exceeding performance of expensive inertial sensors in terms of drift characteristics and noise levels, they cannot provide the absolute positioning in required levels for many practical applications.

Cross-view image matching represents one of the most challenging problems in computer vision, particularly when attempting to establish correspondences between images captured from dramatically different viewpoints such as aerial and satellite perspectives (Zhang & Singh, 2014). Traditional feature matching algorithms often fail when dealing with the substantial geometric distortions, scale differences, and appearance variations inherent in comparing images captured from different altitudes and viewing angles.

2.2. Inertial Navigation Systems and Sensor Fusion

Inertial navigation represents one of the oldest autonomous navigation techniques, with theoretical foundations dating back to Newton's laws of motion and practical implementations that have evolved continuously over several centuries. The basic principle involves measuring all forces acting on a body and using those measurements to calculate position through mathematical integration, providing a self-contained navigation solution that requires no external references or signals (Savage, 2000).

Modern inertial measurement units rely heavily on MEMS technology that has revolutionized the size, weight, cost, and power consumption characteristics of inertial sensors over the past two decades (Barbour & Schmidt, 2001). Contemporary MEMS accelerometers and gyroscopes are orders of magnitude smaller and less expensive than their mechanical predecessors, making them practical for small UAV applications where size, weight, cost, and power consumption constraints are critical design considerations.

The miniaturization enabled by MEMS technology comes with significant performance trade-offs that cannot be ignored in practical navigation applications. MEMS sensors typically exhibit higher noise levels, greater bias instability, increased temperature sensitivity, and reduced long-term stability compared to larger, more expensive alternatives (Allan, 1966). These characteristics can lead to navigation errors that grow rapidly over time. This creates instability over inertial navigation which causes unsuitable applications which require sustained accuracy.

The fundamental challenge in inertial navigation comes forward from the mathematical integration process which is required to convert acceleration measurements into position calculations via integration steps. This process requires two successive integrations: first to obtain velocity from acceleration measurements, then to obtain position from velocity estimates (Titterton et al., 2004). Double integration causes even small errors in acceleration measurements become amplified. When it is considered as long-time calculations of these measurements, position errors grow.

The Kalman filter provides a mathematically optimal framework for combining measurements from multiple sensors with different characteristics, error properties, and rates through statistical estimation techniques (Kalman, 1960). The algorithm considers weights for different measurements. These weights are defined according to their estimated reliability and uncertainty. Therefore, compared to simple averaging or other fusion techniques, the filter provides increased performance.

The EKF -as it is seen- extends the basic Kalman filter to handle nonlinear systems such as UAV navigation. In the UAV navigation there are relationships between sensor measurements and vehicle state. That relationship includes complex trigonometric functions, coordinate transformations, and nonlinear dynamics (Welch & Bishop, 2006). EKF implementations for UAV navigation must deal with nonlinearities in attitude representation, coupling between translational and rotational motion, and geometric relationships between sensor measurements and state variables. Li and Mourikis (2013) developed one of the most influential approaches to visual-inertial navigation using a tightly coupled EKF formulation that recognizes fundamental correlation between visual and inertial measurements. Their key insight recognized that visual and inertial measurements are fundamentally correlated because they observe the same underlying vehicle motion from different perspectives, enabling more accurate state estimation through proper modeling.

The Multi-State Constraint Kalman Filter brings an advanced approach that leads some limitations of traditional EKF implementations by maintaining multiple camera poses in the state vector rather than just the current vehicle state (Mourikis & Roumeliotis, 2007). It is thought that this approach can improve the navigation estimation and reduce the

effects of linearization errors, because these errors can easily degrade EKF performance in dynamic scenarios.

2.3. Simultaneous Localization and Mapping Approaches

SLAM represents one of the most fundamental and challenging problems in mobile robotics by constructing maps of environments which it is inside. In the process, the algorithm simultaneously uses those maps to determine the robot's location within the map. This creates a circular dependency that has fascinated researchers for decades and led to numerous innovative algorithmic solutions and practical implementations (Durrant-Whyte & Bailey, 2006). Early developed visual SLAM (vSLAM) systems showed that the fundamental problem could be solved using only visual information from cameras without requiring expensive laser range finders (lidar sensor), or any other active sensors. Moreover, MonoSLAM proved that a single camera could support real-time SLAM, although the approach was limited to relatively small environments like indoors due to computational constraints (Davison et al., 2007).

The Parallel Tracking and Mapping system offers solutions about scalability limitations in earlier approaches by separating tracking and mapping functions into parallel computational threads that could operate at different frequencies and with different computational priorities (Klein & Murray, 2007). This method allowed the system to have real-time tracking performance for UAV’s control while building increasingly detailed maps in the background without compromising computational power which is needed for real-time calculations.

ORB-SLAM represents the most successful visual SLAM system developed to date. It provides robust operation across diverse environments with potential loop closure detection, localization capabilities, and map management features (Mur-Artal et al., 2015). By using ORB features in SLAM operations, a balance between computational efficiency and robustness in various environments was successfully achieved.

During the development of SLAM approaches, loop closure detection got involved various SLAM systems to eliminate the accumulated drift via recognition of previously visited locations. This brings establishment of constraints that improve global map consistency. When a SLAM system recognizes that it has returned to a previously mapped

area, it can establish geometric constraints between the current location and the previously visited location, enabling global optimization algorithms to correct accumulated errors (Galvez-Lopez & Tardos, 2012).

Beside all of this knowledge, SLAM approaches fundamentally cannot provide global positioning awareness. That's because of the lack of external reference information. While these systems are good at building detailed local maps and tracking vehicle position within those maps, they cannot determine absolute coordinates or enable navigation to specific global locations without additional information sources. The recreated map of the environment begins from the ashes.

2.4. Deep Learning and Machine Learning Approaches

The deep learning revolution that began in the early 2010s has impacted computer vision and robotics research, including UAV navigation applications and autonomous systems development. The breakthrough occurred when Krizhevsky et al. (2012) demonstrated that deep convolutional neural networks could achieve dramatically superior performance on image classification tasks compared to traditional computer vision approaches.

PoseNet, introduced by Kendall et al. (2015), represented one of the first successful applications of deep learning to camera localization problems in indoor environments. The system learns to directly process camera poses from RGB images through end-to-end training, eliminating the need for explicit feature extraction, matching, and geometric computation steps that is known from traditional approaches. However, questions still remain regarding PoseNet's performance in outdoor environments, its ability to generalize to locations not encountered during training, and its robustness to environmental variations such as lighting changes, weather effects, or seasonal appearance differences. The system requires extensive training data from the target environment, which may not be practical for many UAV applications (Kendall et al., 2015).

The development of learned feature extractors and matchers have become an active research area. With several approaches superior performance is demonstrated compared to traditional hand-crafted features such as SIFT, SURF, and ORB in specific application domains. SuperPoint -one of the methods developed in this area- provides a self-supervised approach for learning interest point detection and description that achieves

state-of-the-art performance on various computer vision benchmarks (DeTone et al., 2018). Similarly, SuperGlue addresses the feature matching problem through a graph neural network approach. It can learn to establish correspondences between images more effectively than traditional matching algorithms based on nearest neighbor search or ratio tests (Sarlin et al., 2020). SuperGlue comes forward in challenging scenarios involving repetitive textures, illumination changes, viewpoint variations, and some other conditions that often cause problems for traditional matching approaches.

2.5. Multi-Sensor Fusion and Advanced Estimation Techniques

Multi-sensor measurement fusion constitutes a major challenge in modern navigation system development. This difficulty arises because sophisticated mathematical techniques are essential for handling diverse sensor properties: update rates, noise characteristics, and reliability parameters. Theoretical foundations build upon Bayesian estimation theory, which establishes a principled methodology for integrating prior knowledge with new measurements (Bar-Shalom et al., 2001). This approach enables information combination from various sources having different characteristics—such as reliability levels and temporal properties—while managing uncertainty throughout the process. Measurement uncertainty stems from sensor noise, environmental conditions, and data quality factors. Consequently, Bayesian methods achieve optimal measurement integration while accounting for individual uncertainties and correlations.

Kalman filtering remains the most widely used approach for linear sensor fusion problems, providing mathematically optimal estimates under assumptions of linear system dynamics, Gaussian noise distributions, and assumed system parameters differences. The original Kalman filter addresses linear systems with additive Gaussian noise, while the EKF extends these capabilities to nonlinear systems through linearization around current state estimates (Kalman, 1960). On the other hand, the Unscented Kalman Filter (UKF) represents some limitations of EKF linearization through a deterministic sampling approach that can more accurately propagate uncertainty through nonlinear transformations without requiring analytical derivatives (Julier & Uhlmann, 1997). The UKF uses a set of sample points to capture the mean and covariance of the state distribution. Then projects these points through nonlinear system dynamics to compute updated estimates.

On the other hand, factor graph optimization represents a compelling alternative to traditional filtering methods. It provides the ability to perform global optimization over trajectories while incorporating various types of constraints, measurements, and prior information (Dellaert & Kaess, 2017). Unlike conventional filters that handle measurements one by one, this methodology evaluates all available data points and identifies optimum trajectory explanation.

To conclude, robust estimation techniques which are developing constantly have become increasingly important as the navigation systems are deployed in real-world environments. In these areas sensor measurements may be corrupted by outliers, exhibit non-Gaussian noise characteristics, or become temporarily unreliable. The Mahalanobis distance provides a statistical test for evaluating consistency of measurements with current state estimates, enabling rejection of outliers that could severely degrade estimation performance (Mahalanobis, 1936).

3. METHODOLOGY

3.1. Computer Vision-Based Virtual GPS using Google Maps for UAV Navigation

3.1.1. Overview

This section presents a vision-based Virtual GPS solution that leverages high-resolution satellite imagery specifically from Google Maps as a static global reference. The UAV is assumed to be equipped with a downward-facing camera, capturing real-time images of the terrain below. These images are then matched against the satellite reference map to estimate the UAV's global pose (position and heading).

The approach follows an integrated process as listed below:

- ✓ Preprocessing the satellite image to obtain a grayscale, feature-rich map
- ✓ Simulating UAV motion along various trajectories (e.g., figure-eight, circular, sinusoidal) to provide a ground truth benchmark
- ✓ Extracting image patches from the satellite map corresponding to the UAV's camera view using geometric transformations (rotation, scaling, cropping)
- ✓ Applying Speeded-Up Robust Features (SURF) to detect and match keypoints between the UAV and satellite images
- ✓ Estimating a similarity transformation to recover the UAV's position and orientation
- ✓ Refining the pose through bias correction based on centroid alignment and feature distribution

This vision-based localization system is designed to operate in real-time or near real-time and does not rely on GPS or pre-installed visual landmarks. The technique is modular and sensor-agnostic, meaning it can later be integrated with inertial navigation systems (INS) using filtering frameworks such as the Extended Kalman Filter (EKF).

The following subsections describe each stage of the algorithm in detail, including the mathematical models, implementation in MATLAB, and performance evaluation using synthetic UAV trajectories. This visual localization module forms the foundation for the hybrid Visual-Inertial Navigation System (VINS) developed in subsequent sections.

3.1.2. Satellite Imagery Preprocessing

Accurate visual localization begins with the acquisition and preprocessing of high-resolution satellite imagery, which serves as a static global reference map for the UAV navigation system. The preprocessing pipeline includes color-space transformation, dimensionality reduction, and selective cropping to isolate pertinent sections of the map. This step ensures consistency in feature detection and reduces computational complexity during image matching.

3.1.2.1. Image Acquisition and Grayscale Conversion

The original satellite image, acquired from Google Maps and stored in RGB format, is converted into a grayscale representation. Grayscale conversion is a common preprocessing step in computer vision tasks that do not require color information. This reduction as given in Eqn. 3.1, simplifies subsequent computations while retaining essential texture and contrast features necessary for keypoint detection:

```
matlab
satelliteImg = imread('Satellite_Imagery.jpg');
satelliteGray = rgb2gray(satelliteImg);
[h, w] = size(satelliteGray);
```

$$I_{gray}(x, y) = 0.2989 \cdot R(x, y) + 0.5870 \cdot G(x, y) + 0.1140 \cdot B(x, y) \qquad (3.1)$$

These weights reflect the human visual system's sensitivity to green more than to red or blue. MATLAB's 'rgb2gray' function internally implements this transformation.

3.1.2.2. Image Dimensions and Reference Map Size

The image dimensions are stored as $h\,x\,w$, which are essential for later tasks like boundary checking and mask generation. These are referenced throughout the image transformation and feature matching pipeline.

```
matlab
[h, w] = size(satelliteGray);
```

This information is also critical for constructing binary masks during UAV region extraction and verifying that image operations do not exceed matrix bounds.

3.1.2.3. Cropping UAV-Related Regions

To simulate a UAV's onboard image capture and enable image registration, subregions of the global map are extracted. These subregions mimic the UAV's visual field at specific

poses. This process involves defining a bounding box centered at the UAV's estimated location and oriented according to the UAV's heading. The size of the crop is defined by:

```
matlab
cropSize = [100,100]; % width and height in pixels
```

At each simulation step, a rotated rectangular region of interest (ROI) is dynamically extracted based on the UAV's pose and scaled to reflect altitude variation. The use of 'poly2mask' creates a logical binary mask that isolates the UAV's image section:

```
matlab
mask = poly2mask(rotatedCorners(:,1), rotatedCorners(:,2), h, w);
satelliteGrayUAV = satelliteGray;
satelliteGrayUAV(~mask) = 0; % Zero out pixels outside the UAV region
```

This transformation simulates the UAV camera's field of view, and the resulting masked image is subsequently used for feature matching.

3.1.2.4. Coordinate Transformation and Alignment

To align the extracted UAV region with the reference map, a back-rotation around the centroid of the UAV region is performed. This compensates for heading changes and ensures that image features are geometrically consistent across time steps. Pixel coordinates inside the UAV mask are transformed using an inverse rotation matrix as given in Eqn. 3.2.

$$R^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.2}$$

This rotation realigns the image patch to a standard orientation before cropping the minimum bounding rectangle. The transformation is applied as:

```
matlab
R_inv = [cosd(theta), sind(theta); -sind(theta), cosd(theta)];
rotatedCoords = (R_inv * pixelCoords')';
```

This ensures that the extracted sub-image is always aligned to a canonical reference frame, improving the reliability of subsequent feature detection and matching.

3.1.2.5. Summary

The preprocessing phase prepares the satellite image for downstream vision-based pose estimation. The algorithm ensures both computational efficiency and spatial consistency in feature matching by converting to grayscale, extracting context-aware UAV-centered

patches, and aligning them with the global reference map. This lays the foundation for accurate and robust visual localization in GPS-denied scenarios.

3.1.3. UAV Trajectory Simulation

To validate the proposed vision-based localization algorithm under various flight scenarios, multiple UAV trajectories are simulated within the satellite map. These include common flight patterns such as figure-eight, circular, and sinusoidal paths, each modeling realistic aerial survey and surveillance maneuvers. The simulation provides ground truth positions and headings against which the performance of the localization method can be quantitatively evaluated.

3.1.3.1. Trajectory Parametrization

All trajectories are defined over a common time interval $t \in [0,2\pi]$, discretized into $N = 50$ simulation steps:

```
matlab
t = linspace(0, 2*pi, numSteps); % Time parameter
numSteps = 50; % Number of simulation steps
```

The initial position of the UAV on the satellite image is defined as:

```
matlab
initialPosition = [400, 500];
```

This serves as a reference point for all subsequent path definitions.

3.1.3.2. Figure-Eight Path

A classic figure-eight trajectory is defined using sine and double-frequency sine functions:

```
matlab
x = 600 * sin(t); % Horizontal oscillation
y = 800 * sin(2*t); % Vertical oscillation with doubled frequency
relativeTranslations = [x' + initialPosition(1), y' + initialPosition(2)];
```

Mathematical expression of the trajectory is given in Eqn. 3.3.

$$x(t) = A_x \cdot \sin(t), \quad y(t) = A_y \cdot \sin(2t) \tag{3.3}$$

where $A_x = 600, A_y = 800$. This path creates intersecting loops, challenging the algorithm's ability to disambiguate visually similar regions.

3.1.3.3. Circular Path

To simulate consistent motion at a fixed altitude and turning rate, a circular trajectory is defined as:

```
matlab
radius = 500;
x = radius * cos(t);
y = radius * sin(t);
relativeTranslations = [x' + initialPosition(1), y' + initialPosition(2)];
```

As expressed in Eqn. 3.4.

$$x(t) = r \cdot \cos(t), \quad y(t) = r \cdot \sin(t) \tag{3.4}$$

where $r = 500$. This path serves as a baseline for evaluating steady-state behavior of the vision-based localization.

3.1.3.4. Sinusoidal Path

To emulate horizontal movement with vertical oscillation, a sinusoidal trajectory is defined:

```
matlab
amplitude = 200; wavelength = 800;
x = linspace(x_start, x_start + wavelength, numSteps);
y = amplitude * sin(2*t) + y_start;
```

Expressed as in Eqn. 3.5.

$$x(t) = x_0 + \frac{\lambda}{2\pi}t, \quad y(t) = A \cdot sin(2t) + y_0 \tag{3.5}$$

The amplitude $A = 200$ and wavelength $\lambda = 800$. This trajectory tests the localization algorithm's responsiveness to non-uniform motion.

3.1.3.5. Heading Angle Estimation

To simulate a realistic UAV motion model, heading angles are computed from the trajectory displacements:

```
matlab
for i = 2:numSteps
dx = relativeTranslations(i,1) - relativeTranslations(i-1,1);
dy = relativeTranslations(i,2) - relativeTranslations(i-1,2);
groundTruthHeading(i) = atan2d(dy, dx);
end
```

The heading θ_i is computed as given in Eqn. 3.6.

$$\theta_i = \tan^{-1}\left(\frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right) \quad (3.6)$$

This generates a time-varying heading profile for the UAV, which is critical for simulating camera orientation and enabling correct image extraction.

3.1.3.6. Summary

By simulating diverse trajectory types and dynamically computing positions and headings, this step provides a robust synthetic testbed for evaluating the vision-based virtual GPS system. These controlled motions establish the ground truth required for localization error analysis, discussed in subsequent sections.

3.1.4. UAV Image Transformation and Extraction

To simulate the UAV's visual perception at each timestep, a sub-image representing the UAV's field of view is dynamically extracted and geometrically transformed. This process enables realistic simulation of onboard camera captures with heading and altitude variations. Three primary operations are applied: rotation, back-rotation, and scaling.

3.1.4.1. Rotated Bounding Box Extraction

At each simulation step, a rectangular patch centered at the UAV's ground truth position is rotated according to its heading angle θ. A rotation matrix is defined as given in Eqn. 3.7.

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad (3.7)$$

The corners of the bounding box are rotated using this matrix and translated to the UAV's current position:

```
matlab
R = [cosd(theta), -sind(theta); sind(theta), cosd(theta)];
rotatedCorners = (R * corners')' + center;
```

This forms the boundaries of a rotated rectangular region of interest (ROI) to be masked from the satellite image.

3.1.4.2. Image Masking and Back-Rotation

A binary mask is created using 'poly2mask', isolating the rotated UAV region from the satellite image. The masked region is then inverse-rotated around its centroid to restore a canonical orientation as given in Eqn. 3.8.

$$R^{-1}(\theta) = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.8}$$

This transformation ensures consistency in feature matching across steps:

```
matlab
pixelCoords = [x_idx, y_idx] - centroid;
R_inv = [cosd(theta), sind(theta); -sind(theta), cosd(theta)];
rotatedCoords = (R_inv * pixelCoords')';
```

Valid pixels are relocated back to the original image coordinates to reconstruct the upright UAV image segment.

3.1.4.3. Scaling Transformation

To simulate changes in altitude, a scaling transformation is applied to the back-rotated image segment. A uniform affine transformation matrix is used as given in Eqn. 3.9.

$$H_{scale} = \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.9}$$

In MATLAB:

```
matlab
H1_true = [scale_xy, 0, 0; 0, scale_xy, 0; 0, 0, 1];
tform1 = affine2d(H1_true');
scaledUAV = imwarp(croppedUAV, tform1, 'OutputView', ...
    imref2d(size(croppedUAV)), 'FillValues', 0);
```

The scaling factor s is derived from a predefined profile to simulate zoom levels as the UAV changes altitude. This scaled patch is then used for feature extraction.

3.1.4.4. Summary

The UAV image transformation pipeline accurately replicates the UAV's perception of the terrain under dynamic flight conditions. By applying geometric transformations -rotation, back-rotation, and scaling- the system generates consistent, orientation-normalized sub-images. These serve as the foundation for robust feature matching and reliable pose estimation in the following stages.

3.1.5. Feature Extraction and Matching

A critical component of the proposed vision-based localization system is the robust detection and matching of key features between the UAV's local view and the global satellite reference image. This section outlines the mathematical foundation, algorithmic implementation, and MATLAB-based execution of the feature extraction and matching pipeline using Speeded-Up Robust Features (SURF) and Random Sample Consensus (RANSAC).

3.1.5.1. SURF Feature Detection

Speeded-Up Robust Features (SURF) is employed to extract keypoints and descriptors from both the UAV (local) and satellite (global) images. SURF is chosen for its scale and rotation invariance, and its computational efficiency compared to other detectors such as SIFT.

The SURF algorithm relies on the Hessian matrix to detect blob-like structures. For an image $I(x, y)$, the Hessian matrix H at scale σ is defined as given in Eqn. 3.10.

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix} \tag{3.10}$$

Here, L_{xx}, L_{yy}, and L_{xy} are the convolutions of the image with second-order Gaussian derivatives. Interest points are identified at locations where the determinant of the Hessian matrix is maximal.

In MATLAB, feature detection is performed as follows:

```
matlab
featureThreshold = 20;
pointsSat = detectSURFFeatures(satelliteGray, 'MetricThreshold', ...
    featureThreshold);
pointsUAV = detectSURFFeatures(scaledUAV, 'MetricThreshold', ...
    featureThreshold);
```

The parameter 'MetricThreshold' controls the sensitivity of the detector. A lower value yields more features but may include noise; a higher value ensures stronger, more reliable keypoints.

3.1.5.2. Feature Description and Matching

After detection, feature descriptors are computed. These descriptors are high-dimensional vectors that capture the local image gradient distribution around each keypoint. They are designed to be robust to changes in illumination, scale, and rotation.

```
matlab
[featuresSat, validPointsSat] = extractFeatures(satelliteGray, pointsSat);
[featuresUAV, validPointsUAV] = extractFeatures(scaledUAV, pointsUAV);
```

Feature matching is then performed using descriptor similarity, typically based on the Euclidean distance. The 'matchFeatures' function implements a brute-force nearest neighbor matcher, along with a ratio test to suppress ambiguous matches:

```
matlab
indexPairs = matchFeatures(featuresUAV, featuresSat, 'MaxRatio', 0.6, ...
    'Unique', true);
```

The ratio test (inspired by Lowe's method) compares the best and second-best matches for each feature. A match is retained only if the condition given in Eqn. 3.11 is ensured.

$$\frac{d_1}{d_2} < MaxRatio \tag{3.11}$$

where d_1 and d_2 are the distances to the closest and second-closest matches, respectively. This prevents false positives due to repetitive structures in satellite imagery.

3.1.5.3. RANSAC-Based Outlier Removal

Although SURF-based matching is robust, mismatches may still occur due to occlusion, distortion, or repetitive patterns. To refine the matches, RANSAC (Random Sample Consensus) is employed to eliminate outliers and estimate a geometric transformation.

```
matlab
[tformEstimated, inlierIdx] = estimateGeometricTransform2D( ...
    matchedPointsUAV, matchedPointsSat, 'similarity', 'MaxDistance', 3);
```

The "similarity" model estimates a transformation matrix that includes rotation, uniform scaling, and translation. The algorithm randomly selects minimal subsets of matches, computes the transform, and retains the model with the highest number of inliers based on the distance threshold as given in Eqn. 3.12.

$$\|T(x_i) - x_i'\| < MaxDistance, \quad \forall i \in inliers \tag{3.12}$$

Where $T(x_i)$ is the transformed point and x_i' is the actual matched point in the satellite image.

The filtered inliers are then extracted:

```
matlab
filteredMatchedPointsUAV = matchedPointsUAV(inlierIdx, :);
filteredMatchedPointsSat = matchedPointsSat(inlierIdx, :);
```

These matched point sets form the basis for accurate position and orientation estimation of the UAV.

3.1.5.4. Visualization and Debugging

For verification, matched features are visualized using:

```
matlab
showMatchedFeatures(satelliteGray, scaledUAV, ...
    filteredMatchedPointsSat, filteredMatchedPointsUAV, 'montage');
```

This visual inspection is critical to ensure that the matched features lie on consistent landmarks in both images.

3.1.5.5. Summary

The feature extraction and matching stage is fundamental to the Virtual GPS process. By leveraging SURF for invariant keypoint detection, robust matching heuristics, and RANSAC for geometric validation, the system ensures high-fidelity spatial alignment between UAV observations and the global map. The output of this module -the similarity transform and filtered correspondences- directly feeds into the pose estimation process described in the next section.

3.1.6. Pose Estimation and Correction

Once robust feature correspondences are established between the UAV and satellite images, the next step is to estimate the UAV's pose -its position and heading. This is achieved through geometric transformation estimation and bias correction procedures.

3.1.6.1. Estimating the Similarity Transformation

A similarity transformation model is used to capture the UAV image's scale, rotation, and translation relative to the satellite image. The transformation matrix $H \in \mathbb{R}^{3\times3}$ is estimated from the inlier feature correspondences via RANSAC:

```
matlab
H_est = tformEstimated.T;
H_est(3,:) = [0, 0, 1];
```

The general form of a 2D similarity transformation matrix is given in Eqn. 3.13.

$$H = \begin{bmatrix} a & b & t_x \\ -c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

where a, b, c, d encode rotation and uniform scaling, and t_x, t_y represent translation. The rotation angle θ and scale factor s are derived as given in Eqn. 3.14.

$$s = \frac{1}{\sqrt{a^2 + c^2}}, \quad \theta = tan^{-1}\left(\frac{c}{a}\right) \tag{3.14}$$

In MATLAB:

```
matlab
a = H_est(1,1);
c = H_est(2,1);
scale_exy = 1 / sqrt(a^2 + c^2);
estimatedHeadings(i) = atan2d(c, a);
```

3.1.6.2. Centroid-Based Bias Correction

The pose estimation may be biased due to misalignment between the detected feature centroid and the actual geometric center of the UAV image. This is corrected using a multi-step procedure:

1. Compute expected centroid in UAV image (center of image):

```
matlab
expectedUAVCentroid = [half_w, half_h];
```

2. Compute bias between expected and matched centroid:

```
matlab
matchedUAVCentroid = mean(filteredMatchedPointsUAV.Location, 1)';
biasUAV = matchedUAVCentroid - expectedUAVCentroid';
```

3. Undo scale and rotation effects to translate bias to satellite reference frame:

```
matlab
biasUAV_unscaled = biasUAV / scaleFactors(i);
R_inv = [cosd(-estimatedHeadings(i)), sind(-estimatedHeadings(i)); ...
    -sind(-estimatedHeadings(i)), cosd(-estimatedHeadings(i))];
biasUAV_corrected = R_inv * biasUAV_unscaled;
```

4. Adjust estimated satellite position using corrected bias:

```
matlab
rawSatCentroid = mean(filteredMatchedPointsSat.Location, 1)';
correctedCentroidSat = rawSatCentroid - biasUAV_corrected;
```

This corrected centroid (x, y) is recorded as the estimated UAV position.

3.1.6.3. Summary

By extracting rotation, scale, and translation from the similarity transform, and applying centroid-based corrections, the system yields a precise estimate of the UAV's position and orientation. This enables real-time geolocation solely through visual input.

3.1.7. Error Analysis and Performance Evaluation

To validate the effectiveness of the vision-based localization system, detailed error metrics are computed by comparing the estimated positions and headings with ground truth values from the simulation.

3.1.7.1. Absolute and Percentage Errors

The absolute position error in X and Y directions is defined as given in Eqn. 3.15.

$$E_x = |\hat{x} - x|, \quad E_y = |\hat{y} - y| \tag{3.15}$$

Percentage errors are calculated as given in Eqn. 3.16.

$$E_{x,\%} = \left(\frac{E_x}{|x|}\right) \times 100\%, \quad E_{y,\%} = \left(\frac{E_y}{|y|}\right) \times 100\% \tag{3.16}$$

In MATLAB:

```
matlab
positionErrorX = abs(estimatedPositions(:,1) - truePositions(:,1));
positionErrorY = abs(estimatedPositions(:,2) - truePositions(:,2));
percentageErrorPosX = (positionErrorX ./ abs(truePositions(:,1))) * 100;
percentageErrorPosY = (positionErrorY ./ abs(truePositions(:,2))) * 100;
```

Similarly, heading errors are computed using as given in Eqn. 3.17.

$$E_\theta = \left|\hat{\theta} - \theta\right|, \quad E_{\theta,\%} = \left(\frac{E_\theta}{|\theta|}\right) \times 100\% \tag{3.17}$$

3.1.7.2. Incremental Motion Analysis

To assess temporal consistency, incremental displacements are compared step-by-step:

```
matlab
trueIncremental = diff(truePositions);
estimatedIncremental = diff(estimatedPositions);
iabsoluteErrorX = abs(estimatedIncremental(:,1) - trueIncremental(:,1));
iabsoluteErrorY = abs(estimatedIncremental(:,2) - trueIncremental(:,2));
ipercentageErrorX = (iabsoluteErrorX ./ abs(trueIncremental(:,1))) * 100;
ipercentageErrorY = (iabsoluteErrorY ./ abs(trueIncremental(:,2))) * 100;
```

This helps identify drift or deviations in the estimated trajectory.

3.1.7.3. Visualization and Comparative Evaluation

Multiple plots are generated to visualize and compare performance:

- Trajectory comparison between ground truth and estimates
- Plot of absolute and percentage errors over time
- Heading estimation vs. ground truth
- Scale factor tracking and comparison

These outputs confirm the algorithm's ability to maintain reliable localization throughout the simulated flight.

3.1.8. Conclusion

This section has presented a comprehensive framework for visual localization of UAVs in GPS-denied environments using publicly available satellite imagery as a global reference. By combining trajectory simulation, geometric image transformation, feature-based matching, and robust pose estimation techniques, the proposed Virtual GPS system can estimate the UAV's global position and heading solely from visual input.

The use of SURF features ensures robustness to changes in scale, rotation, and partial occlusion, while the application of RANSAC-based filtering significantly reduces the influence of mismatched keypoints. The similarity transformation not only enables position estimation but also recovers orientation and scale changes, mimicking the effects of heading and altitude variations. Additionally, centroid-based correction strategies account for visual misalignments, improving the precision of the final pose estimate.

Although performance may vary depending on terrain texture, image resolution, and environmental conditions, the approach demonstrates a strong potential as a lightweight, vision-only alternative to GPS. The modularity of the pipeline also makes it suitable for extension to real-time onboard systems.

In the next section, this vision-based localization system will be further enhanced through integration with inertial navigation data using Kalman filtering. The resulting sensor fusion system is expected to yield a more resilient, drift-corrected navigation framework that balances the complementary strengths of vision and inertial sensing.

3.2. Dynamic System Modelling of UAV

3.2.1. Overview

The visual-inertial navigation system developed in this research is designed to be platform-independent, which means that its principles can be applied to a wide area of Unmanned Aerial Vehicle (UAV) types. The core requirement for the system's applicability is that the vehicle operates as a rigid body and allows for the integration of an inertial measurement unit (IMU) and a downward-facing camera (which can be improved as any moving camera with further studies). The methodology is particularly well-suited for multirotor UAVs, which have become standard platforms for numerous applications due to their vertical take-off and landing (VTOL) capabilities and high maneuverability (Hassanalian & Abdelkefi, 2017).

Common multirotor configurations such as quadcopters, hexacopters, and octocopters are all compatible with the proposed navigation framework. Quadcopters offer simplicity and efficiency, making them suitable for many research and commercial tasks. Hexacopters and octocopters, with their increased number of rotors, provide greater payload capacity and a higher degree of fault tolerance, as they can often maintain stable flight even after one or more motor failures (Osmic et al., 2020). This inherent redundancy makes them preferable for critical missions where higher reliability is required.

While the algorithms presented in this thesis are broadly applicable, the dynamic model used for simulation and the design of the Nonlinear Model Predictive Controller (NMPC) are specifically parameterized for a heavy-lift octocopter. This choice was made to ensure that the system is evaluated against a challenging, high-inertia platform representative of those used in demanding industrial, logistical, and reconnaissance applications, where operating in GPS-denied environments is a frequent and critical challenge.

3.2.2. 6-DOF Dynamic Model of a Multirotor UAV

The motion of a multirotor UAV, as a rigid body moving in three-dimensional space, is described by a six-degrees-of-freedom (6-DOF) model, which includes three translational and three rotational components. To analyze this motion, two essential coordinate frames are utilized: the non-accelerating, Earth-fixed inertial frame -denoted as I- and the body frame -denoted as B-, which is fixed to the UAV's center of gravity and rotates with it

(Hassanalian & Abdelkefi, 2017). The relationship between these frames, along with the primary forces and moments acting on the UAV, is illustrated in Figure 3.1.

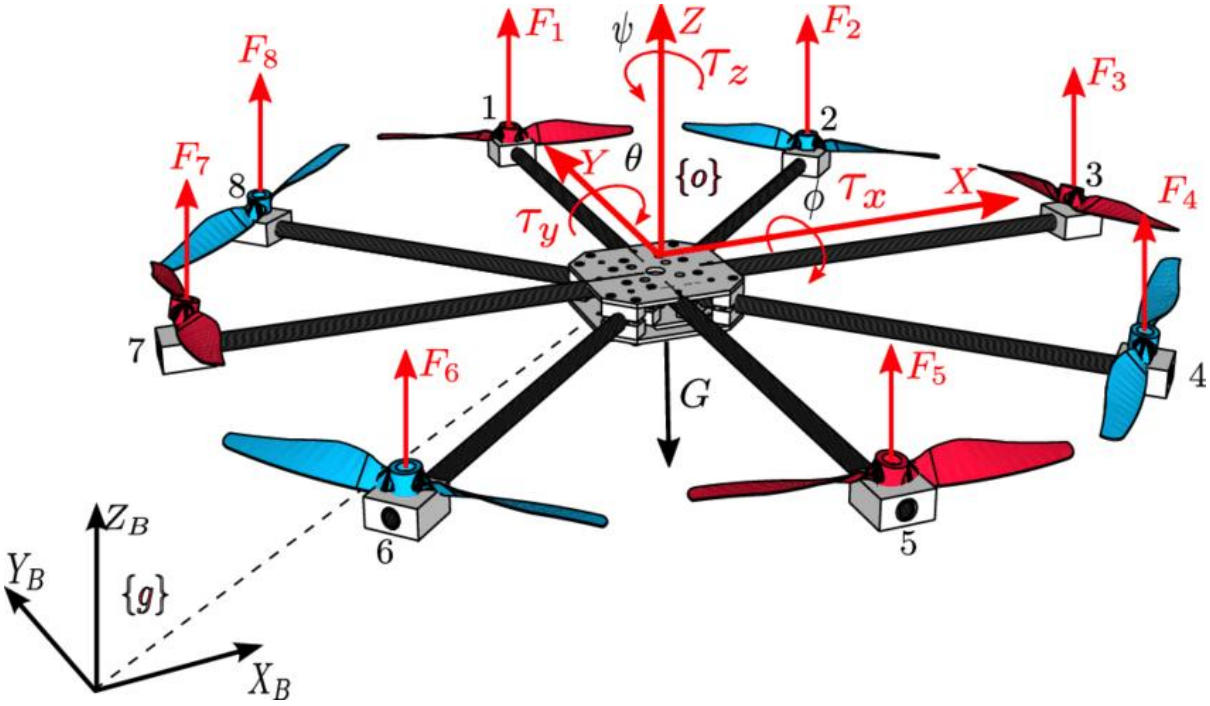

Figure 3.1. Body and ground fixed frames for an octocopter system

The complete state of the UAV is typically described by a 12-dimensional vector, encompassing its position and orientation in the inertial frame, and its linear and angular velocities in the body frame. The equations of motion are derived from Newton-Euler formulations, which provide a robust framework for describing the coupled dynamics of a rigid body (Luukkonen, 2011).

3.2.2.1. Translational Dynamics

The translational motion describes the acceleration of the UAV's center of gravity with respect to the inertial frame. According to Newton's second law, the net force acting on the UAV equals its mass times its inertial acceleration. The primary forces are the total thrust generated by the rotors, which acts along the body frame's Z-axis, and gravity, which acts along the inertial frame's Z-axis. To combine these forces, the thrust must be rotated from the body frame to the inertial frame. This leads to the translational dynamic equation as in Eqn. 3.18.

$$m\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = R(\phi,\theta,\psi)\begin{bmatrix}0\\ 0\\ -T\end{bmatrix} + \begin{bmatrix}0\\ 0\\ mg\end{bmatrix} \tag{3.18}$$

where:

m is the total mass of the UAV,

(x, y, z) are the position coordinates in the inertial frame, and $(\ddot{x}, \ddot{y}, \ddot{z})$ are their second time derivatives (linear accelerations).

g is the acceleration due to gravity,

T is the total thrust force produced by all rotors, directed opposite to the body frame's z-axis (Z_B),

$R(\phi, \theta, \psi)$ is the rotation matrix that transforms vectors from the body frame to the inertial frame, defined by the Euler angles: roll (ϕ), pitch (θ), and yaw (ψ). The ZYX convention for this rotation matrix is commonly used (Osmic et al., 2020) as given between Eqn. 3.19 and 3.22.

$$R(1) = \begin{bmatrix}\cos(\psi)\cos(\theta)\\ \sin(\psi)\cos(\theta)\\ -\sin(\theta)\end{bmatrix} \tag{3.19}$$

$$R(2) = \begin{bmatrix}\cos(\psi)\sin(\theta)\sin(\phi) - \sin(\psi)\cos(\phi)\\ \sin(\psi)\sin(\theta)\sin(\phi) + \cos(\psi)\cos(\phi)\\ \cos(\theta)\sin(\phi)\end{bmatrix} \tag{3.20}$$

$$R(3) = \begin{bmatrix}\cos(\psi)\sin(\theta)\cos(\phi) + \sin(\psi)\sin(\phi)\\ \sin(\psi)\sin(\theta)\cos(\phi) - \cos(\psi)\sin(\phi)\\ \cos(\theta)\cos(\phi)\end{bmatrix} \tag{3.21}$$

where:

$$R(\phi,\theta,\psi) = [\, R(1)\ \ R(2)\ \ R(3)\,] \tag{3.22}$$

3.2.2.2. Rotational Dynamics

The rotational motion of the UAV is described in the body frame by Euler's equations for a rigid body. These equations relate the applied external moments (torques) to the rate of change of the UAV's angular velocity as described in Eqn. 3.23.

$$I\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\tau_x\\ \tau_y\\ \tau_z\end{bmatrix} - \begin{bmatrix}p\\ q\\ r\end{bmatrix} \times \left(I\begin{bmatrix}p\\ q\\ r\end{bmatrix}\right) \tag{3.23}$$

where:

(p, q, r) are the angular velocities (roll rate, pitch rate, and yaw rate) in the body frame, and $(\dot{p}, \dot{q}, \dot{r})$ are their time derivatives (angular accelerations).

(τ_x, τ_y, τ_z) are the external moments (torques) about the body frame axes, generated by the differential thrust of the rotors.

I is the inertia tensor of the UAV, which is a 3x3 matrix. Assuming the body axes are aligned with the principal axes of inertia, this tensor simplifies to a diagonal matrix as given in Eqn. 3.24.

$$I = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{3.24}$$

The term $[p; q; r] \times (I\ [p; q; r])$ represents the gyroscopic effects that arise from the coupling of rotations in a non-inertial frame. These two sets of equations -translational and rotational- provide a comprehensive model for the real-world motion of a multirotor UAV and form the basis from which simplified models can be derived for specific control and estimation tasks (Luukkonen, 2011).

3.2.3. Dynamic Model of a 2D UAV at Constant Altitude

In this work, for the prediction stages of the estimation and control algorithms, such as the Extended Kalman Filter (EKF), a 6-state dynamic model of a 2D UAV operating at a constant altitude is defined. This model captures the UAV's translational and rotational motion in a 2D plane, incorporating forces and moments applied in the body frame, while its position and orientation evolve in the inertial (global) frame.

The state vector is defined as given in Eqn. 3.25.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ u \\ v \\ \omega \end{bmatrix} \tag{3.25}$$

Where:

- x, y: Position in the inertial frame
- θ: Heading angle (yaw) in the inertial frame
- u: Longitudinal velocity in the body frame (along UAV forward axis)

- v: Lateral velocity in the body frame (side-slip)
- ω: Yaw rate (rotation rate around vertical axis, body frame)

The control input vector consists of parameters given in Eqn. 3.26.

$$\mathbf{F} = \begin{bmatrix} F_x^b \\ F_y^b \end{bmatrix}, \qquad M_z \tag{3.26}$$

Where:

- F_x^b: Body-frame force in the forward direction
- F_y^b: Body-frame force in the lateral direction
- M_z: Yaw moment (torque around z-axis)

The translational acceleration components are derived using Newton's second law in the body frame, accounting for Coriolis acceleration due to yaw rotation as given in Eqn. 3.27 and 3.28.

$$\dot{u} = \frac{F_x^b}{m} + \omega v \tag{3.27}$$

$$\dot{v} = \frac{F_y^b}{m} - \omega u \tag{3.28}$$

Here, m represents the mass of the UAV, while ωv (adds to u) and $-\omega v$ (subtracts from v) represent pseudo forces due to rotation. These equations represent body-frame accelerations affected by both thrust and rotational dynamics. Yaw motion is modeled via angular acceleration derived from Euler's equation of rotation and given in Eqn. 3.29.

$$\dot{\omega} = \frac{M_z}{I_z} \tag{3.29}$$

Where M_z is the control moment applied in yaw I_z is the moment of inertia about the vertical axis (z-axis). This equation governs how the UAV's heading rate evolves under applied torque. The position and heading evolution in the inertial frame are obtained by transforming the body-frame velocities to the inertial frame using a rotation by the current heading angle θ as given between Eqn. 3.30 and 3.32.

$$\dot{x} = u\cos\theta - v\sin\theta \tag{3.30}$$

$$\dot{y} = u\sin\theta + v\cos\theta \tag{3.31}$$

$$\dot{\theta} = \omega \tag{3.32}$$

This transformation rotates the local body-frame velocity vector $[u\ v]^T$ into the inertial frame, effectively translating the UAV in space. Combining all components, the full state derivative is given in Eqn. 3.33.

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{u} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} u\cos\theta - v\sin\theta \\ u\sin\theta + v\cos\theta \\ \omega \\ \frac{F_x^b}{m} + \omega v \\ \frac{F_y^b}{m} - \omega u \\ \frac{M_z}{I_z} \end{bmatrix} \tag{3.33}$$

It is crucial to note the distinction: the position variables x, y and the heading angle θ are defined in the inertial (world) frame, while the velocity components u, v and the angular velocity ω are defined in the body frame of the UAV. The control inputs F_x^b, F_y^b and the moment M_z are also expressed in the body frame. This distinction ensures proper transformations are applied when converting between measurement, dynamics, and control formulations.

This model provides a physically consistent, minimal state representation suitable for discrete-time integration (e.g., using Runge-Kutta or Euler), and for use in estimation filters such as EKF. In practice, it forms the prediction model $f(x, u)$ within the Kalman filtering framework.

3.2.4. Designed Model Parameters

The main characteristics of the UAV model which is used in this work are summarized in Table 3.1. The total mass of the simulated UAV is defined as 37.2 kg. This value is consistent with the operational weight of heavy-lift platforms designed for significant payload capacity. For the 2D planar simulation, the moment of inertia about the yaw axis (Izz) is the critical rotational parameter. A value of 8.15 kg·m² is used, which is a realistic estimate for a large-diameter octocopter frame, informed by experimental studies on similar multirotor vehicles.

The propulsion system is modeled based on eight high-power electric motors, collectively providing a total maximum lift thrust of approximately 7690 N. For the 2D planar simulation, the parameter F_max does not represent a constrained horizontal force but rather the theoretical maximum thrust available from the propulsion system. The value used as the control input limit in the NMPC controller, F_max = 7683.2 N, effectively serves as an unconstrained ceiling, allowing the optimizer to command any required force within the vehicle's physical capabilities. The maximum yaw moment (M_max) is defined as 120 N·m, representing the vehicle's capacity for rotational acceleration generated by the differential thrust across the octocopter's frame.

Table 3.1. Parameters of the designed UAV Model for simulation

Parameter	Symbol	Value	Unit
Mass	m	37.2	kg
Yaw Moment of Inertia	Izz	8.15	kg·m²
Max. Control Force Limit	F_max	7683.2	N
Max. Yaw Moment	M_max	120	N·m
Gyroscope Bias Drift (Z-axis)	-	±0.05	dps
Gyroscope White Noise (Z-axis)	-	0.015	dps RMS
Accelerometer Bias Drift (X,Y-axes)	-	±0.0049	m/s²
Accelerometer White Noise (X,Y-axes)	-	0.00225	m/s² RMS

To enhance the simulation’s fidelity, the inertial measurement model incorporates random errors representative of a modern, high-performance MEMS IMU. The gyroscope model for the yaw axis includes a bias drift of ±0.05 °/s and white noise of 0.015 °/s RMS. Similarly, the accelerometer model for the horizontal body axes includes a bias drift of ±0.0049 m/s² and white noise of 0.00225 m/s² RMS. These error parameters are fundamental to the EKF's process noise formulation and provide a realistic basis for evaluating the estimator's performance in the presence of sensor imperfections. On the other hand, in the scope of this work, accelerometer bias and noise parameters are corrupted to obtain better visual observability across the results on satellite images as mentioned in Section 3.5.

All these parameters in conjunction define a high-fidelity dynamic model that serves as the basis for the simulation tests so that the performance of the proposed visual-inertial navigation and control system is validated against an aggressive and realistic UAV platform.

3.3. Nonlinear MPC and EKF-Based Visual-Inertial Control Loop for UAV Navigation

3.3.1. Overview of Visual-Inertial Sensor Fusion

In environments where Global Positioning System (GPS) signals are unavailable or unreliable, Unmanned Aerial Vehicles (UAVs) must rely on their onboard sensors for accurate localization. This methodology proposes a robust solution by integrating a vision-based virtual GPS system with an Inertial Navigation System (INS) using a Kalman Filter framework. The visual system provides drift-free but potentially sparse and noisy global pose estimates, while the INS delivers high-frequency motion information that is susceptible to drift. The primary objective of this sensor fusion is to leverage the complementary characteristics of both sensors, thereby achieving a more accurate and reliable state estimate.

The mathematical formulation and theoretical framework of the Kalman Filter used in this work are designed to fuse visual localization outputs (position and heading) with inertial measurements (accelerometer and gyroscope). The model is initially structured in continuous time to align with the physical characteristics of the sensors and is subsequently discretized for implementation in digital systems, such as MATLAB.

3.3.2. Kinematic Extended Kalman Filter (EKF) Design for Planar UAV Localization Using IMU Measurements

3.3.2.1. Introduction

In GPS-denied environments, the accurate localization of UAVs (Unmanned Aerial Vehicles) is critical and often relies on dead-reckoning methods that combine inertial measurements and onboard estimation algorithms. In this context, a 2D kinematic Extended Kalman Filter (EKF) is developed to estimate the planar position, heading, and body-frame velocities of a UAV using only inertial measurements: body-frame linear accelerations (a_x, a_y) and yaw angular acceleration (ω).

3.3.2.2. State Definition

A 6-dimensional state vector is defined that captures both inertial position and body-frame dynamics in Eqn. 3.34.

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \\ u \\ v \\ \omega \end{bmatrix} \tag{3.34}$$

where:

x, y — Position in inertial (world) coordinates $[rad]$

θ — Heading angle $[rad]$

u — Body-frame velocity along the x-axis $[\frac{m}{s}]$

v — Body-frame velocity along the y-axis $[\frac{m}{s}]$

ω — Yaw rate $[\frac{rad}{s}]$

This formulation decouples velocity direction from the heading, enabling the model to capture drift and slip effects.

3.3.2.3. Process Model (Prediction Step)

The continuous-time nonlinear dynamics are derived from kinematics and body-frame accelerations as given in Eqn. 3.35.

$$\begin{aligned} \dot{x} &= u\cos\theta - v\sin\theta \\ \dot{y} &= u\sin\theta + v\cos\theta \\ \dot{\theta} &= \omega \\ \dot{u} &= a_x \\ \dot{v} &= a_y \\ \dot{\omega} &= \alpha \end{aligned} \tag{3.35}$$

Discretized with time step Δt, the predicted state is given in Eqn. 3.36.

$$\mathbf{x}_{k+1}^{-} = \mathbf{x}_k + \Delta t \cdot \begin{bmatrix} u\cos\theta - v\sin\theta \\ u\sin\theta + v\cos\theta \\ \omega \\ a_x \\ a_y \\ \alpha \end{bmatrix} \tag{3.36}$$

3.3.2.4. Measurement Model

Assume the IMU provides:

a_x, a_y: body-frame linear acceleration along x and y, respectively

$\alpha = \dot{\omega}$: angular acceleration

These are treated as inputs, not direct measurements of the state, and are used only in the prediction step. No measurement update is performed unless absolute sensors (e.g., GPS or visual localization) are available.

3.3.2.5. Jacobian of Process Model

The Jacobian matrix $\mathbf{F} = \frac{\partial f}{\partial \mathbf{x}}$ is given in Eqn. 3.37.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & -\Delta t(u\sin\theta + v\cos\theta) & \Delta t\cos\theta & -\Delta t\sin\theta & 0 \\ 0 & 1 & \Delta t(u\cos\theta - v\sin\theta) & \Delta t\sin\theta & \Delta t\cos\theta & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.37}$$

This matrix is used for state covariance propagation.

3.3.2.6. Process Noise Covariance [Q]

The process noise is considered for uncertainty in the dynamics, especially due to input noise (accelerometer and gyroscope noise). A velocity-dependent noise model is employed as in Eqn. 3.38.

$$\mathbf{Q} = \text{diag}\left(\sigma_x^2,\ \sigma_y^2,\ \sigma_\theta^2,\ \sigma_u^2,\ \sigma_v^2,\ \sigma_\omega^2\right) \tag{3.38}$$

However, some elements are scaled based on motion intensity as given in Eqn. 3.39 and 3.40.

$$\sigma_x^2, \sigma_y^2 \propto 1 + \| [u\ v] \|^2 \tag{3.39}$$

$$\sigma_\theta^2 \propto 1 + \omega^2 \tag{3.40}$$

To rotate positional uncertainty into the inertial frame, the upper-left 2x2 block of $\mathbf{Q}$ is computed as given in Eqn. 3.41.

$$\mathbf{Q}_{xy} = \mathbf{R}(\theta) \cdot \mathbf{Q}_{body} \cdot \mathbf{R}(\theta)^T \tag{3.41}$$

where $\mathbf{Q}_{body} = \text{diag}(\sigma_x^2, \sigma_y^2)$, and $\mathbf{R}(\theta)$ is the body-to-inertial rotation matrix.

3.3.2.7. Covariance Update (Prediction)

Covariance update is calculated as in Eqn. 3.42.

$$\mathbf{P}_{k+1}^{-} = \mathbf{F}_k \, \mathbf{P}_k \, \mathbf{F}_k^{\top} + \mathbf{Q}_k \tag{3.42}$$

If no absolute measurement is available, the state update is as given in Eqn. 3.43.

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1}^{-}, \quad \mathbf{P}_{k+1} = \mathbf{P}_{k+1}^{-} \tag{3.43}$$

3.3.2.8. Optional Measurement Update

If occasional absolute pose information becomes available (e.g., satellite-based localization or GPS), a measurement update can be performed as described between Eqn. 3.44 and 3.47.

$$\mathbf{z} = [x, y, \theta]^{\top} \tag{3.44}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.45}$$

$$\mathbf{K} = \mathbf{P}^{-}\mathbf{H}^{\top}(\mathbf{H}\mathbf{P}^{-}\mathbf{H}^{\top} + \mathbf{R})^{-1} \tag{3.46}$$

$$\mathbf{x}^{+} = \mathbf{x}^{-} + \mathbf{K}(\mathbf{z} - \mathbf{H}\mathbf{x}^{-}), \quad \mathbf{P}^{+} = (I - \mathbf{K}\mathbf{H})\mathbf{P}^{-} \tag{3.47}$$

3.3.2.9. Summary

This kinematic EKF formulation:

- Incorporates IMU-based inertial integration
- Models drift via separate u, v states
- Dynamically adapts process noise based on motion intensity
- Allows optional absolute correction when external sensors are available

It provides a robust and lightweight solution for planar UAV localization in GPS-denied environments using only inertial sensors.

3.3.3. Theoretical Framework of Nonlinear Model Predictive Control (NMPC) for UAVs

3.3.3.1. Introduction to NMPC

Model Predictive Control (MPC) is a class of optimal control strategies where a system optimizes future control actions over a finite prediction horizon using a known model of the system. When the system dynamics or constraints are nonlinear, the approach is

referred to as Nonlinear MPC (NMPC). In NMPC, at each control step, the following optimization problem that is given in Eqn. 3.48 is solved:

$$\min_{\mathbf{u}_{0:N-1}} J(\mathbf{x}_0, \mathbf{u}_{0:N-1}) \tag{3.48}$$

subject to:

- The nonlinear system dynamics,
- Initial condition x_0,
- Constraints on inputs and (optionally) states.

Only the first control input $\mathbf{u}_0$ is applied to the system; then, the optimization is repeated at the next time step in a receding horizon fashion.

3.3.3.2. State and Control Definitions

For a 3-DOF planar UAV with fixed altitude, the state vector is given in Eqn. 3.49.

$$x = \begin{bmatrix} x \\ y \\ \theta \\ u \\ v \\ \omega \end{bmatrix} \in \mathbb{R}^6 \tag{3.49}$$

Where:

- x, y: UAV position in the inertial frame,
- θ: yaw angle (heading),
- u, v: body-frame translational velocities,
- ω: yaw rate.

The control input vector is given by Eqn. 3.50.

$$u = \begin{bmatrix} F_x \\ F_y \\ M_z \end{bmatrix} \in \mathbb{R}^3 \tag{3.50}$$

Where:

• F_x, F_y: body-frame forces,

• M_z: moment about the vertical axis.

3.3.3.3. System Dynamics

The continuous-time nonlinear system dynamics can be written as given in Eqn. 3.51.

$$\dot{\mathbf{x}} = f(\mathbf{x},\mathbf{u}) \tag{3.51}$$

Specifically, for a 3-DOF UAV parameters are described in between Eqn. 3.52 and 3.57.

$$\dot{x} = u\cos\theta - v\sin\theta \tag{3.52}$$

$$\dot{y} = u\cos\theta - v\sin\theta \tag{3.53}$$

$$\dot{\theta} = \omega \tag{3.54}$$

$$\dot{u} = \frac{F_x}{m} + \omega v \tag{3.55}$$

$$\dot{v} = \frac{F_y}{m} - \omega u \tag{3.56}$$

$$\dot{\omega} = \frac{M_z}{I_z} \tag{3.57}$$

where m is the mass and I_z is the moment of inertia about the yaw axis. These equations are discretized using numerical integration as given in Eqn. 3.58, typically Runge-Kutta 4th-order (RK4), to form the discrete-time prediction model used in NMPC.

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_\mathrm{k}, \mathbf{u}_k, \Delta t) \tag{3.58}$$

3.3.3.4. Cost Function

The NMPC controller minimizes a quadratic cost function over a horizon of length N, typically in the form of Eqn. 3.59.

$$J = \sum_{k=0}^{N-1} \left[\left(\mathbf{x}_k - \mathbf{x}_{ref}\right)^T Q \left(\mathbf{x}_k - \mathbf{x}_{ref}\right) + \mathbf{u}_k^T R \mathbf{u}_k \right] \tag{3.59}$$

Where:

• $\mathbf{x}_{ref} \in \mathbb{R}^6$: reference state (typically position and heading),

• $Q \in \mathbb{R}^{6x6}$: positive semi-definite matrix penalizing state errors,

• $R \in \mathbb{R}^{3x3}$: positive definite matrix penalizing control effort.

In simplified or regulation-style NMPC, only part of the state vector may be penalized, e.g. Eqn. 3.60.

$$Q = diag\left(q_x, q_y, q_\theta, 0,0,0\right) \tag{3.60}$$

so that velocity components u, v, ω are not penalized directly.

3.3.3.5. Constraints

NMPC explicitly handles constraints on the inputs and optionally on the states given in Eqn. 3.61.

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max}, \quad \forall k \in [0, N-1] \tag{3.61}$$

In the UAV application, the limitation given in between Eqn. 3.62 and 3.64 ensures that the control actions are physically realizable by the UAV actuators.

$$F_x \in [-F_{max}, F_{max}] \tag{3.62}$$

$$F_y \in [-F_{max}, F_{max}] \tag{3.63}$$

$$M_z \in [-M_{max}, M_{max}] \tag{3.64}$$

3.3.3.6. Trajectory Tracking vs. Regulation

NMPC can be formulated either as:

1. Tracking control: where the reference $\mathbf{x}_{ref}(k)$ is a full time-varying trajectory,
2. Regulation: where the reference is a fixed point $\mathbf{x}_{ref}$.

The proposed system employs a regulation-style cost function where the first point of the trajectory segment is used as the constant target over the horizon. This approach simplifies computation and avoids issues with rapidly changing references in noisy environments such as image-based localization.

3.3.3.7. Adaptive Weighting (Online Q Adaptation)

The implemented NMPC includes adaptive Q weighting:

1. If position or heading error is small: smaller Q → smoother control
2. If errors are large: larger Q → aggressive convergence

This is achieved by scaling each diagonal entry of Q based on a normalized error term e_i/e_{thresh}, then clamping and interpolating as given in Eqn. 3.65.

$$Q_i = (1 - \alpha_i)Q_{low,i} + \alpha_i Q_{high,i} \tag{3.65}$$

Where α_i is described in Eqn. 3.66.

$$\alpha_i = \min\left(\frac{|e_i|}{e_{thresh,i}}, 1\right) \tag{3.66}$$

$Q_{low,i}, Q_{high,i}$ are the minimum and maximum weights for state component i. This ensures robustness and adaptive aggressiveness of the controller.

3.3.3.8. Online Optimization Problem

At each time step, NMPC solves a nonlinear optimization problem given in Eqn. 3.67,

$$\min_{\mathbf{u}_{0:N-1}} J(\mathbf{x}_0, \mathbf{u}_{0:N-1}) \tag{3.67}$$

subject to Eqn. 3.68 and 3.69.

$$\mathbf{x}_{k+1} = \Phi(\mathbf{x}_k, \mathbf{u}_k, \Delta t) \tag{3.68}$$

$$\mathbf{u}_{min} \leq \mathbf{u}_k \leq \mathbf{u}_{max} \tag{3.69}$$

This is solved using a gradient-based nonlinear optimizer, such as Sequential Quadratic Programming (SQP) via 'fmincon'. Only the first control input $\mathbf{u}_0$ is applied to the real system. At the next timestep, the optimization is repeated with updated state information obtained via EKF.

3.3.4. UAV Simulation: Main Loop Flowchart (Algorithm Description)

The UAV simulation main loop implements a step-by-step algorithm description. At each time step i, the following operations are executed:

1. Retrieve Reference Trajectory Segment

   A segment of the reference trajectory spanning the next N steps is extracted as in Eqn. 3.70.

   $$traj_{segment} = \{\mathbf{x}_{ref}(i), \mathbf{x}_{ref}(i+1), \dots, \mathbf{x}_{ref}(i+N-1)\} \tag{3.70}$$

   This segment is used as the prediction target in the NMPC optimization.

2. Solve the NMPC Optimization Problem

   Using the current estimated state $\mathbf{x}_i$, the following finite-horizon optimal control problem is solved as given in Eqn. 3.71.

   $$\mathbf{u}^*_{0:N-1} = arg\min J(\mathbf{x}_i, \mathbf{u}_{0:N-1}) \tag{3.71}$$

   Subject to nonlinear UAV dynamics and input constraints. Only the first control input $\mathbf{u}_0$ is applied to the system.

3. Simulate the True UAV Dynamics (Prediction)

   The predicted (true) state is propagated forward using a Runge-Kutta 4th-order integrator (RK4) as described in Eqn. 3.72.

   $$\mathbf{x}^{i+1}_{pred} = \Phi\left(\mathbf{x}^{i+1}_{pred}, \mathbf{u}_0, \Delta t\right) \tag{3.72}$$

   This simulates the physical evolution of the UAV under the selected control input.

4. Simulate Visual Pose Measurement (Image-Based GPS)

A simulated pose estimate $(\hat{x}, \hat{y}, \hat{\theta})$ is generated using an image-based localization algorithm such as feature matching with a satellite map.

5. Simulate IMU Measurements

Synthetic IMU data including accelerometer and gyroscope readings are generated, incorporating sensor bias and noise. These measurements are used for the state estimation in the subsequent step.

6. Validate Absolute Measurement (Every nth Steps)

Every nth steps, if visual measurements are available, the Mahalanobis distance is computed as in Eqn. 3.73.

$$d_M = \left(\mathbf{z}_{meas} - \mathbf{x}_{pred}\right)^T P^{-1} \left(\mathbf{z}_{meas} - \mathbf{x}_{pred}\right) \tag{3.73}$$

If d_M exceeds the predefined threshold, the visual measurement is rejected. Otherwise, it is accepted and used in the EKF update.

7. Run Extended Kalman Filter (EKF) for State Estimation

An EKF update is performed using:

IMU data (always available)

Visual data (if accepted in Step 6)

The updated state estimate is given in Eqn. 3.74.

$$\hat{\mathbf{x}}^i = EKF\left(\hat{\mathbf{x}}^i, \mathbf{z}_{imu}, \mathbf{z}_{vis}\right) \tag{3.74}$$

8. Log Simulation Data

Simulation data is logged for subsequent analysis and evaluation.

9. Adapt NMPC Cost Weights (Q Matrix)

The tracking error is computed as given in Eqn. 3.75.

$$\mathbf{e} = \hat{\mathbf{x}}^i - \mathbf{x}_{ref}^i \tag{3.75}$$

The diagonal entries of the cost matrix Q are adjusted based on the magnitude of position and heading errors. This makes the controller more aggressive when errors are large and smoother when tracking is accurate.

10. Visualize (Optional)

Real-time plots of the UAV trajectory, control inputs, heading angles, and reference tracking performance are optionally generated.

11. Check Termination Condition

    If $(i < numSteps - N)$, the algorithm proceeds to the next time step. Otherwise, the simulation is terminated.

12. End of Loop

3.4. Nonlinear Model Predictive Controller (NMPC) Formulations for Different Methods

3.4.1. Overview

In this work, 4 different approaches are used to obtain cost for MPC. These are linear path, exponential position, regulating and switching approaches which are described as below.

3.4.2. Regulating NMPC for a 3-DOF Planar UAV

3.4.2.1. Model and Discretization

The UAV state and input are given in Eqn. 3.76.

$$x = [x,\ y,\ \theta,\ u,\ v,\ \omega]^\top \in \mathbb{R}^6, \qquad u = [F_x,\ F_y,\ M_z]^\top \in \mathbb{R}^3 \tag{3.76}$$

Continuous-time dynamics (planar, fixed altitude) are given in Eqn. 3.77.

$$\begin{aligned} &\dot{x} = u\cos\theta - v\sin\theta, \dot{y} = u\sin\theta + v\cos\theta, \dot{\theta} = \omega \\ &\dot{u} = F_x/m + \omega v, \quad \dot{v} = F_y/m - \omega u, \quad \dot{\omega} = M_z/I_z \end{aligned} \tag{3.77}$$

The discrete prediction model given in 3.78 is obtained via Runge–Kutta 4 integration with sampling time Δt:

$$x_{k+1} = F(x_k, u_k; \Delta t) \tag{3.78}$$

Heading angles are maintained continuously using wrap-to-π unwrapping.

3.4.2.2. Decision Variables via Move-Blocking

Instead of optimizing every u_k for $k = 0, \dots, N-1$, the controller optimizes *anchor controls* as given in Eqn. 3.79,

$$U^{(j)} = [F_x^{(j)},\ F_y^{(j)},\ M_z^{(j)}]^\top, \quad j = 1, \dots, n_c \tag{3.79}$$

stacked as in Eqn. 3.80.

$$U_{\text{anc}} = [(U^{(1)})^\top \quad \dots \quad (U^{(n_c)})^\top]^\top \in \mathbb{R}^{3n_c} \tag{3.80}$$

The horizon inputs are obtained by linear interpolation as described in Eqn. 3.81.

$$u_k = \text{interp}\big(\{(t_j^{\text{anc}}, U^{(j)})\}, t_k\big), \quad t_k = k\Delta t \tag{3.81}$$

This reduces dimension, smooths the input sequence, and ensures that actuator limits remain satisfied under convex interpolation.

3.4.2.3. Reference for Regulation

In the formulation given by Eqn. 3.82, the reference is a *fixed terminal pose*

$$r_\star = \begin{bmatrix} x_\star \\ y_\star \\ \theta_\star \end{bmatrix} \tag{3.82}$$

taken from the trajectory input (with $\theta_\star$ converted to radians). The same $r_\star$ is applied for all steps $k = 1, \dots, N$.

To maintain heading continuity, an unwrapped heading ψ_k is updated at each step as given in Eqn. 3.83.

$$\psi_k = \psi_{k-1} + \text{wrapToPi}(\theta_k - \psi_{k-1}), \quad \psi_1 = \theta_1 \tag{3.83}$$

so that heading errors remain continuous.

3.4.2.4. Prediction and Error Variables

Predicted states $\{x_k\}$ are generated using the discrete model with the interpolated inputs $\{u_k\}$.

Errors relative to the constant target are defined as given in Eqn. 3.84.

$$e_{p,k} = \begin{bmatrix} x_k - x_\star \\ y_k - y_\star \end{bmatrix}, \qquad e_{\theta,k} = \psi_k - \theta_\star \tag{3.84}$$

3.4.2.5. Stage Cost and Objective

The quadratic weights are given in Eqn. 3.85.

$$Q = \text{diag}(q_x, q_y, q_\theta) \succcurlyeq 0, \quad R = \text{diag}(r_x, r_y, r_\theta) \succcurlyeq 0, \quad Q_{xy} = \text{diag}(q_x, q_y) \tag{3.85}$$

The stage cost is given in 3.86.

$$\ell_k = e_{p,k}^\top Q_{xy} e_{p,k} + q_\theta \, e_{\theta,k}^2 + u_k^\top R u_k \tag{3.86}$$

The total horizon cost is as described in Eqn. 3.87.

$$J(U_{\text{anc}}) = \sum_{k=1}^{N} \ell_k \tag{3.87}$$

3.4.2.6. Constraints

Anchor inputs are constrained by actuator limits as given in Eqn. 3.88.

$$-F_{\max} \le F_x^{(j)}, F_y^{(j)} \le F_{\max}, \qquad -M_{\max} \le M_z^{(j)} \le M_{\max} \tag{3.88}$$

Because interpolation is convex, all interpolated u_k satisfy the same bounds.

3.4.2.7. Optimization Problem

At each sampling instant, the optimization problem is solved as in Eqn. 3.89

$$\min_{U_{\text{anc}}} J(U_{\text{anc}}) \quad \text{s.t. box constraints} \tag{3.89}$$

This finite-horizon nonlinear program is solved with Sequential Quadratic Programming (SQP, `fmincon`). Although the problem is nonconvex due to nonlinear dynamics and angle handling, convergence is reliable with appropriate weights, bounds, and initialization.

3.4.2.8. Discussion and Rationale

- **Regulation:** The reference is constant, making this suitable for waypoint capture, docking, or position-hold tasks.
- **Heading unwrapping:** Continuous heading updates avoid spurious 2π jumps in the cost.
- **Move-blocking:** Reduces dimensionality (from $3N$ to $3n_c$), yields smooth controls, and guarantees that interpolated inputs stay within limits.

3.4.2.9. Possible Extensions

- Terminal cost $(x_N - r_\star)^\top P (x_N - r_\star)$ for stronger endpoint regulation.
- Input-rate penalties $\| \Delta u_k \|_S^2$ for smoother controls.
- Velocity penalties on $[u, v, \omega]$ if stronger damping is desired.
- Hard state constraints such as keep-out zones or speed limits if required.

3.4.3. Switching NMPC (SWNMPC) for a 3-DOF Planar UAV

3.4.3.1. Model and Discretization

State and input are given in Eqn. 3.90.

$$x = [\, x,\ y,\ \theta,\ u,\ v,\ \omega \,]^\top \in \mathbb{R}^6, \qquad u = [\, F_x,\ F_y,\ M_z \,]^\top \in \mathbb{R}^3 \tag{3.90}$$

Continuous-time dynamics (planar, fixed altitude) is given by Eqn. 3.91.

$$\begin{aligned} &\dot{x} = u\cos\theta - v\sin\theta, \dot{y} = u\sin\theta + v\cos\theta, \dot{\theta} = \omega \\ &\dot{u} = F_x/m + \omega v, \quad \dot{v} = F_y/m - \omega u, \quad \dot{\omega} = M_z/I_z \end{aligned} \tag{3.91}$$

Discretization uses RK4 with sampling time Δt as described in Eqn. 3.92.

$$x_{k+1} = F(x_k, u_k; \Delta t) \tag{3.92}$$

Heading is kept continuous by wrapping differences to $(-\pi, \pi]$.

3.4.3.2. Move-Blocking and Interpolation

Decision variables are *anchor inputs* as given in Eqn. 3.93.

$$U^{(j)} = [\, F_x^{(j)},\, F_y^{(j)},\, M_z^{(j)}]^\top, \quad j = 1, \ldots, n_c \tag{3.93}$$

stacked as $U_{\text{anc}} \in \mathbb{R}^{3n_c}$. The horizon inputs are obtained by piecewise-linear interpolation are given in Eqn. 3.94.

$$u_k = \text{interp}\big(\{\big(t_j^{\text{anc}}, U^{(j)}\big)\},\, t_k\big), \quad t_k = k\Delta t \tag{3.94}$$

Move-blocking reduces dimensionality, smooths inputs, and preserves actuator bounds under convex interpolation.

3.4.3.3. Switching Logic (Heading Threshold)

Let the desired waypoint be read from the first row of the trajectory input as in Eqn. 3.95.

$$\big(x_g,\, y_g,\, \theta_g^\circ\big) \quad \text{with} \quad \theta_g = \pi/180\ \theta_g^\circ \tag{3.95}$$

Compute the instantaneous heading error in degrees as described in Eqn. 3.96.

$$e_\theta^\circ = \big|\, 180/\pi\ \theta_{\text{est}} - \theta_g^\circ \big| \tag{3.96}$$

Given a threshold Θ_{th}:

- **Heading-only mode** if $e_\theta^\circ > \Theta_{\text{th}}$.
- **Translation-only mode** otherwise.

This yields a simple hybrid policy prioritizing fast yaw alignment before significant translation.

3.4.3.4. Mode A: Heading-Only Regulation

3.4.3.4.1. Reference.

A constant heading reference aligned with the current goal direction is computed as in Eqn. 3.97 from the vector $\delta = p_g - p_0$ with $p_0 = [x_0, y_0]^\top$, $p_g = [x_g, y_g]^\top$.

$$\theta_{\text{raw}} = \text{atan2}(\delta_y, \delta_x), \qquad \theta_{\text{ref}}(k) = \theta_{\text{prev}} + \text{wrapToPi}(\theta_{\text{raw}} - \theta_{\text{prev}}) \ \ \forall k \qquad (3.97)$$

3.4.3.4.2. Input structure.

Only yaw moment is allowed; body-force inputs are suppressed as given in Eqn. 3.98.

$$u_k = \begin{bmatrix} 0 \\ 0 \\ M_{z,k} \end{bmatrix} \qquad (3.98)$$

3.4.3.4.3. Error groups and cost.

Maintain an unwrapped heading ψ_k as described in Eqn. 3.99.

$$\psi_k = \psi_{k-1} + \text{wrapToPi}(\theta_k - \psi_{k-1}), \qquad \psi_1 = \theta_1 \qquad (3.99)$$

Defining $e_{\theta,k} = \psi_k - \theta_{\text{ref}}(k)$, with weights $Q = \text{diag}(q_x, q_y, q_\theta)$ and $R = \text{diag}(0,0, r_\theta)$, the stage cost is calculated as in Eqn. 3.100,

$$\ell_k = q_\theta \, e_{\theta,k}^2 + u_k^\top R u_k \qquad (3.100)$$

while the horizon cost is $J = \sum_{k=1}^{N} \ell_k$.

3.4.3.5. Mode B: Translation-Only Tracking (Heading Maintenance)

3.4.3.5.1. Reference.

Linear interpolation of position from current to goal over the horizon is given in Eqn. 3.101.

$$p_{\text{ref}}(k) = (1 - \alpha_k) p_0 + \alpha_k p_g, \qquad \alpha_k = \frac{k-1}{N-1} \qquad (3.101)$$

3.4.3.5.2. Input structure.

Only body forces are allowed; yaw moment is suppressed as described in Eqn. 3.102

$$u_k = \begin{bmatrix} F_{x,k} \\ F_{y,k} \\ 0 \end{bmatrix} \qquad (3.102)$$

3.4.3.5.3. Error groups and cost.

With position error $e_{p,k} = [\, x_k - x_{\text{ref}}(k),\ y_k - y_{\text{ref}}(k) \,]^\top$ while $Q_{xy} = \text{diag}(q_x, q_y)$ and $R = \text{diag}(r_x, r_y, 0)$, costs are calculated as given in Eqn. 3.103.

$$\ell_k = e_{p,k}^{\top} Q_{xy} e_{p,k} + u_k^{\top} R u_k, \qquad J = \sum_{k=1}^{N} \ell_k \tag{3.103}$$

3.4.3.6. Constraints

Anchor inputs satisfy actuator box limits as in Eqn. 3.104.

$$-F_{\max} \le F_x^{(j)}, F_y^{(j)} \le F_{\max}, \qquad -M_{\max} \le M_z^{(j)} \le M_{\max} \tag{3.104}$$

By convex interpolation, the interpolated u_k respect the same bounds.

3.4.3.7. Optimization at Each Sampling Instant

Given x_0, Q, and the active mode (chosen via the threshold rule), the finite-horizon NLP is given in Eqn. 3.105.

$$\min_{U_{\text{anc}}} J(U_{\text{anc}}) \quad \text{s.t. box constraints} \tag{3.105}$$

with dynamics enforced inside the cost through the prediction map $x_{k+1} = F(x_k, u_k; \Delta t)$. A Sequential Quadratic Programming solver (SQP, `fmincon`) is used. The problem is nonlinear and nonconvex, but the move-blocked parameterization, heading unwrapping, and mode-specific input structure improve numerical robustness.

3.4.3.8. Behavior and Design Rationale

- **Priority to yaw alignment.** When heading error is large, the controller corrects yaw first (no translation), reducing lateral slip and overshoot.
- **Efficient translation.** Once aligned, translation proceeds along a straight path with heading held implicitly by $M_z = 0$.
- **Smoothness and feasibility.** Move-blocking yields smooth inputs and keeps interpolated controls within bounds.
- **Simple tuning.** The threshold Θ_{th} trades off alignment strictness vs. progress speed.

3.4.3.9. Possible Extensions

- Add a terminal cost $(x_N - r_N)^{\top} P (x_N - r_N)$ in Mode B for endpoint accuracy.
- Penalize input rates $\| \Delta u_k \|_S^2$ in both modes for extra smoothness.
- Include small cross-coupling penalties (e.g., $q_u \| [u, v] \|^2$) to modulate body-speed damping.

Replace the hard switch with a hysteresis band or a smooth blending weight to avoid chattering.

3.4.4. Nonlinear MPC with Linear Path and Directional Heading

3.4.4.1. Model and Discretization

The 3-DOF UAV state and input are defined as in Eqn. 3.106.

$$x = [\,x,\ y,\ \theta,\ u,\ v,\ \omega\,]^{\top} \in \mathbb{R}^6, \quad u = [\,F_x,\ F_y,\ M_z\,]^{\top} \in \mathbb{R}^3 \tag{3.106}$$

The continuous-time dynamics given in Eqn. 3.107 consist of inertial kinematics and body-frame translational/rotational dynamics.

$$\begin{aligned} &\dot{x} = u\cos\theta - v\sin\theta, \dot{y} = u\sin\theta + v\cos\theta,\ \dot{\theta} = \omega \\ &\dot{u} = F_x/m + \omega v, \quad \dot{v} = F_y/m - \omega u, \quad \dot{\omega} = M_z/I_z \end{aligned} \tag{3.107}$$

The discrete-time model given in Eqn. 3.108. uses Runge–Kutta 4 integration with step size Δt.

$$x_{k+1} = F(x_k, u_k; \Delta t) \tag{3.108}$$

Heading is updated with angle wrapping to preserve continuity.

3.4.4.2. Decision Variables via Move-Blocking

Instead of optimizing all N inputs directly, the method optimizes n_c *anchor inputs* as given in Eqn. 3.109,

$$U^{(j)} = [\,F_x^{(j)},\ F_y^{(j)},\ M_z^{(j)}]^{\top}, \quad j = 1, \dots, n_c \tag{3.109}$$

stacked as in Eqn. 3.110.

$$U_{\text{anc}} = [(U^{(1)})^{\top} \quad \dots \quad (U^{(n_c)})^{\top}]^{\top} \in \mathbb{R}^{3n_c} \tag{3.110}$$

The control over the horizon is recovered by linear interpolation as given in Eqn. 3.111.

$$u_k = \text{interp}\big(\{(t_j^{\text{anc}}, U^{(j)})\},\ t_k\big), \quad t_k = k\Delta t \tag{3.111}$$

This *move-blocking* approach reduces dimensionality, smooths the control input, and preserves box constraints under interpolation.

3.4.4.3. Reference Generation

Given the current state x_0 and a goal waypoint (x_g, y_g, θ_g), the reference trajectory is constructed as follows.

3.4.4.3.1. Linear position interpolation.

With $p_0 = [x_0,\, y_0]^\top$ and $p_g = [x_g,\, y_g]^\top$, Eqn. 3.112 is defined.

$$p_{\text{ref}}(k) = (1 - \alpha_k)p_0 + \alpha_k p_g, \quad \alpha_k = \frac{k-1}{N-1} \tag{3.112}$$

3.4.4.3.2. Directional heading.

The reference heading follows the tangent of the interpolated path as given by Eqn. 3.113.

$$\begin{aligned} \delta_k &= p_{\text{ref}}(k+1) - p_{\text{ref}}(k) \\ \theta_{\text{raw},k} &= \text{atan2}\big(\delta_{k,y},\, \delta_{k,x}\big) \\ \theta_{\text{ref},k} &= \theta_{\text{ref},k-1} + \text{wrapToPi}\big(\theta_{\text{raw},k} - \theta_{\text{ref},k-1}\big) \end{aligned} \tag{3.113}$$

The final reference vector is obtained as in Eqn. 3.114.

$$r_k = \begin{bmatrix} x_{\text{ref}}(k) \\ y_{\text{ref}}(k) \\ \theta_{\text{ref}}(k) \end{bmatrix} \tag{3.114}$$

3.4.4.4. Prediction and Error Variables

Predicted states $\{x_k\}$ are generated with $x_{k+1} = F(x_k, u_k; \Delta t)$. Tracking errors are grouped as in Eqn. 3.115.

$$e_{p,k} = \begin{bmatrix} x_k - x_{\text{ref}}(k) \\ y_k - y_{\text{ref}}(k) \end{bmatrix}, \quad e_{\theta,k} = \text{wrapToPi}\big(\theta_k - \theta_{\text{ref}}(k)\big) \tag{3.115}$$

3.4.4.5. Stage Cost and Objective

With $Q = \text{diag}(q_x, q_y, q_\theta)$ and $R = \text{diag}(r_x, r_y, r_\theta)$, the stage cost is given in Eqn. 3.116.

$$\ell_k = e_{p,k}^\top Q_{xy} e_{p,k} + q_\theta\, e_{\theta,k}^2 + u_k^\top R\, u_k \tag{3.116}$$

where $Q_{xy} = \text{diag}(q_x, q_y)$.

The horizon cost is calculated as in Eqn. 3.117.

$$J(U_{\text{anc}}) = \sum_{k=1}^{N} \ell_k \tag{3.117}$$

3.4.4.6. Constraints and Solver

Anchor controls are bounded as in Eqn. 3.118.

$$-F_{\max} \le F_x^{(j)}, F_y^{(j)} \le F_{\max}, \quad -M_{\max} \le M_z^{(j)} \le M_{\max} \tag{3.118}$$

Since interpolation is convex, these limits also hold for interpolated u_k.

The resulting nonlinear program given in Eqn. 3.119 is solved at each step.

$$\min_{U_{\text{anc}}} J(U_{\text{anc}}) \quad \text{s.t. box constraints} \tag{3.119}$$

A Sequential Quadratic Programming solver (SQP, `fmincon`) is used. Despite nonconvex dynamics, the formulation converges reliably with proper weights and initialization.

3.4.4.7. Advantages of the Linear Path Formulation

- Straight-line position reference ensures predictable transient behavior.
- Directional heading aligns the vehicle with the path direction and maintains angle continuity.
- Move-blocking yields smooth inputs with reduced decision variables.

3.4.5. Nonlinear MPC Formulation with Exponential Path and Continuous Heading

3.4.5.1. Model and Discretization

The UAV state and input vectors are given in Eqn. 3.120.

$$x = [\, x,\ y,\ \theta,\ u,\ v,\ \omega \,]^{\mathsf{T}} \in \mathbb{R}^6, \qquad u = [\, F_x,\ F_y,\ M_z \,]^{\mathsf{T}} \in \mathbb{R}^3 \tag{3.120}$$

Continuous dynamics given in Eqn. 3.121 consist of inertial kinematics and body-frame dynamics.

$$\begin{aligned} &\dot{x} = u\cos\theta - v\sin\theta, \dot{y} = u\sin\theta + v\cos\theta, \ \dot{\theta} = \omega \\ &\dot{u} = F_x/m + \omega v, \quad \dot{v} = F_y/m - \omega u, \quad \dot{\omega} = M_z/I_z \end{aligned} \tag{3.121}$$

The discrete prediction model is described in Eqn. 3.122 with sampling time Δt.

$$x_{k+1} = F(x_k, u_k; \Delta t) \tag{3.122}$$

where $F(\cdot)$ denotes the RK4 integration. Heading angles are kept continuous with wrap-to-π updates.

3.4.5.2. Decision Variables via Move-Blocking

Instead of optimizing each u_k for $k = 0, \dots, N-1$, the controller optimizes a smaller set of *anchor controls* as given in Eqn. 3.123.

$$U^{(j)} = [\, F_x^{(j)},\, F_y^{(j)},\, M_z^{(j)}]^\top, \quad j = 1, \ldots, n_c \tag{3.123}$$

These are stacked into Eqn. 3.124.

$$U_{\text{anc}} = [(U^{(1)})^\top \quad \ldots \quad (U^{(n_c)})^\top]^\top \in \mathbb{R}^{3n_c} \tag{3.124}$$

The full input sequence is obtained by piecewise-linear interpolation as in Eqn. 3.125.

$$u_k = \text{interp}\big(\{(t_j^{\text{anc}}, U^{(j)})\},\, t_k\big), \quad t_k = k\Delta t \tag{3.125}$$

This *move-blocking* reduces decision dimension, smooths inputs, and preserves bounds under convex interpolation.

3.4.5.3. Reference Generation (Exponential Position + Continuous Heading)

A soft point-to-point reference is generated from the current state toward the goal waypoint.

3.4.5.3.1. Exponential position reference.

With $p_0 = [x_0,\, y_0]^\top$ and $p_g = [x_g,\, y_g]^\top$, the exponential approach is described in Eqn. 3.126.

$$p_{\text{ref}}(t) = p_0 + \big(1 - e^{-\lambda t}\big)\big(p_g - p_0\big), \quad \lambda > 0 \tag{3.126}$$

3.4.5.3.2. Continuous heading reference.

The raw heading is given in Eqn. 3.127.

$$\theta_{\text{raw}} = \text{atan2}\big(y_g - y_0,\, x_g - x_0\big) \tag{3.127}$$

For continuity, Eqn. 3.128 is obtained.

$$\theta_{\text{ref}} = \theta_{\text{prev}} + \text{wrapToPi}\big(\theta_{\text{raw}} - \theta_{\text{prev}}\big) \tag{3.128}$$

held constant along the horizon and updated between calls.

Thus, the step-k reference is given in Eqn. 3.129.

$$r_k = [\, x_{\text{ref}}(k),\; y_{\text{ref}}(k),\; \theta_{\text{ref}}(k)\,]^\top \tag{3.129}$$

3.4.5.4. Prediction and Error Variables

Predicted states $\{x_k\}_{k=1}^{N}$ are obtained by applying the discrete model with interpolated controls.

Tracking errors are defined as given in Eqn. 3.130.

$$e_{p,k} = \begin{bmatrix} x_k - x_{\text{ref}}(k) \\ y_k - y_{\text{ref}}(k) \end{bmatrix}, \qquad e_{\theta,k} = \text{wrapToPi}\big(\theta_k - \theta_{\text{ref}}(k)\big) \tag{3.130}$$

3.4.5.5. Stage Cost and Total Objective

Weights are given in Eqn. 3.131.

$$Q = \text{diag}\big(q_x, q_y, q_\theta\big) \succcurlyeq 0, \qquad R = \text{diag}\big(r_x, r_y, r_\theta\big) \succcurlyeq 0 \tag{3.131}$$

Let $Q_{xy} = \text{diag}(q_x, q_y)$. The stage cost is given in Eqn. 3.132.

$$\ell_k = e_{p,k}^\top Q_{xy} e_{p,k} + q_\theta e_{\theta,k}^2 + u_k^\top R u_k \tag{3.132}$$

The horizon cost is described in Eqn. 3.133.

$$J(U_{\text{anc}}) = \sum_{k=1}^{N} \ell_k \tag{3.133}$$

3.4.5.6. Constraints

Anchor inputs are bounded as in Eqn. 3.134.

$$-F_{\max} \le F_x^{(j)}, F_y^{(j)} \le F_{\max}, \quad -M_{\max} \le M_z^{(j)} \le M_{\max} \tag{3.134}$$

By convexity, interpolated inputs u_k remain within the same bounds.

3.4.5.7. Optimization Problem

At each sampling instant, the finite-horizon NLP is calculated from Eqn. 3.135.

$$\min_{U_{\text{anc}}} J(U_{\text{anc}}) \quad \text{s.t. box constraints} \tag{3.135}$$

This is solved using a Sequential Quadratic Programming method (SQP, `fmincon`). Although nonconvex, reliable convergence is observed with suitable weights and bounds.

3.4.5.8. Practical Rationale

- Move-blocking reduces dimension from $3N$ to $3n_c$ and yields smoother inputs.
- Exponential reference avoids abrupt jumps, improving numerical stability.
- Continuous heading eliminates wrap-around artifacts in angular error.
- Adaptive Q (optional) increases tracking aggressiveness when errors exceed thresholds.

3.4.5.9. Possible Extensions

- Add a terminal cost $(x_N - r_N)^\top P(x_N - r_N)$
- Penalize input rates Δu_k
- Extend Q to velocity states $[u, v, \omega]$
- Impose hard state constraints for obstacle avoidance

3.4.5.10. Notation Summary

- $x = [x, y, \theta, u, v, \omega]^\top$: position, heading, body velocities, yaw rate
- $u = [F_x, F_y, M_z]^\top$: body-frame forces and yaw moment
- $F(\cdot)$: RK4 discretization of the model
- $r_k = [x_{\text{ref}}(k), y_{\text{ref}}(k), \theta_{\text{ref}}(k)]^\top$: reference
- $e_{p,k}, e_{\theta,k}$: position and heading errors
- $Q = \text{diag}(q_x, q_y, q_\theta)$, $R = \text{diag}(r_x, r_y, r_\theta)$: weights
- U_{anc}: stacked anchor inputs, interpolated into $\{u_k\}$

3.5. General Assumptions

In this work, certain assumptions were made because of the simulations and calculations. Throughout the work, given rules are applied, and results are achieved accordingly.

- ✓ The height of the UAV assumed as in the level flight i.e. constant altitude
- ✓ The UAV has an onboard looking down camera which provides image frames to inspect in Virtual GPS algorithm
- ✓ UAV's camera imagery is provided from map imagery using real state of the UAV and cropping the image assuming the real UAV is above that location and looking down camera is taking pictures of that area
- ✓ Since an applicable camera an IMU have different data collection rates, in the simulation while INS measurements are achieved in 50 Hz, virtual GPS corrections are applied in 5 Hz
- ✓ Since the designed UAV's accelerometer noise and bias specifications are too low to visually observe results over map imagery, these new set of parameters as given below are used:

Table 3.2. Used accelerometer bias and noise values of the selected UAV model for simulation

Parameter	Value	Unit
Accelerometer Bias Drift (X,Y-axes)	±0.1	m/s^2
Accelerometer White Noise (X,Y-axes)	0.1	m/s^2 RMS

- ✓ While designed UAV's parameters are used to predict real state of it i.e. true source, scaled (Mass Scale = 1.05, Inertial Scale = 0.975) parameters i.e. corrupted parameters are used in the INS positioning along with EKF algorithm
- ✓ For more realistic simulation, at the beginning of simulation real position of the UAV differs from the knowledge of the EKF which is 50 meters each along x-axis (east-west) and y-axis (north-south) and 10° along z-axis i.e. heading angle

4. RESULTS

4.1. Image-Based Positioning Analysis

4.1.1. Overview

The position and heading of the UAV which are used instead of GNSS source in the simulation is obtained from UAV imagery. Besides the whole simulation, another comprehensive analysis is performed to obtain accuracies for these parameters. First, test area is determined considering the environment diversity as given in Figure 4.1.

Figure 4.1. Satellite image of the image-based positioning analysis test area

Then, 5 different point sets are determined which are analyzed in further sections. These sets are a set of random hand-picked position/headings, an 8-shape route, a circle route, a diamond route and a bowtie route. With different approaches, it is aimed to observe variety of position integrity for routes and individual virtual GPS analysis performance of the algorithm. Noting that the algorithm is not expected to achieve high precision in ambiguous areas such as flat surfaces or water surfaces which do not have any correlation in such frame. Nevertheless, these areas will be inspected in the integrated simulation and fully taken into account in the process including accepting/rejecting decision.

Because of the same algorithm used in the simulation run within the scope of thesis, virtual positions and headings considered as physical attitude of the UAV. Similarly, the

EKF algorithm which selects the integrity of the visual based calculation, estimations within the selected range of errors are accepted while the others are rejected. Main acceptance criteria are selected as:

- ✓ Absolute resultant position error is lower than 10 meters or absolute heading angle error is lower than 5°, those who cannot satisfy these criteria are rejected and indicated as red
- ✓ Low error (indicated as green): position error is lower than the sum of mean error and standard deviation
- ✓ Medium error (indicated as yellow): position error is higher than the low error limit and lower than the sum of mean error and two times standard deviation
- ✓ High error (indicated as orange): position error is higher than the medium error limit and lower than 10 meters absolute value of position error

Likely, invalid results are rejected in this process since they cannot pass the acceptance criteria in the simulation.

With the known positions, estimated positions and headings of the virtual UAVs are compared. This comparison is confined to the accepted estimations, since rejected estimation contains indetermined or largely miscalculated solutions.

4.1.2. Random Hand-Picked States of the UAV

Simulating UAVs in virtual positions requires the heading parameters which are preferred as random angles. Hand-picked positions and randomly selected headings are shown in Figure 4.2.

Figure 4.2. Selected Virtual UAVs as Determining the Positions and Heading Angles

In this part of the analysis, a total of 28 frames is inspected as the virtual UAVs individually. When the results are inspected, 21 of the 28 frames are accepted as usable in EKF (in the simulation constructed within the thesis). The estimation proceeds with 7 rejected frames. Rejected frames are mostly caused by lack of features because of the territory within the frame. Absolute errors obtained from estimated and true locations are presented statistically in Table 4.1 knowing that error analysis is done with only accepted frames while presenting rejected (high error) frames in Figure 4.4. Moreover, the error histogram representing in Figure 4.3 describes error distribution of virtual position analysis of accepted frames.

Table 4.1. Statistical summary of the virtual position analysis for hand-picked locations

Error Metric	x-axis [m]	y-axis [m]	Resultant [m]	Heading [°]
RMSE	1.1722	1.0934	1.6030	0.5873
MAE	1.0337	0.9603	1.5317	0.4969
Standard Deviation	0.5663	0.5357	0.4845	0.3209
Maximum Error	2.0791	2.1379	2.5890	1.2901

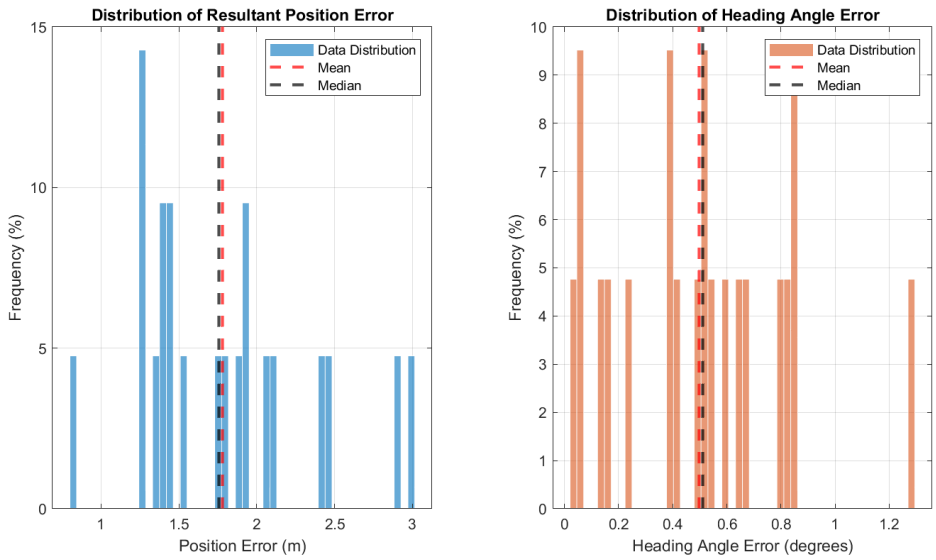

Figure 4.3. Error histogram for hand-picked locations from virtual position analysis

Besides the individual inspection and statistical results, the whole analysis scenario is given in Figure 4.4 and it can be examined elaborately. Virtual UAV locations marked as red are the ones which lack of accurate estimations.

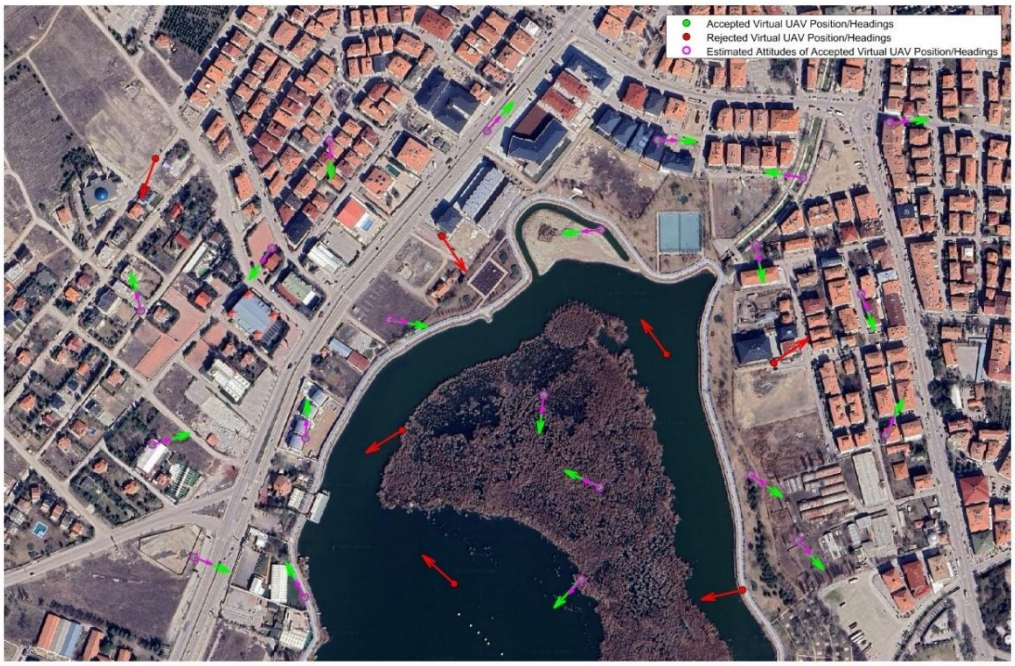

Figure 4.4. Results of the virtual position analysis for hand-picked locations over satellite image

4.1.3. UAV States over the 8-Shape Path

The 8-shape is one of the main paths in this work while simulating UAV. A total of 1000 frames are individually inspected and 32 of them are inconclusive due to poor environment. On the other hand, 848 frames are analyzed with low error, 48 frames are analyzed with medium error, and 34 frames are analyzed with high error considering the criteria mentioned in Section 4.1.1. Finally, solutions in 38 frames are rejected because of the unacceptable error. Absolute errors obtained from estimated and true locations are presented statistically in Table 4.2 knowing that error analysis is done with frames except invalid ones while presenting all frames in Figure 4.5 over the satellite image. Moreover, the error histogram representing in Figure 4.6 describes error distribution of virtual position analysis of accepted frames.

Table 4.2. Statistical summary of the virtual position analysis for 8-shape trajectory

Error Metric	x-axis [m]	y-axis [m]	Resultant [m]	Heading [°]
RMSE	1.2485	1.4061	1.8804	0.7169
MAE	1.0362	1.1272	1.7046	0.4379
Standard Deviation	0.6969	0.8411	0.7943	0.5679
Maximum Error	5.8606	6.5081	7.0253	4.9524

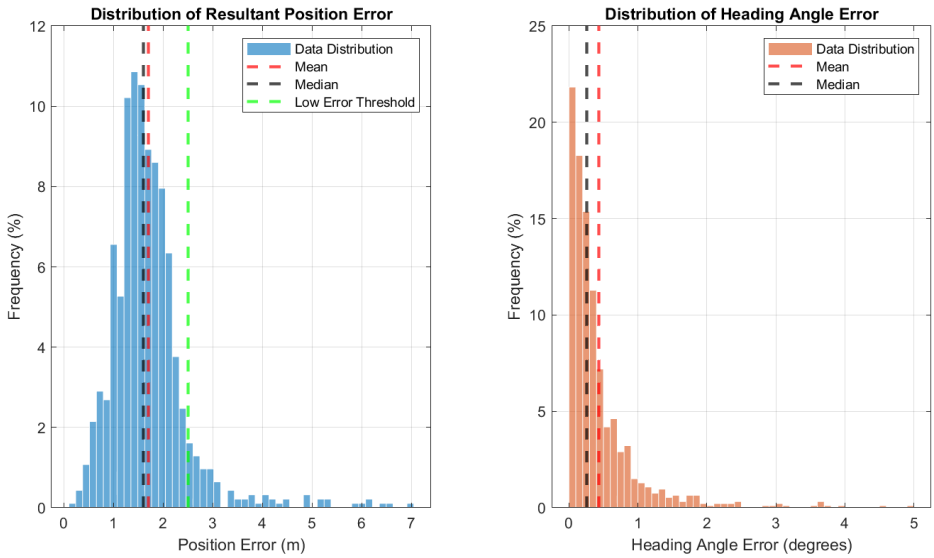

Figure 4.5. Error histogram for 8-shape trajectory from virtual position analysis

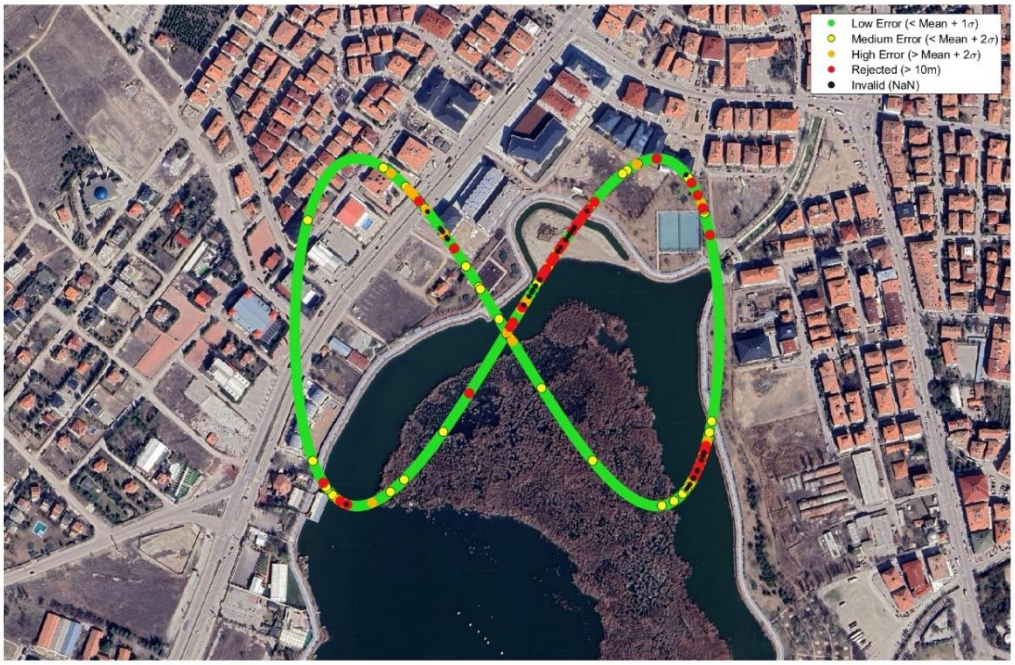

Figure 4.6. Results of the virtual position analysis for 8-shape trajectory over satellite image

4.1.4. UAV States over the Circular Path

In the UAV simulation although the circular path is not considered as a priority, the analysis of this path is important due to its constant heading angle change. A total of 1200 frames are individually inspected and 56 of them are inconclusive due to poor environment. On the other hand, 973 frames are analyzed with low error, 116 frames are analyzed with medium error, and 29 frames are analyzed with high error considering the criteria mentioned in Section 4.1.1. Finally, solutions in 26 frames are rejected because of the unacceptable error. Absolute errors obtained from estimated and true locations are presented statistically in Table 4.3 knowing that error analysis is done with frames except invalid ones while presenting all frames in Figure 4.7 over the satellite image. Moreover, the error histogram representing in Figure 4.8 describes error distribution of virtual position analysis of accepted frames.

Table 4.3. Statistical summary of the virtual position analysis for circular trajectory

Error Metric	x-axis [m]	y-axis [m]	Resultant [m]	Heading [°]
RMSE	1.2333	1.2307	1.7423	0.6317
MAE	1.0741	1.0321	1.6349	0.4310
Standard Deviation	0.6063	0.6708	0.6025	0.4620
Maximum Error	4.0930	6.0703	6.8700	4.9196

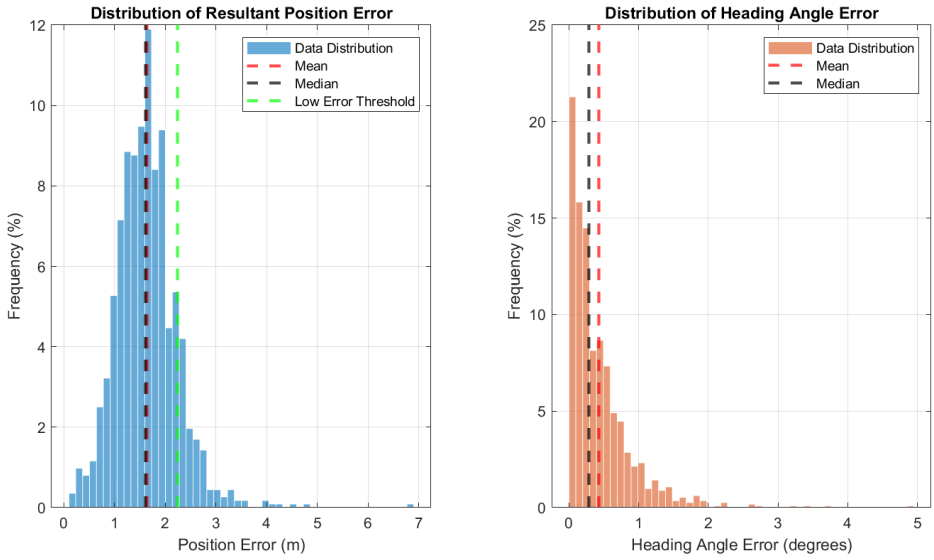

Figure 4.7. Error histogram for circular trajectory from virtual position analysis

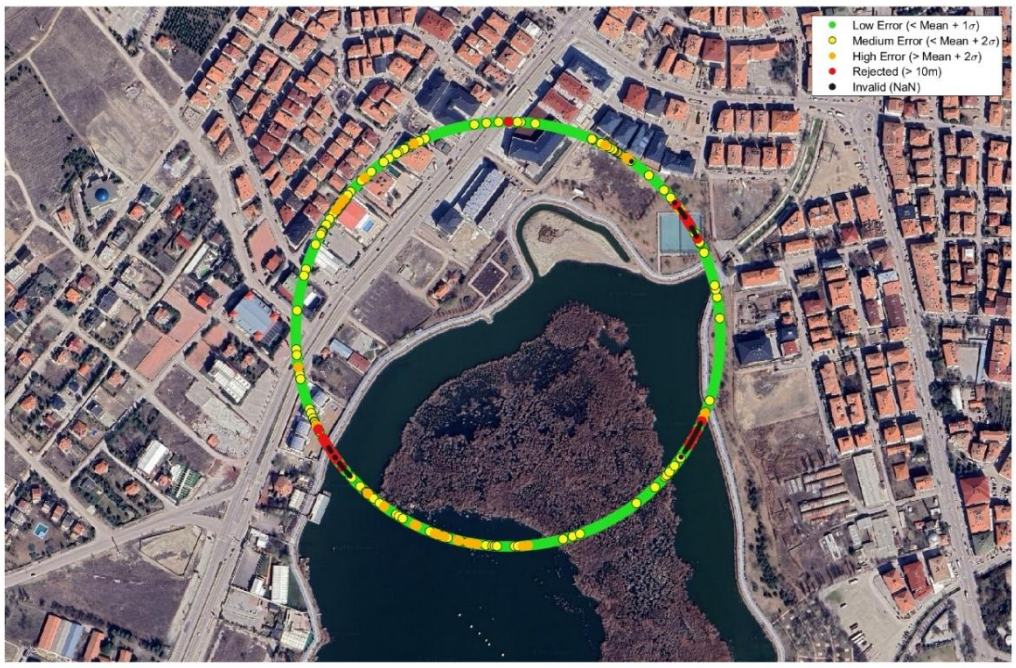

Figure 4.8. Results of the virtual position analysis for circular trajectory over satellite image

4.1.5. UAV States over the Diamond Path

The diamond path is the other main path in the simulation of UAV in this work. A total of 1997 frames are individually inspected and 113 of them are inconclusive due to poor environment. On the other hand, 1561 frames are analyzed with low error, 100 frames are analyzed with medium error, and 76 frames are analyzed with high error considering the criteria mentioned in Section 4.1.1. Finally, solutions in 147 frames are rejected because of the unacceptable error. Absolute errors obtained from estimated and true locations are presented statistically in Table 4.4 knowing that error analysis is done with frames except invalid ones while presenting all frames in Figure 4.9 over the satellite image. Moreover, the error histogram representing in Figure 4.10 describes error distribution of virtual position analysis of accepted frames.

Table 4.4. Statistical summary of the virtual position analysis for diamond trajectory

Error Metric	x-axis [m]	y-axis [m]	Resultant [m]	Heading [°]
RMSE	1.4940	1.5140	2.1270	0.9909
MAE	1.1421	1.1721	1.8560	0.6491
Standard Deviation	0.9635	0.9585	1.0392	0.7488
Maximum Error	8.6842	7.8333	9.6839	4.9661

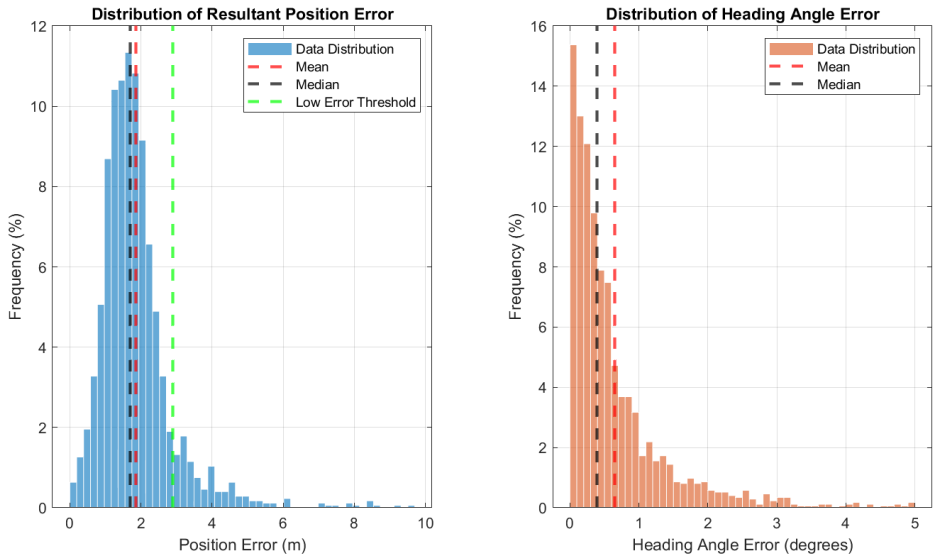

Figure 4.9. Error histogram for diamond trajectory from virtual position analysis

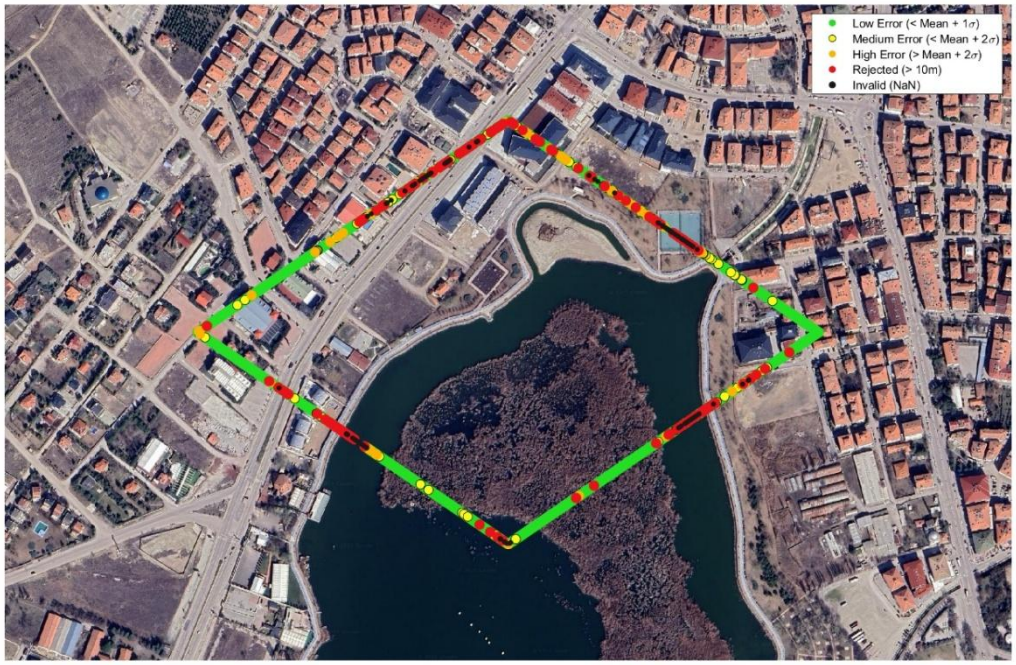

Figure 4.10. Results of the virtual position analysis for diamond trajectory over satellite image

4.1.6. UAV States over the Bowtie Path

In the UAV simulation bowtie path is the last scenario that is investigated in the image-based positioning analysis. A total of 1975 frames are individually inspected and 243 of them are inconclusive due to poor environment. On the other hand, 1438 frames are analyzed with low error, 140 frames are analyzed with medium error, and 53 frames are analyzed with high error considering the criteria mentioned in Section 4.1.1. Finally, solutions in 101 frames are rejected because of the unacceptable error. Absolute errors obtained from estimated and true locations are presented statistically in Table 4.5 knowing that error analysis is done with frames except invalid ones while presenting all frames in Figure 4.11 over the satellite image. Moreover, the error histogram representing in Figure 4.12 describes error distribution of virtual position analysis of accepted frames.

Table 4.5. Statistical summary of the virtual position analysis for bowtie trajectory

Error Metric	x-axis [m]	y-axis [m]	Resultant [m]	Heading [°]
RMSE	1.1642	1.2962	1.7423	0.6982
MAE	0.8533	0.9886	1.4860	0.3931
Standard Deviation	0.7922	0.8387	0.9098	0.5772
Maximum Error	7.6806	8.0102	8.4979	4.9998

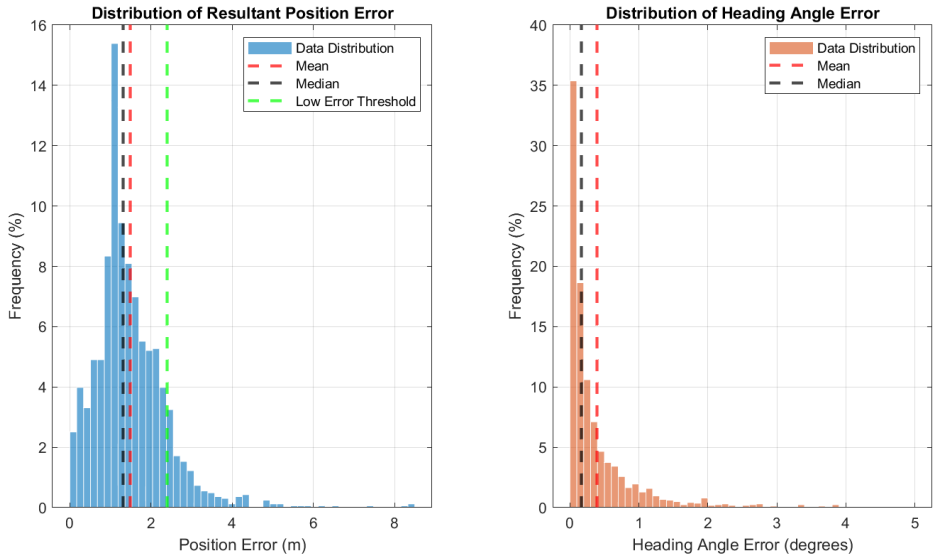

Figure 4.11. Error histogram for bowtie trajectory from virtual position analysis

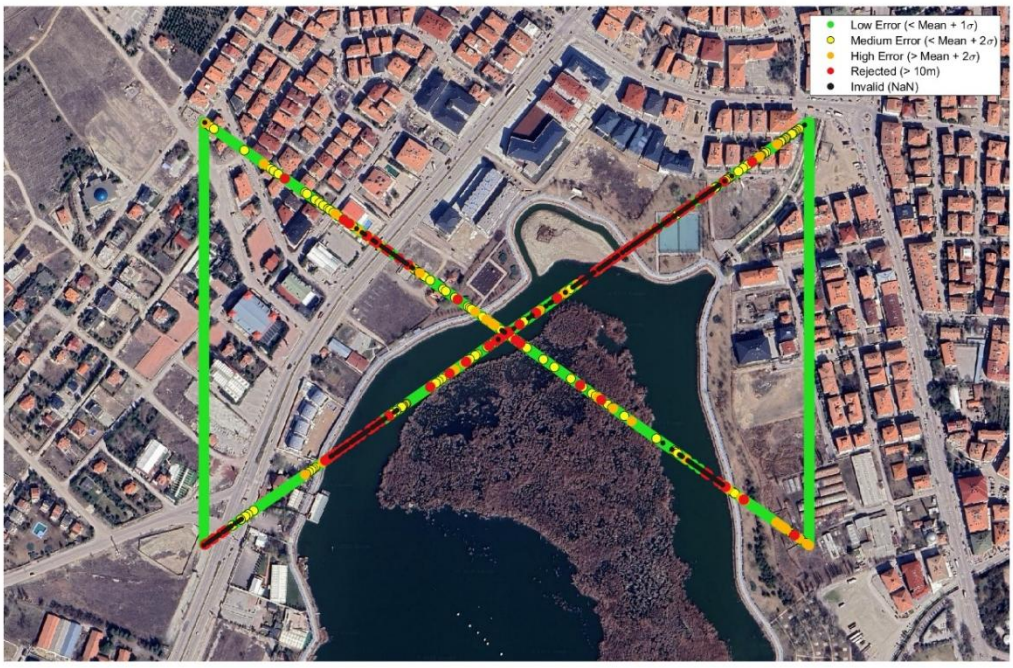

Figure 4.12. Results of the virtual position analysis for bowtie trajectory over satellite image

4.2. Visual-Inertial Fusion and NMPC Tests

Analyses are conducted with 4 different approaches as mentioned in Section 3.4. Cost functions in NMPC have been built with these differences. Moreover, for each cost calculation method, 2 different pre-defined paths are built and simulated which are 8-shape and diamond routes. Those paths are shown in Figure 4.13 over the satellite image.

Figure 4.13. 8-shape and diamond trajectory of UAV used in the simulation over the satellite image

In addition, since the simulation depends on virtual GPS, visual localization problems in simulations should also be considered separately. Therefore, 5 different visual loss scenarios are prepared i.e. no correction from virtual GPS algorithm for a specified time span. These 5 scenarios run for 4 methods with 2 different paths. By observing UAV behavior in total of 40 scenarios in the selected compelling environment, comparison in cost calculation methods is inspected. The 5 different visual loss scenarios are listed and explained below.

- ✓ **Scenario 1:** No Visual Loss: Virtual GPS is running in whole simulation, yet it still has the localization challenge due to geographical formations in the area like all other scenarios

- ✓ **Scenario 2:** Whole Time Visual Loss: This scenario is prepared to understand INS navigation behavior in UAV well and visualize how far uncorrected position of UAV can go
- ✓ **Scenario 3:** First Section Visual Loss: Virtual GPS is manipulatively stopped from the beginning to a certain time of the UAV's flight which corresponds to approximately 800 frames from the start point.
- ✓ **Scenario 4:** Middle Section Visual Loss: In this scenario after a certain time passed (approximately 600 frames) with virtual GPS correction of UAV's flight, global localization is terminated for middle section flight for approximately 1100 frames. After the localization restarts which initializes position recovery, UAV's flight continues until the end.
- ✓ **Scenario 5:** First and Last Section Visual Loss: UAV cannot achieve Visual GPS localization in the first part (approximately 400 frames) of its flight and localization starts with an error in this scenario. Then, after approximately 600 frames the visual localization is again shut off. Since the regaining window is relatively small among other scenarios, regaining correct position of the UAV can be inspected for different cost methods and paths.

4.2.1. Case Study Results

Among all simulation results, the NMPC mode which includes linear path and directional heading (as mentioned in Section 3.4.4) is inspected. With this mode, simulations are run for 5 different vGPS loss scenarios and 2 different trajectories. The results of these simulations are presented in between Figure 4.14 and Figure 4.23.

Moreover, including the linear mode, case studies conducted whole together presented briefly in Table 4.6. The corresponding results for each study are also shown in the table as reference. These results include locations and heading angles obtained from trajectory, EKF output, and states of the UAV which is assumed as true source. In the results there is also difference in the resultant position for all frames including presentations for deliberately placed virtual GPS loss sections in the case studies. The other results besides the NMPC mode with Linear path, are given as packed in APPENDIX A.

Table 4.6. Case studies -and their result references- used in the whole simulation with disparities with respect to method, scenario and trajectory

Case Study #	NMPC method	GPS Scenario	Trajectory	Reference Results
1	Regulating	Scenario 1	8-Shape Path	Figure A.1
2	Regulating	Scenario 1	Diamond Path	Figure A.2
3	Switching	Scenario 1	8-Shape Path	Figure A.3
4	Switching	Scenario 1	Diamond Path	Figure A.4
5	Linear	Scenario 1	8-Shape Path	Figure 4.14
6	Linear	Scenario 1	Diamond Path	Figure 4.15
7	Exponential	Scenario 1	8-Shape Path	Figure A.5
8	Exponential	Scenario 1	Diamond Path	Figure A.6
9	Regulating	Scenario 2	8-Shape Path	Figure A.7
10	Regulating	Scenario 2	Diamond Path	Figure A.8
11	Switching	Scenario 2	8-Shape Path	Figure A.9
12	Switching	Scenario 2	Diamond Path	Figure A.10
13	Linear	Scenario 2	8-Shape Path	Figure 4.16
14	Linear	Scenario 2	Diamond Path	Figure 4.17
15	Exponential	Scenario 2	8-Shape Path	Figure A.11
16	Exponential	Scenario 2	Diamond Path	Figure A.12
17	Regulating	Scenario 3	8-Shape Path	Figure A.13
18	Regulating	Scenario 3	Diamond Path	Figure A.14
19	Switching	Scenario 3	8-Shape Path	Figure A.15
20	Switching	Scenario 3	Diamond Path	Figure A.16
21	Linear	Scenario 3	8-Shape Path	Figure 4.18
22	Linear	Scenario 3	Diamond Path	Figure 4.19
23	Exponential	Scenario 3	8-Shape Path	Figure A.17
24	Exponential	Scenario 3	Diamond Path	Figure A.18
25	Regulating	Scenario 4	8-Shape Path	Figure A.19
26	Regulating	Scenario 4	Diamond Path	Figure A.20
27	Switching	Scenario 4	8-Shape Path	Figure A.21
28	Switching	Scenario 4	Diamond Path	Figure A.22
29	Linear	Scenario 4	8-Shape Path	Figure 4.20
30	Linear	Scenario 4	Diamond Path	Figure 4.21
31	Exponential	Scenario 4	8-Shape Path	Figure A.23
32	Exponential	Scenario 4	Diamond Path	Figure A.24
33	Regulating	Scenario 5	8-Shape Path	Figure A.25
34	Regulating	Scenario 5	Diamond Path	Figure A.26
35	Switching	Scenario 5	8-Shape Path	Figure A.27
36	Switching	Scenario 5	Diamond Path	Figure A.28
37	Linear	Scenario 5	8-Shape Path	Figure 4.22
38	Linear	Scenario 5	Diamond Path	Figure 4.23
39	Exponential	Scenario 5	8-Shape Path	Figure A.29
40	Exponential	Scenario 5	Diamond Path	Figure A.30

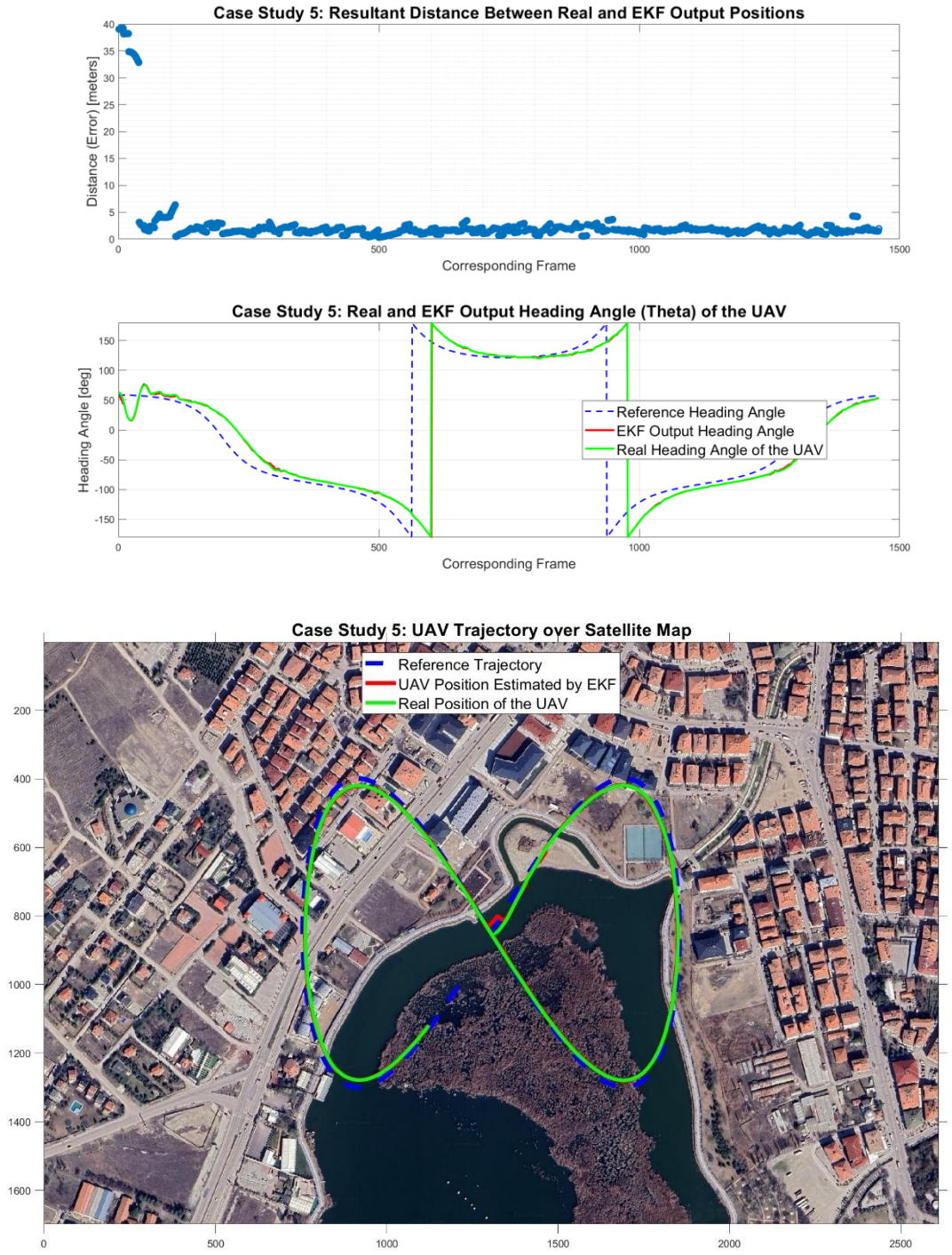

Figure 4.14. Case study 5 (MPC mode Linear, Scenario 1 -no vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

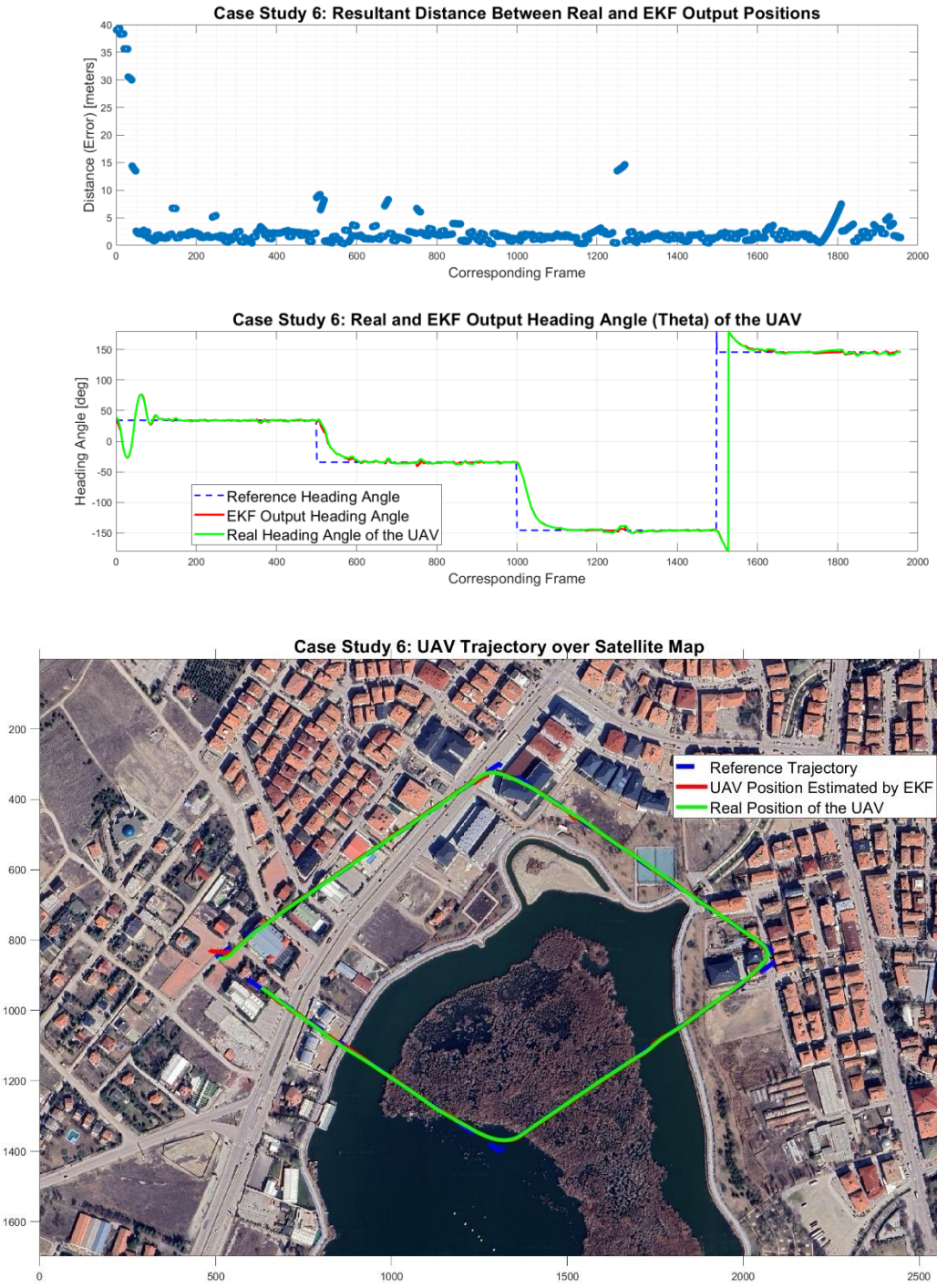

Figure 4.15. Case study 6 (MPC mode Linear, Scenario 1 -no vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

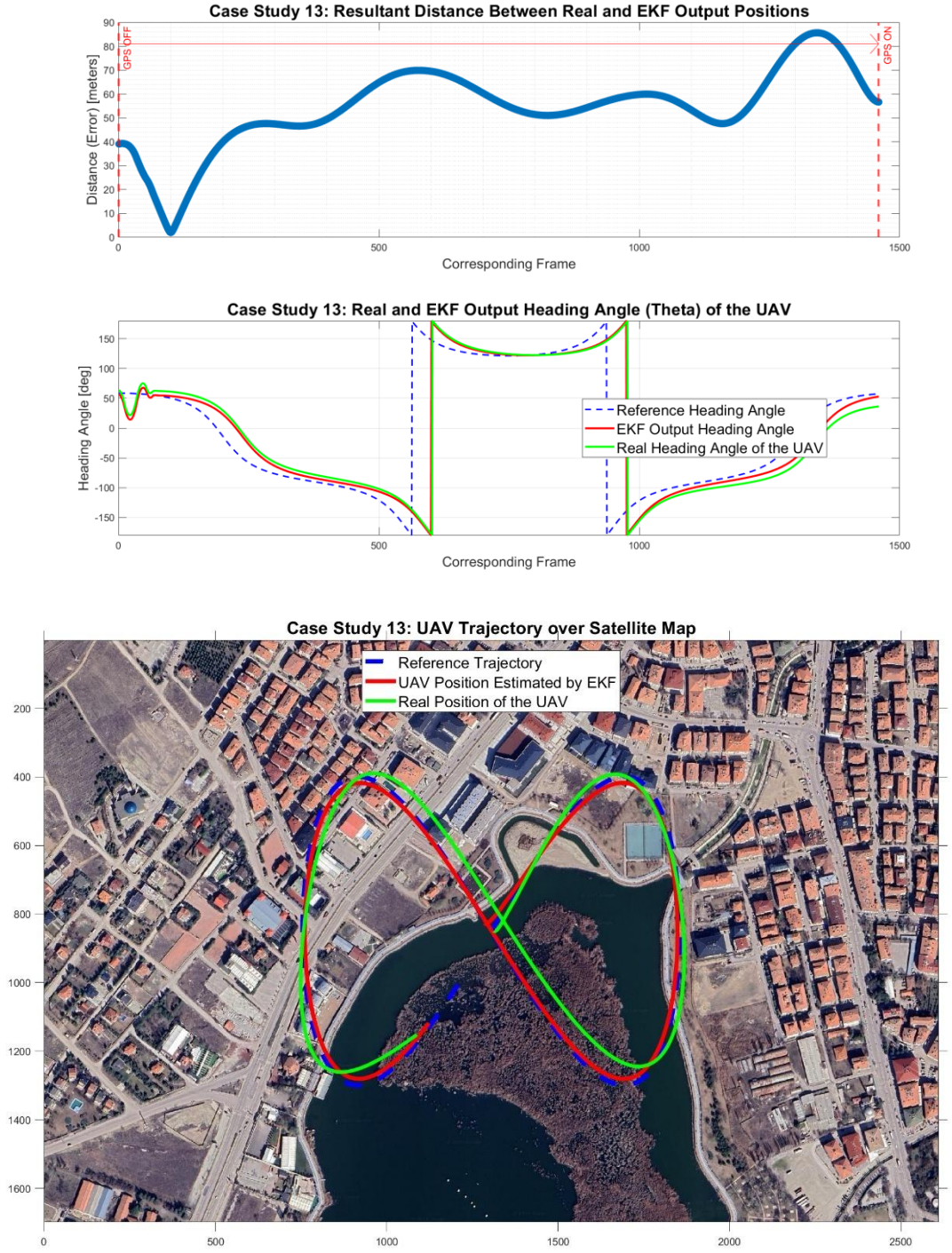

Figure 4.16. Case study 13 (MPC mode Linear, Scenario 2 -whole time vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

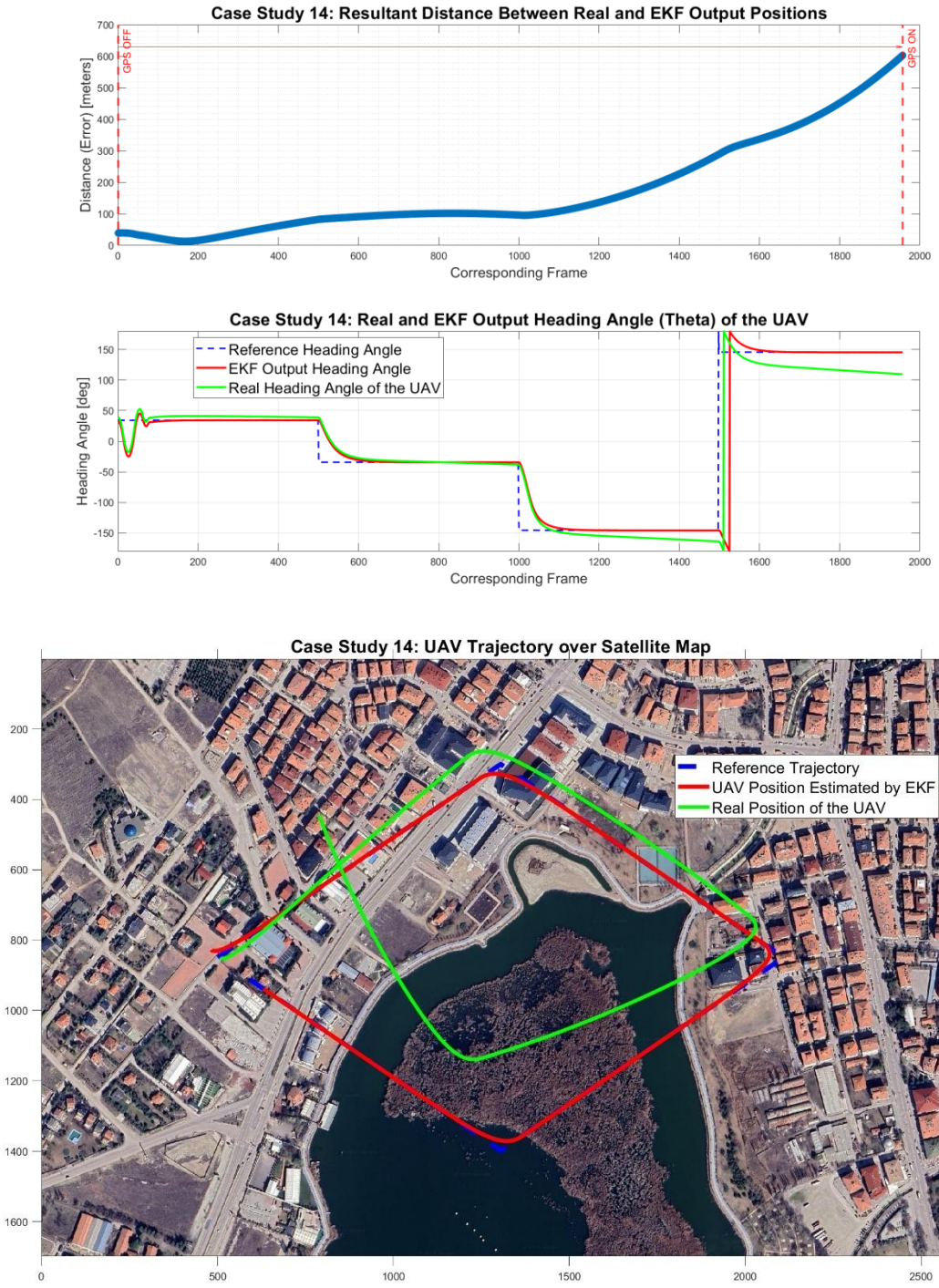

Figure 4.17. Case study 14 (MPC mode Linear, Scenario 2 -whole time vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

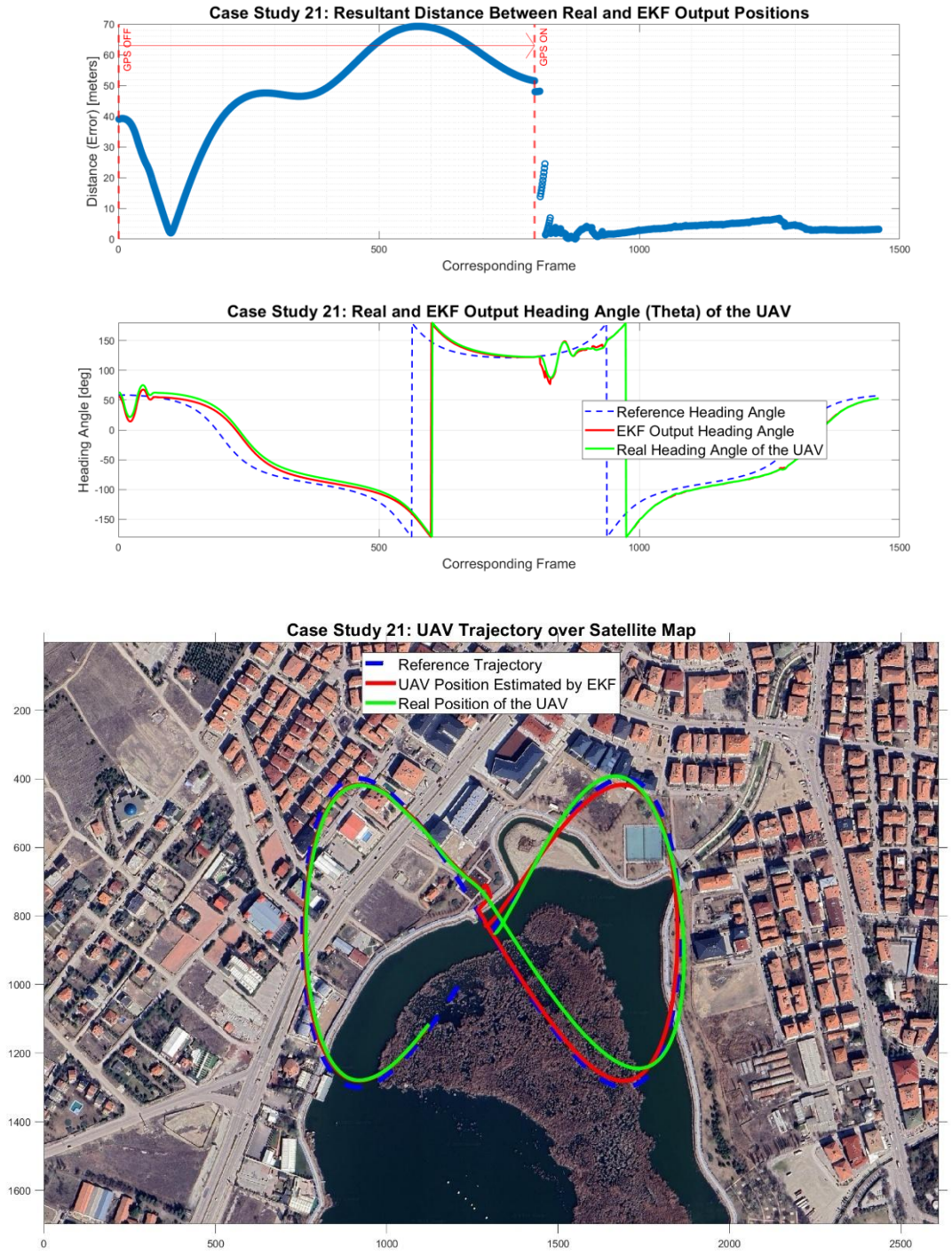

Figure 4.18. Case study 21 (MPC mode Linear, Scenario 3 -first section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

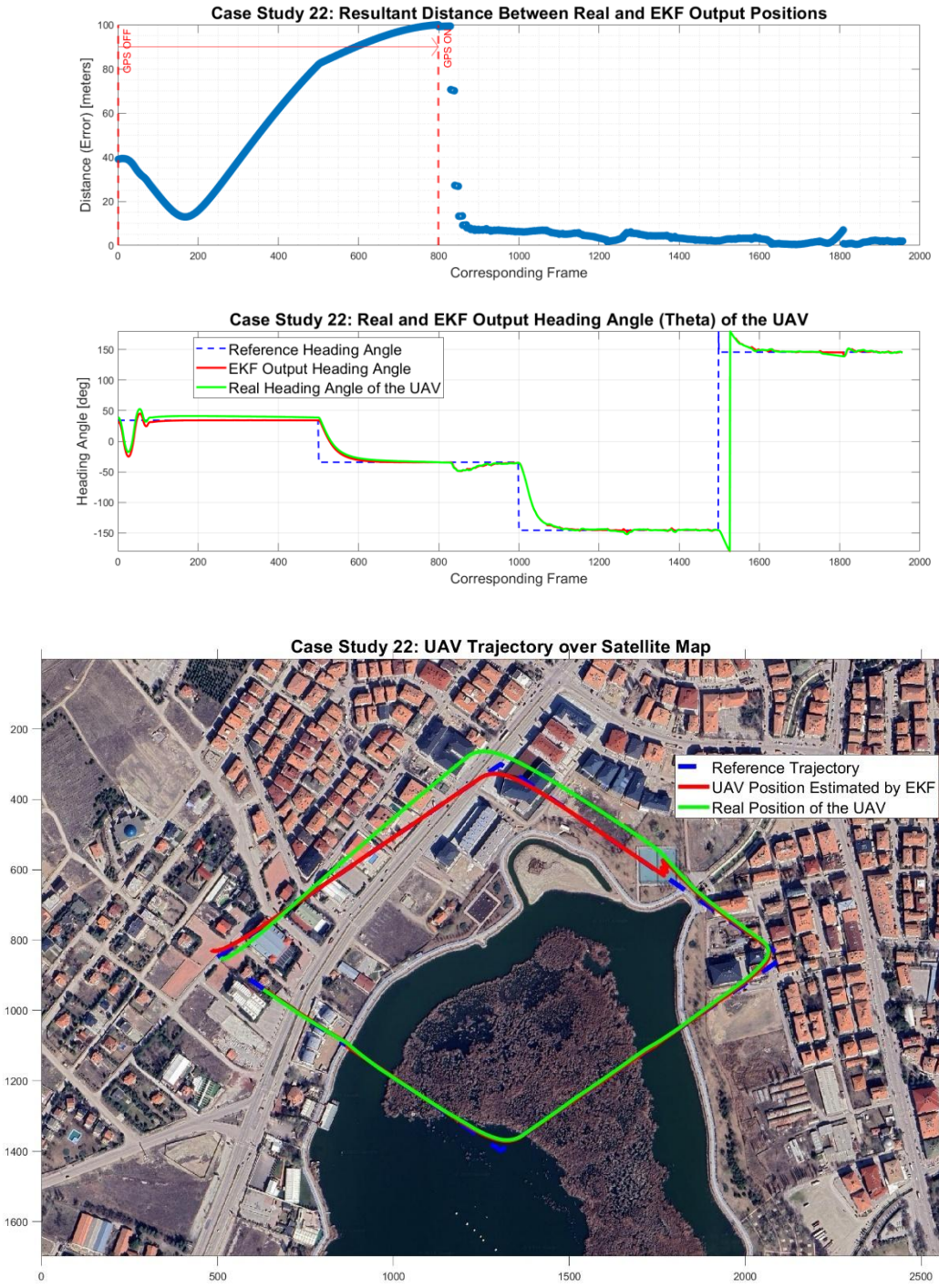

Figure 4.19. Case study 22 (MPC mode Linear, Scenario 3 -first section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

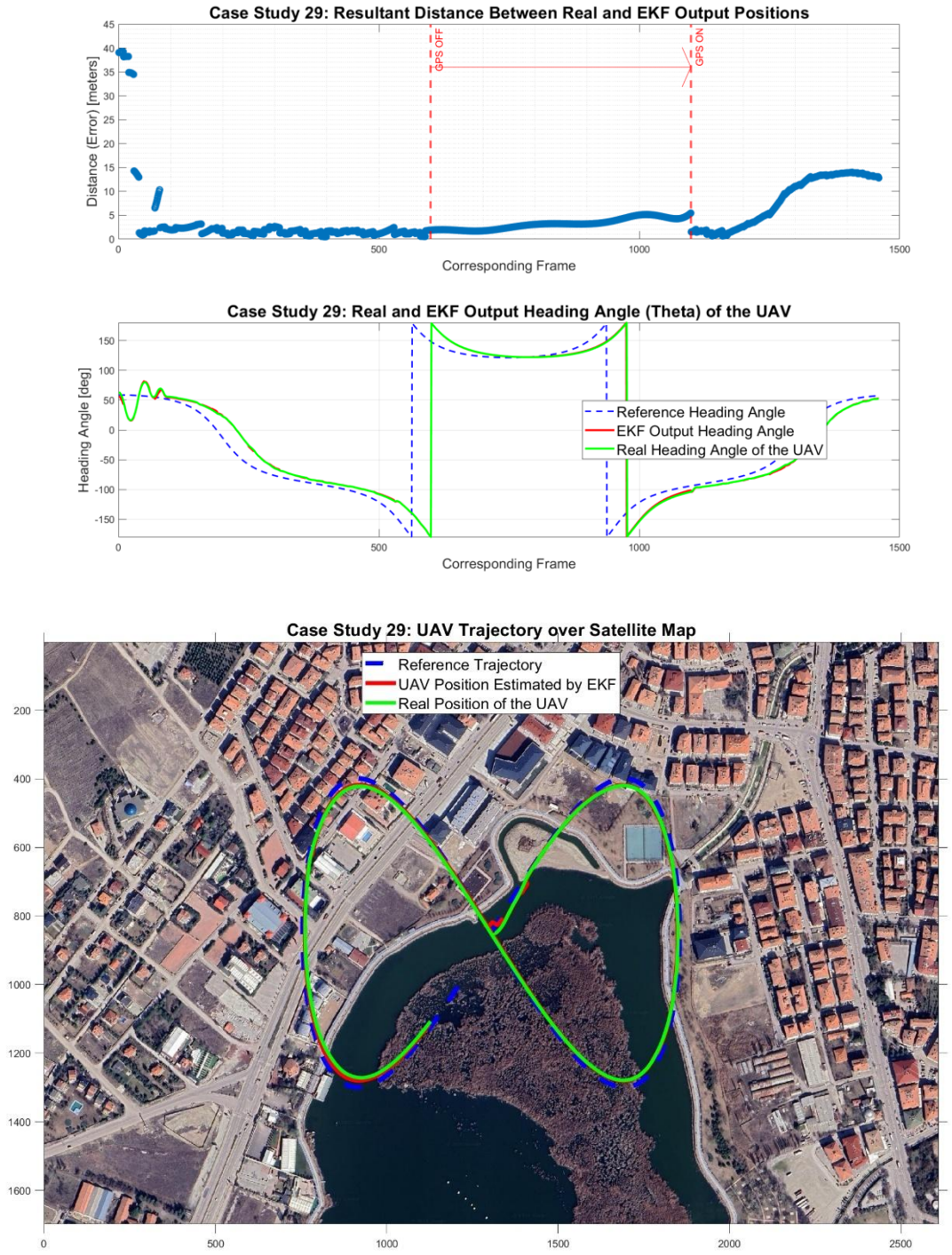

Figure 4.20. Case study 29 (MPC mode Linear, Scenario 4 -middle section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

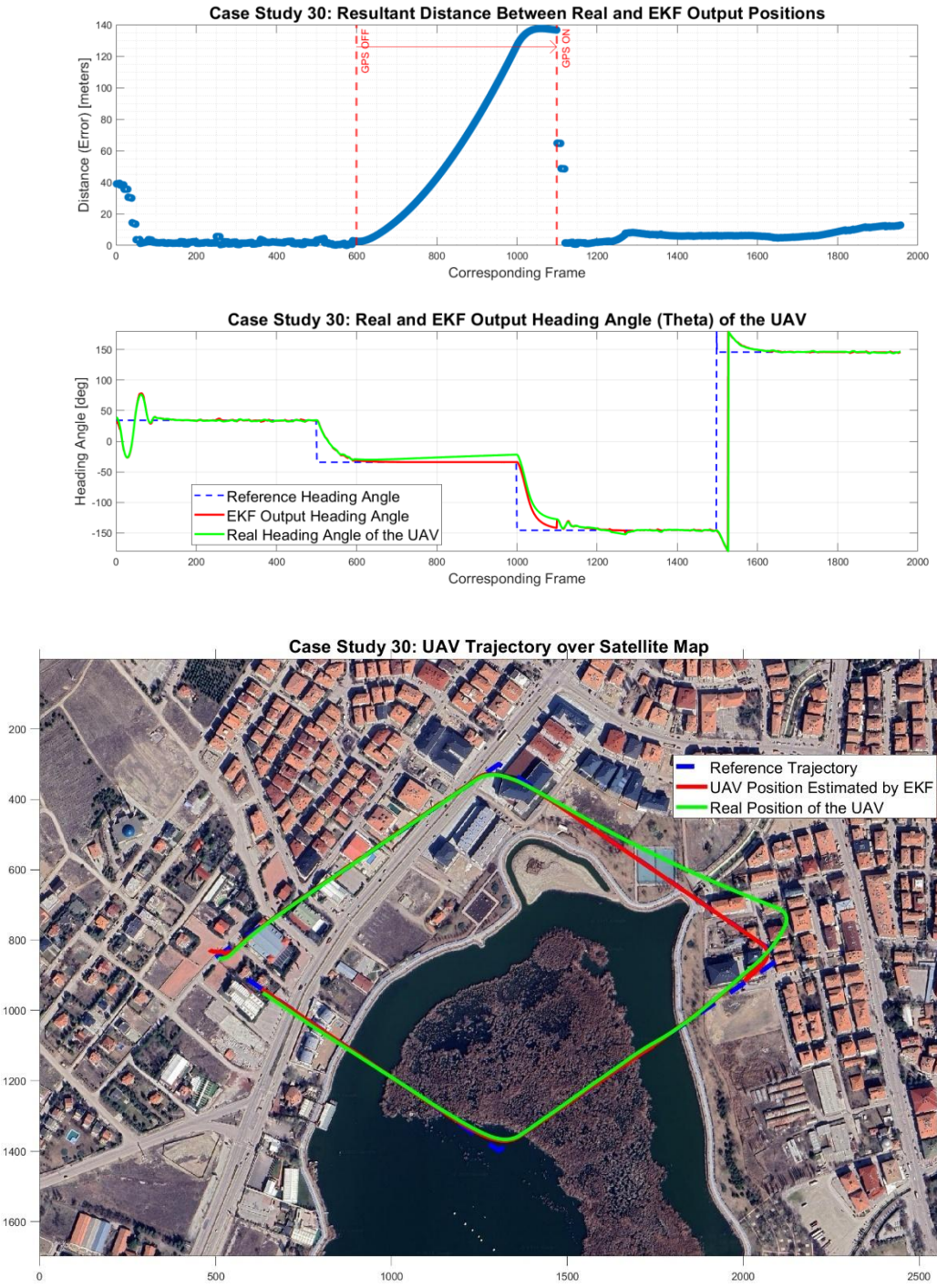

Figure 4.21. Case study 30 (MPC mode Linear, Scenario 4 -middle section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

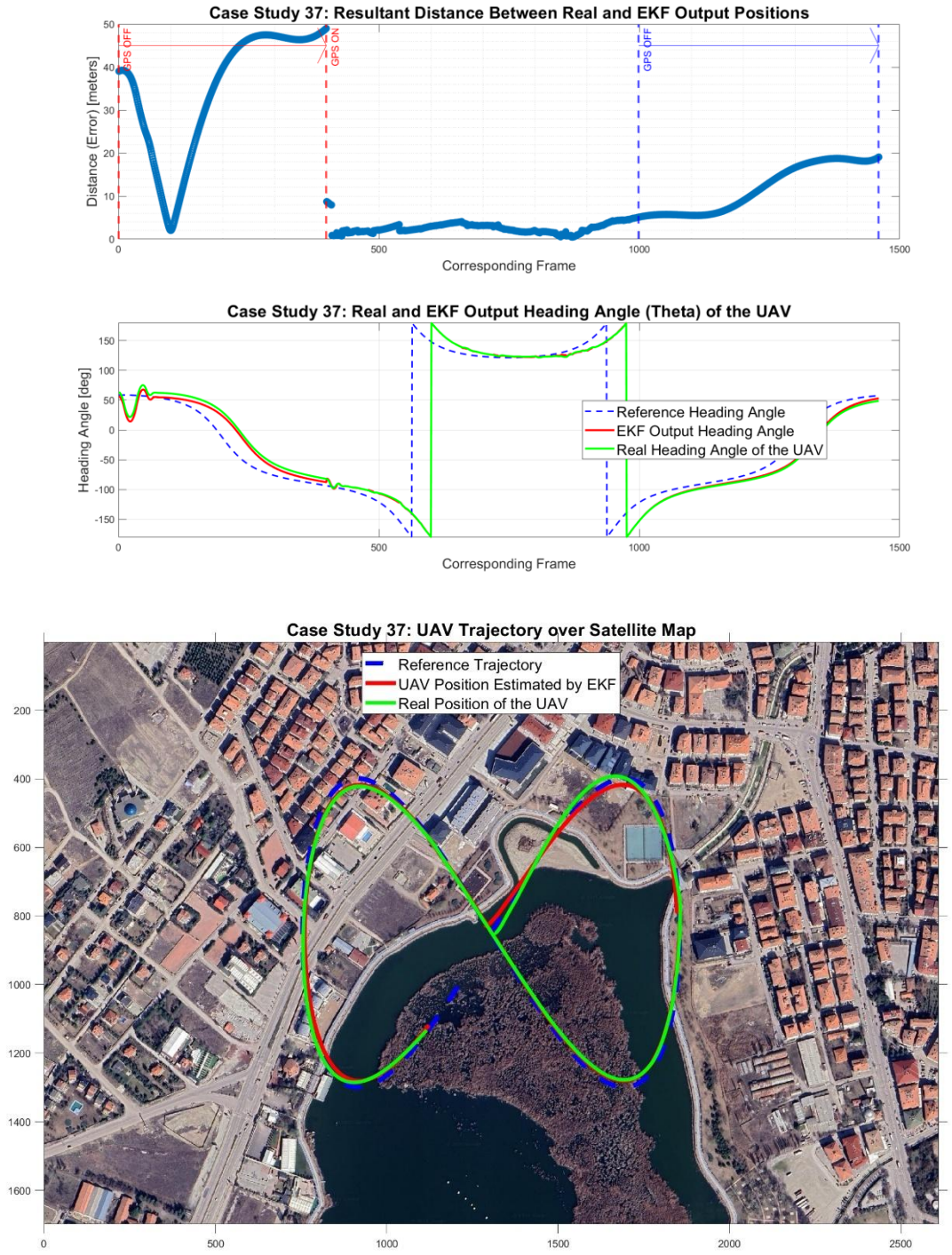

Figure 4.22. Case study 37 (MPC mode Linear, Scenario 5 -first and last section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

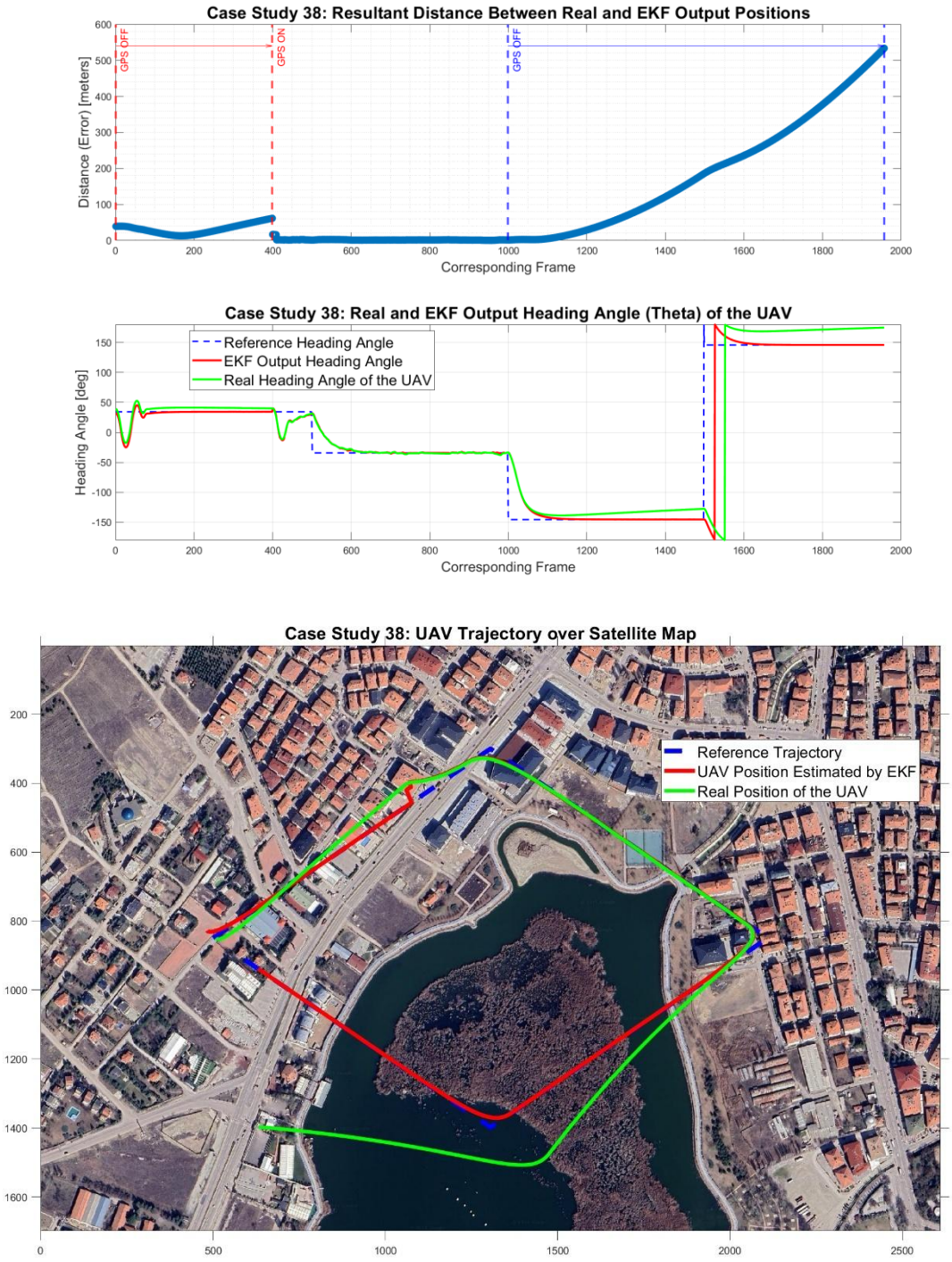

Figure 4.23. Case study 38 (MPC mode Linear, Scenario 5 -first and last section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

5. DISCUSSION AND CONCLUSION

This thesis aimed to handle a major problem for modern Unmanned Aerial Vehicles (UAVs): their extensive use of the Global Positioning System (GPS) for navigation. The goal was to design and test a complete navigation system with the ability to provide accurate global position information where GPS is not available. To achieve this, in the three broad areas are combined: computer vision, sensor fusion, and optimal control. The system is based on a "Virtual GPS" idea, using publicly available satellite imagery from services like Google Maps as a map. In this section, the main outcomes of the simulations are given, the contributions of the study are discussed, the limitations of the research are stated, and future research possibilities are suggested.

5.1. Discussions for Key Findings

The system's performance was validated through many simulations. The experiments were designed to validate each main subsystem of the system: the visual positioning system, the sensor fusion filter, and the flight controller. The experiments were conducted in normal conditions and also in difficult conditions, like losing the visual signal.

5.1.1. Performance of the Visual Localization (Virtual GPS) Module

The most critical element of the navigation system is how it is able to find the global position of the UAV by matching the camera view to a satellite map. The experiments in Section 4 showed that the method holds up well as a standalone source for positions. The system used the SURF detector to find features and RANSAC to remove low-quality matches. In areas with adequate visual information, the system accurately estimated the position.

28 random positions were tested in the first experiment. 21 out of 28 tests had error values above the thresholds, with a final position error (RMSE) of 1.60 meters and a heading error of 0.59 degrees. Other tests on long flight paths (like an 8-shape and a circle) were also satisfactory. The system was consistently at meter-level precision. It was also important to examine where the mistakes are made. As one can see on the map, the biggest mistakes were made over areas of thin visual texture, like over the lake or large open fields. This is a known limitation of computer vision algorithms that use features. It shows

how essential it is to have a reliable sensor fusion system to handle situations where the vision data are unreliable.

5.1.2. Visual-Inertial Fusion Framework

The system combines two types of sensors with an Extended Kalman Filter (EKF). The first sensor is the Virtual GPS, which gives good position updates at low frequency (5 Hz) but does not drift with time. The second sensor is the Inertial Navigation System (INS), which gives continuous motion data at high frequency (50 Hz) but drifts with time. The fusion of these two sensors was found to be highly successful. The EKF used a kinematic model, which had the IMU acceleration measurements as direct inputs to predict the next state of the UAV. This is highly effective to fill the time gaps between the visual updates and gave a smooth position estimate for the controller.

The best evidence of success of the EKF was in experiments when the visual signal is lost. At such moments, the system was using the INS data only, and the position error grew with time as should have been expected. When the visual signal reappeared, the EKF could recover both very quickly and smoothly. This success was due to the application of special adaptive methods in the filter. They kept the filter from making a sudden jump and allowed it to come back on the desired path softly. This suggests that the system is stable and able to respond to the transition of having a visual signal and losing it, which is one of the primary goals of this work.

5.1.3. Comparative Analysis of NMPC Controller Formulations

The research tested four different methods for the Nonlinear Model Predictive Control (NMPC) controller. The methods were tested on different flight paths and with different visual signal loss conditions. The 40 case studies showed that there was no single method that was better than all others. Different strengths of each method existed in terms of tracking accuracy, smoothness of flight, and computation time.

For example, the Switching (SW) controller, which initially corrects heading then moves forward, was good in preventing or recovering large errors but sometimes caused jerky motion. On the other hand, Linear Path and Exponential Path methods produced smoother flights because they optimized position and heading simultaneously. The adaptive part of

the controller, that tuned the control weights based on the size of the error, was crucial to all the techniques. It helped the controller to be soft for small errors and more aggressive for larger errors. This means that NMPC, if applied with an appropriate state estimator, is a very powerful tool for autonomous flight.

5.2. Research Contributions

This thesis provides several important contributions to the field of autonomous UAV navigation:

- ✓ Integrated System Demonstration: The main contribution is the design and complete simulation of a whole navigation and control system. It successfully combines a new Virtual GPS concept with a finely tuned EKF and an advanced NMPC controller.
- ✓ Validation of Public Satellite Imagery: The study shows that publicly accessible satellite maps can be used as a reliable reference for meter-scale UAV positioning. This means special or expensive maps are not required.
- ✓ Solid EKF Design: An EKF based on kinematic properties that has novel adaptive properties is designed. The smooth recoverability of the filter from loss of vision data is a significant practical contribution.
- ✓ Comparative Study of NMPC Strategies: The thesis presents a valuable comparison of different NMPC strategies for tracking a trajectory. This helps in comprehending the trade-offs involved in designing autonomous navigation controllers.

5.3. Limitations of the Study

In this study there are limitations mentioned below that is unrealistic but making the work simpler.

- ✓ Simulation-Based Environment: The entire system was simulated. The real world is more challenging with issues like hardware delay, different types of sensor noises, and the limited capability of an onboard computer to do calculations.
- ✓ Constant Altitude Assumption: The simulation had been optimized for constant altitude 2D flight. Real missions require full 3D control to cope with height, roll, and pitch motion.

- ✓ Idealized Visual Conditions: Since the simulation is not using captured real images, comparison algorithms that is used for positioning work more accurately. Moreover, it did not take real visual problems into account, such as sudden changes of light (e.g., shadows), seasonal conditions (e.g., snow), or blur due to rapid motion.
- ✓ Over-Reliance on Hand-Crafted Features: The system uses the SURF algorithm. While good, more recent deep learning methods for feature detection may fare better in adverse vision scenarios.

5.4. Future Work and Recommendations

Based on this study, several topics for future research are suggested:

- ✓ Advanced Visual Matching Algorithms: Before proceeding with real time applications, as the most important future work, Virtual GPS can be improved by employing modern deep learning-based feature matching algorithms like SuperPoint and SuperGlue as referred in Section 2.4.
- ✓ Extension to a 6-DOF System: The controller and the model need to extend to a full 6-DOF system to manage all of the UAV's flight dimensions, including altitude.
- ✓ Hardware Implementation and Real-World Testing: The second most important thing to do is put this system on an actual UAV and test it in real life. This will show how well the system performs outside of simulation.
- ✓ Long-term Robustness: Future design must address the issue of how to make the system work in long missions, such as how to handle an evolving satellite map over time.
- ✓ Mapping (2 Camera Option): Another work can be performed as the next step is to meet requirements for mapping. Like most VSLAM applications, the improved future system can create maps during UAV’s flight.

5.5. Conclusion

This thesis has shown that it is possible to design an integrated visual-inertial navigation system for UAV GPS-denied flight. The system combines a new Virtual GPS, a rich-featured EKF, and an NMPC controller to provide the complete solution for position estimation and flight control. The simulation results have shown that the system can

tolerate the short/long-term loss of visual measurements and still stabilize the aircraft. Although the research was conducted in simulation and is not perfect, it is a huge step towards making usable technology that can free UAVs from their dependence on GPS. The system presented here is a solid basis to future work and to make fully autonomous air vehicles that can safely fly in any environment.

6. REFERENCES

Aerosystems West. (2024). ASW heavy lift octocopter industrial drone specifications. Aerosystems West Product Line. https://aerosystemswest.com/product/asw-heavy-lift-octocopter-industrial-drone/

Allan, D. W. (1966). Statistics of atomic frequency standards. Proceedings of the IEEE, 54(2), 221-230.

ArduPilot Development Team. (2024). The Cube Orange overview - Copter documentation. ArduPilot Documentation. https://ardupilot.org/copter/docs/common-thecubeorange-overview.html

Bar-Shalom, Y., Li, X. R., & Kirubarajan, T. (2001). Estimation with applications to tracking and navigation: Theory algorithms and software. John Wiley & Sons.

Barbour, N., & Schmidt, G. (2001). Inertial sensor technology trends. IEEE Sensors Journal, 1(4), 332-339.

Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. (2008). Speeded-up robust features (SURF). Computer Vision and Image Understanding, 110(3), 346-359.

Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., & Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. IEEE Transactions on Robotics, 32(6), 1309-1332.

CubePilot. (2024). Cube Orange flight controller specifications. PX4 Autopilot Documentation. https://docs.px4.io/main/en/flight_controller/cubepilot_cube_orange.html

CubePilot. (2024). The Cube module overview - Technical specifications. CubePilot Documentation. https://docs.cubepilot.org/user-guides/autopilot/the-cube-module-overview

Davison, A. J., Reid, I. D., Molton, N. D., & Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(6), 1052-1067.

Dellaert, F., & Kaess, M. (2017). Factor graphs for robot perception. Foundations and Trends in Robotics, 6(1-2), 1-139.

DeTone, D., Malisiewicz, T., & Rabinovich, A. (2018). SuperPoint: Self-supervised interest point detection and description. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 224-236).

Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. IEEE Robotics & Automation Magazine, 13(2), 99-110.

DJI. (2024). A3 Pro flight controller specifications. DJI Official Documentation. https://dl.djicdn.com/downloads/a3/en/A3_and_A3_Pro_User_Manual_en_160520.pdf

DJI. (2024). Matrice 600 Pro user manual and specifications (v1.0). DJI Downloads. https://dl.djicdn.com/downloads/m600%20pro/20180417/Matrice_600_Pro_User_Manual_v1.0_EN.pdf

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6), 381-395.

Foxtech. (2024). THEA 200MP heavy lift coaxial octocopter specifications. Foxtech Drone Store. https://www.foxtechfpv.com/thea-200mp-heavy-lift-coaxial-octocopter.html

Freefly Systems. (2024). Alta X heavy lift drone specifications. Freefly Systems Official Website. https://freeflysystems.com/alta-x

Galvez-Lopez, D., & Tardos, J. D. (2012). Bags of binary words for fast place recognition in image sequences. IEEE Transactions on Robotics, 28(5), 1188-1197.

GetFPV. (2024). CubePilot The Cube Orange+ (IMU V8) specifications. GetFPV Product Database. https://www.getfpv.com/cubepilot-the-cube-orange-imu-v8.html

Groves, P. D. (2013). Principles of GNSS, inertial, and multisensor integrated navigation systems. Artech House.

Hassanalian, M., & Abdelkefi, A. (2017). Classifications, applications, and design challenges of drones: A review. Progress in Aerospace Sciences, 91, 99-131. https://doi.org/10.1016/j.paerosci.2017.04.003

Hofmann-Wellenhof, B., Lichtenegger, H., & Wasle, E. (2008). GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more. Springer Science & Business Media.

JOUAV. (2025, May 21). The ultimate guide to heavy lift drone motors. JOUAV Blog. https://www.jouav.com/blog/heavy-lift-drone-motors.html

Julier, S. J., & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In Signal processing, sensor fusion, and target recognition VI (Vol. 3068, pp. 182-193). SPIE.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. Journal of Basic Engineering, 82(1), 35-45.

Kaplan, E. D., & Hegarty, C. (2017). Understanding GPS/GNSS: Principles and applications. Artech House.

Kendall, A., Grimes, M., & Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In Proceedings of the IEEE International Conference on Computer Vision (pp. 2938-2946).

Klein, G., & Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In Proceedings of the 6th IEEE and ACM International Symposium on Mixed and Augmented Reality (pp. 225-234).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097-1105.

Li, M., & Mourikis, A. I. (2013). High-precision, consistent EKF-based visual-inertial odometry. The International Journal of Robotics Research, 32(6), 690-711.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2), 91-110.

Luukkonen, T. (2011). Modelling and control of quadcopter. Independent research project in applied mathematics, Espoo, 22(22), 1-24.

NASA Ames Research Center. (2017). Estimation, navigation and control of multi-rotor drones in an urban environment (NASA/TM-2017-219672). NASA Technical Reports Server. https://ntrs.nasa.gov/api/citations/20170000461/downloads/20170000461.pdf

Mahalanobis, P. C. (1936). On the generalized distance in statistics. Proceedings of the National Institute of Sciences of India, 2(1), 49-55.

Mourikis, A. I., & Roumeliotis, S. I. (2007). A multi-state constraint Kalman filter for vision-aided inertial navigation. In Proceedings 2007 IEEE International Conference on Robotics and Automation (pp. 3565-3572).

Mouser Electronics. (2024). ICM-20948 TDK InvenSense motion sensor specifications. Mouser Electronics Catalog. https://www.mouser.com/ProductDetail/TDK-InvenSense/ICM-20948

Mur-Artal, R., Montiel, J. M. M., & Tardos, J. D. (2015). ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transactions on Robotics, 31(5), 1147-1163.

Osmic, N., Tahirovic, A., & Petrovic, I. (2020). Risk-sensitive motion planning for MAVs based on mission-related fault-tolerant analysis. Automatika, 61(2), 295-311. https://doi.org/10.1080/00051144.2020.1733324

Patel, B., Barfoot, T. D., & Schoellig, A. P. (2020). Visual localization with Google Earth images for robust global pose estimation of UAVs. IEEE Robotics and Automation Letters, 5(2), 3342-3349.

Psiaki, M. L., & Humphreys, T. E. (2016). GNSS spoofing and detection. Proceedings of the IEEE, 104(6), 1258-1270.

PX4 Development Team. (2024). PX4 autopilot user guide - CubePilot Cube Orange. PX4 Documentation. https://docs.px4.io/main/en/flight_controller/cubepilot_cube_orange.html

Ren, Y., Zhang, X., Ma, G., Cai, B., Liu, T., Lian, X., & Wang, Q. (2024). UAVs-based visual localization via attention-driven feature matching with multi-scale information. Drones, 8(12), 739.

RobotShop. (2024). TMotor UAV brushless motor U15XXL KV29 specifications. RobotShop Product Catalog. https://www.robotshop.com/products/tmotor-uav-brushless-motor-u15xxl-kv29

Sarlin, P. E., DeTone, D., Malisiewicz, T., & Rabinovich, A. (2020). SuperGlue: Learning feature matching with graph neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 4938-4947).

Savage, P. G. (2000). Strapdown analytics. Strapdown Associates.

Scaramuzza, D., & Fraundorfer, F. (2011). Visual odometry [tutorial]. IEEE Robotics & Automation Magazine, 18(4), 80-92.

Setati, T., Botha, N., & Roux, J. M. (2022). Experimental approach to calculate the moments of inertia of a hexacopter unmanned aerial vehicle. MATEC Web of Conferences, 370, Article 05001. https://doi.org/10.1051/matecconf/202237005001

Tarot. (2024). X8 heavy lift octocopter frame specifications (Model TL8X000). Foxtech FPV Store. https://www.foxtechfpv.com/tarot-x8-multicopterumbrella-type-folding-octocopter-with-retractable-landing-gear-p-1777.html

TDK InvenSense. (2016). ICM-20948 world's lowest power 9-axis MotionTracking device datasheet (Rev. 1.3). TDK InvenSense. https://invensense.tdk.com/wp-content/uploads/2016/06/DS-000189-ICM-20948-v1.3.pdf

TDK InvenSense. (2024). ICM-20948 product overview. TDK InvenSense Official Website. https://invensense.tdk.com/products/motion-tracking/9-axis/icm-20948/

Titterton, D., Weston, J. L., & Weston, J. (2004). Strapdown inertial navigation technology. IET.

Tyto Robotics. (2024, May 24). Big brushless motor manufacturers for drones and eVTOL. Tyto Robotics Blog. https://www.tytorobotics.com/blogs/articles/brushless-motor-manufacturers-for-evtol-and-aviation

T-Motor. (2024). E2000 propulsion system specifications. DJI Official Website. https://www.dji.com/e2000/info

T-Motor. (2024). U series motors - Heavy lift UAV propulsion systems. T-Motor UAV Division. https://uav-en.tmotor.com/Multirotor/Motors/u/

T-Motor. (2024). U15XXL KV29 98kg thrust heavy lift drone motor. T-Motor Official Store. https://shop.tmotor.com/products/u15xxl-kv29-98kg-thrust-heavy-lift-drone-motor

T-Motor. (2024). Thunder 300A 24S ESC specifications. T-Motor Store. https://store.tmotor.com/product/u15xxl-manned-aircraft-combo.html

Welch, G., & Bishop, G. (2006). An introduction to the Kalman filter. University of North Carolina at Chapel Hill.

Zhang, J., & Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. In Robotics: Science and Systems (Vol. 2, No. 9).

APPENDICIES

APPENDIX A Results for Case Studies

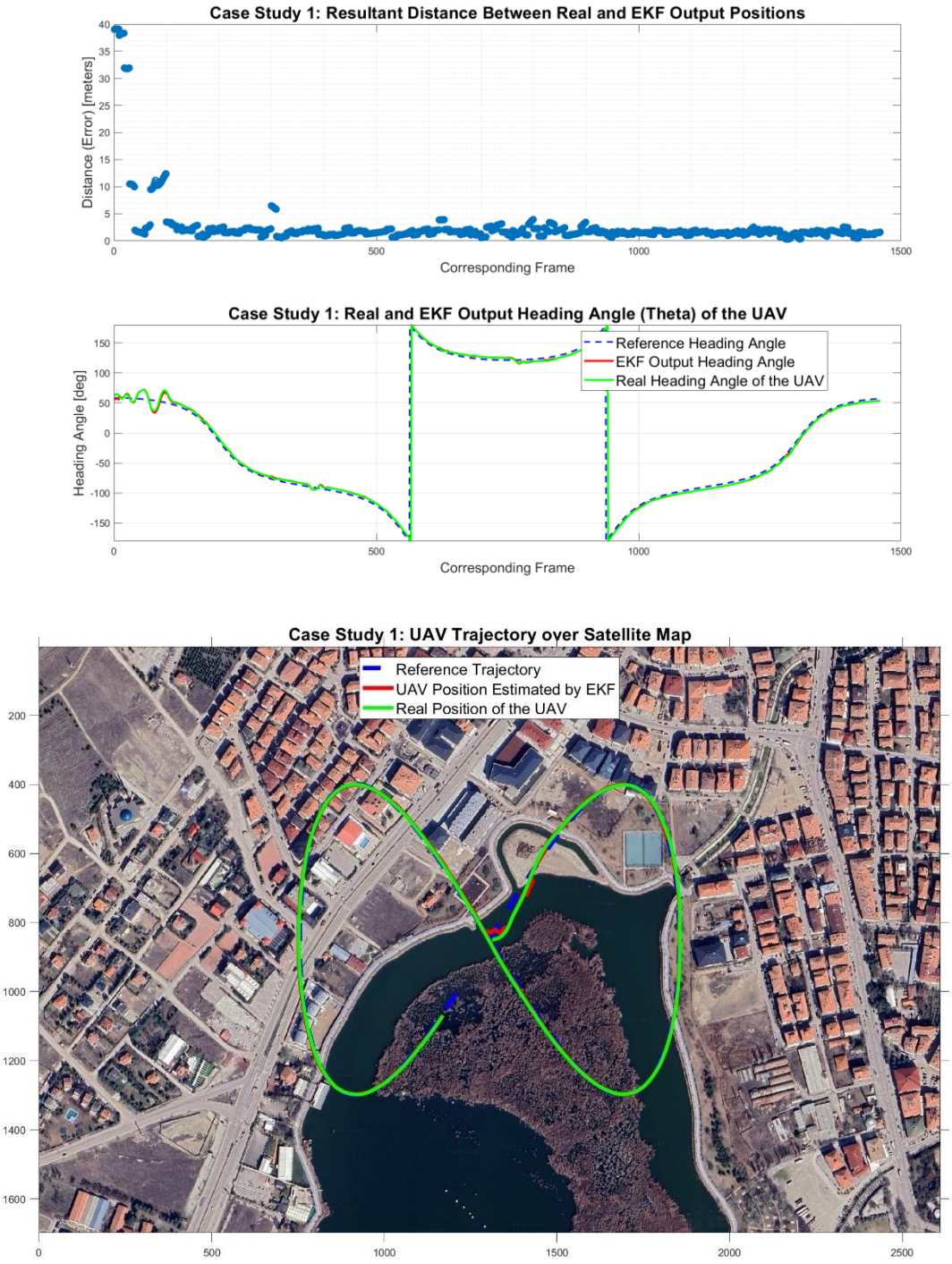

Figure A.1. Case study 1 (MPC mode Regulating, Scenario 1 -no vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

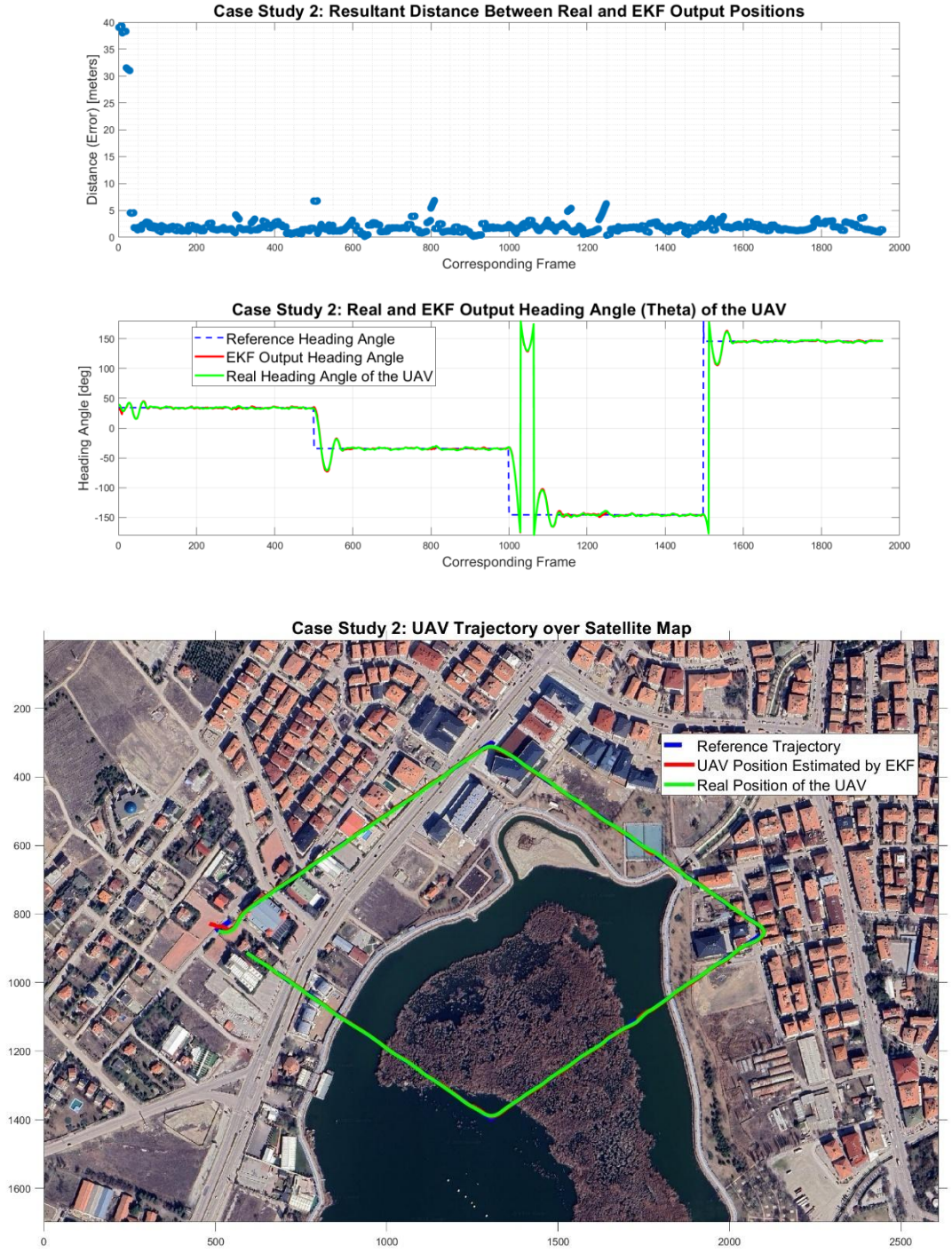

Figure A.2. Case study 2 (MPC mode Regulating, Scenario 1 -no vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

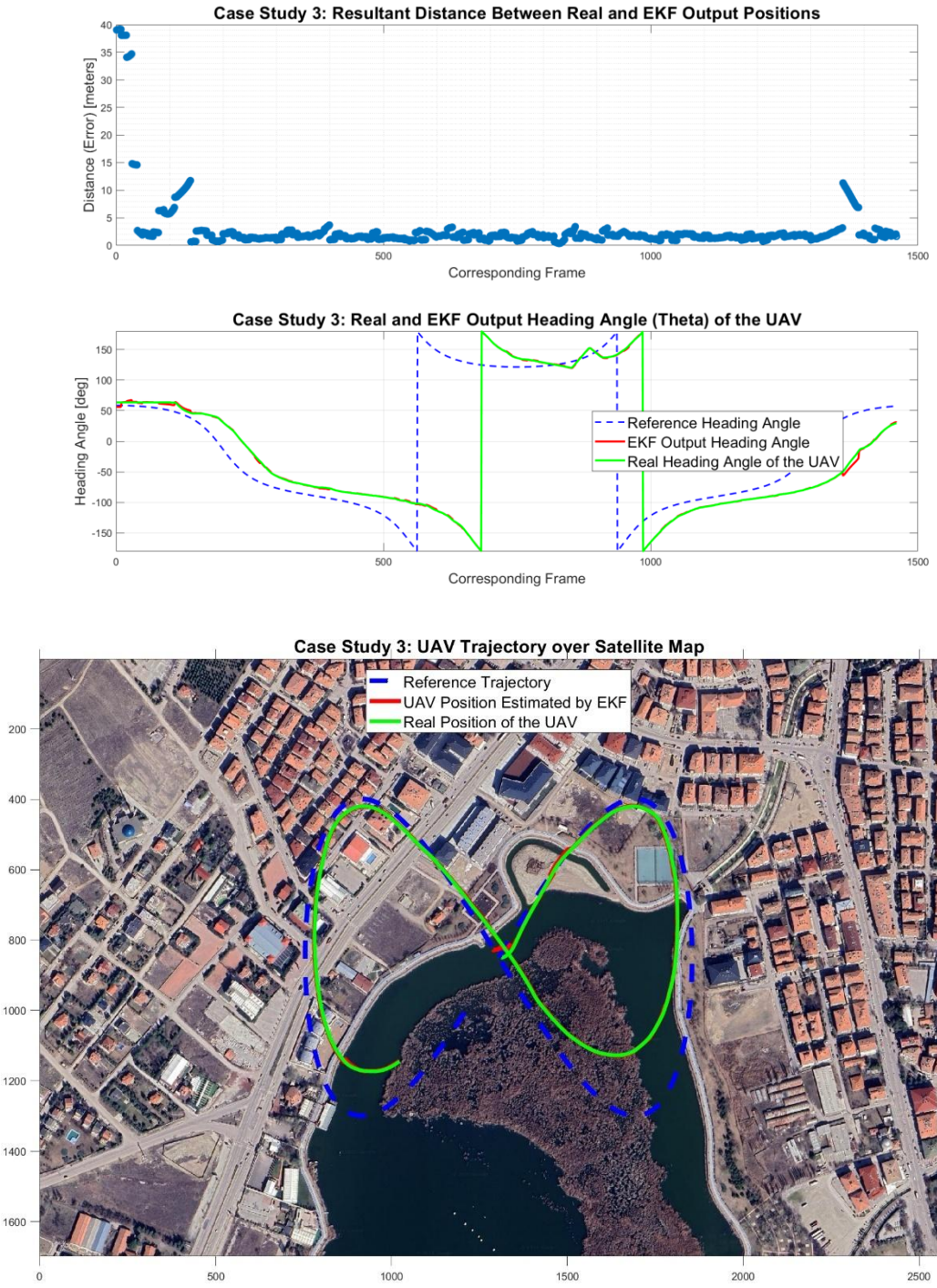

Figure A.3. Case study 3 (MPC mode Switching, Scenario 1 -no vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

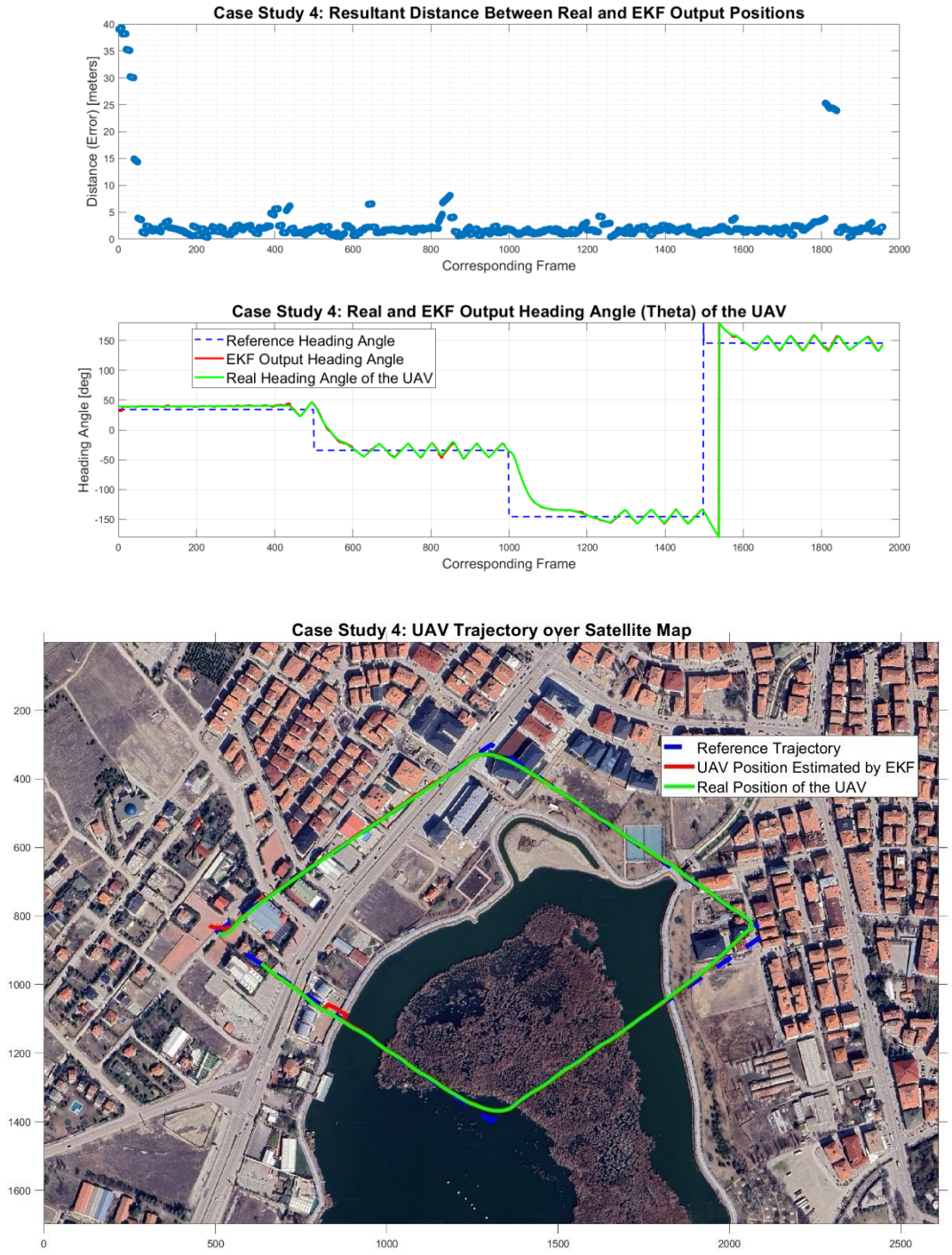

Figure A.4. Case study 4 (MPC mode Switching, Scenario 1 -no vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

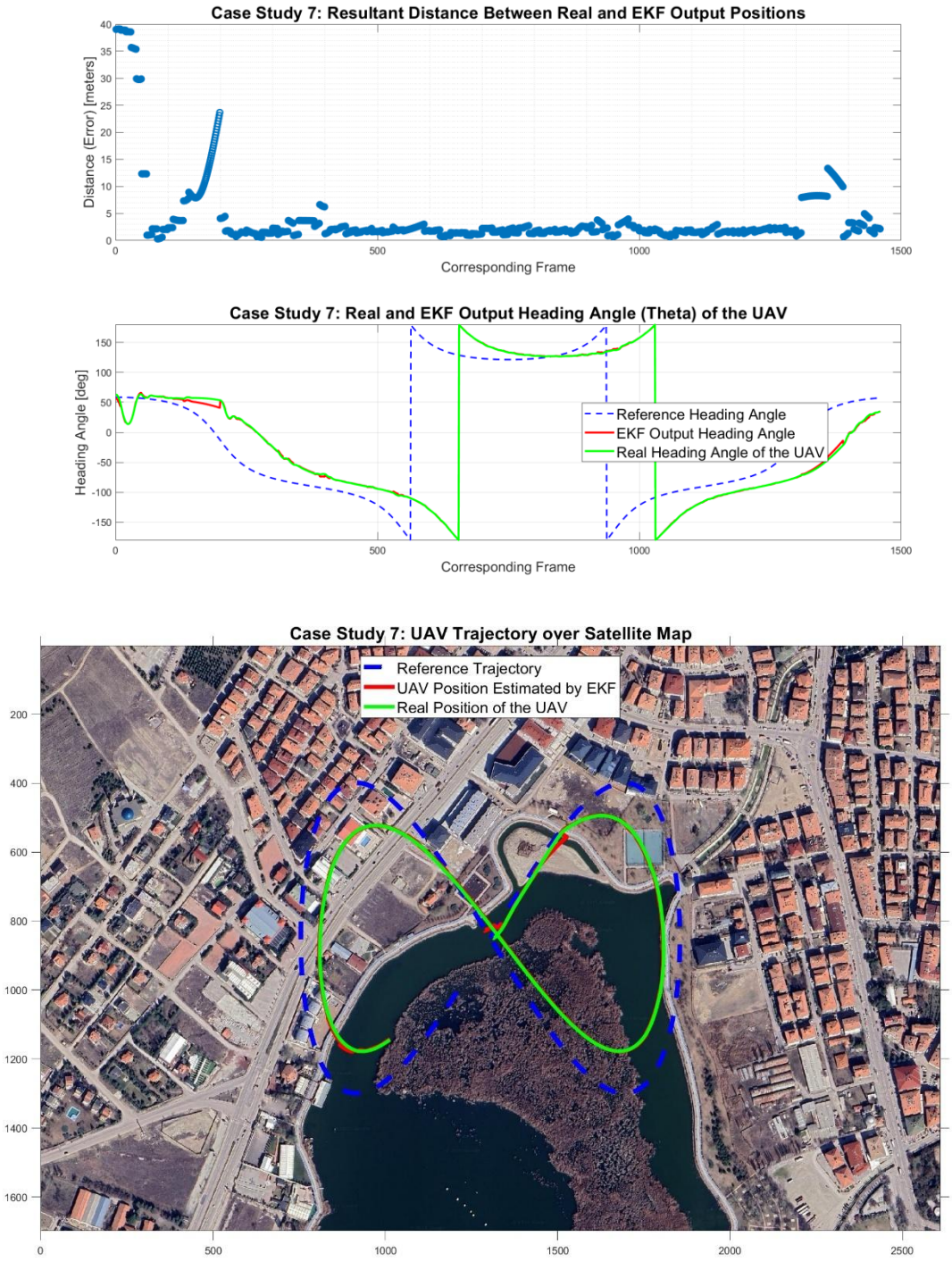

Figure A.5. Case study 7 (MPC mode Exponential, Scenario 1 -no vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

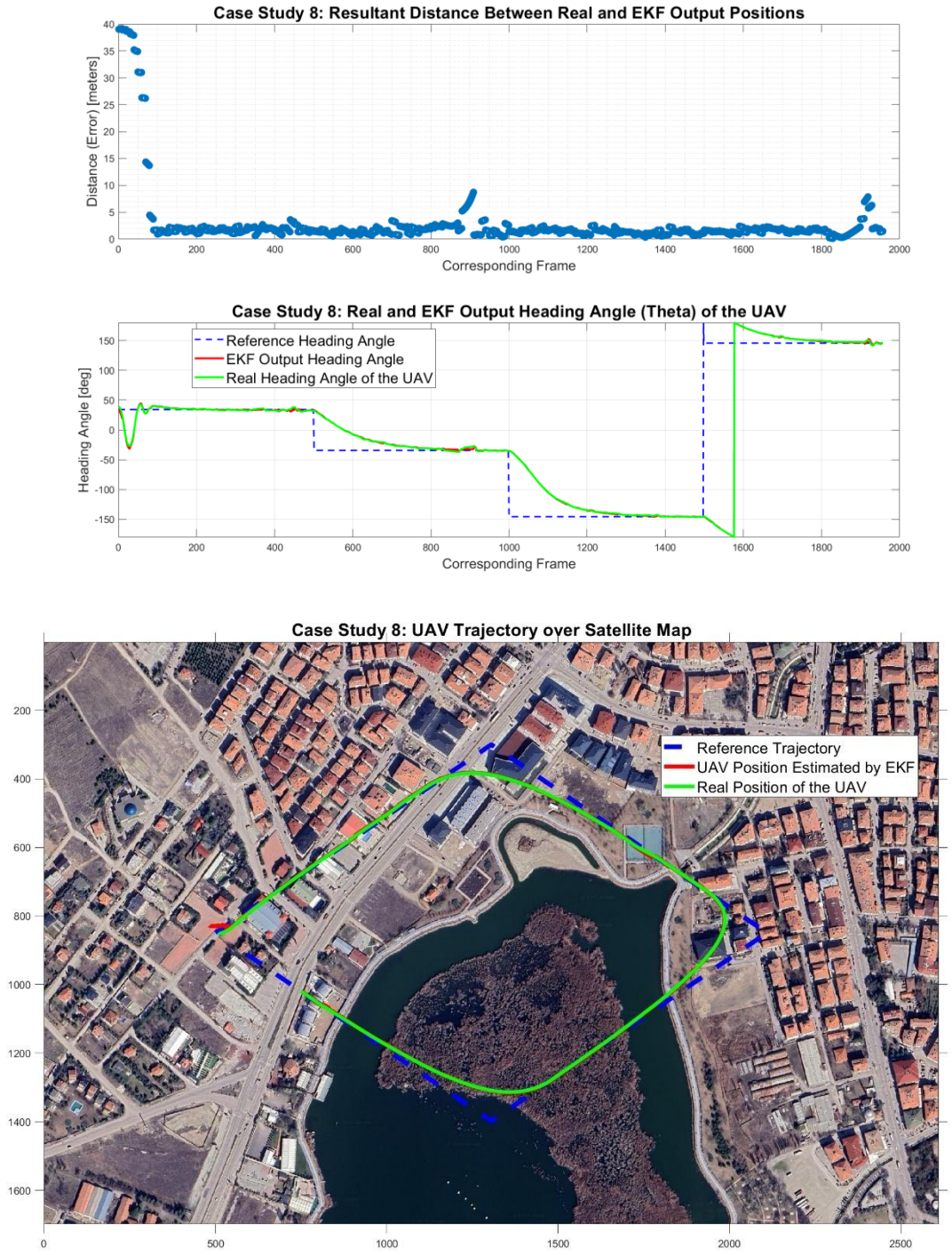

Figure A.6. Case study 8 (MPC mode Exponential, Scenario 1 -no vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

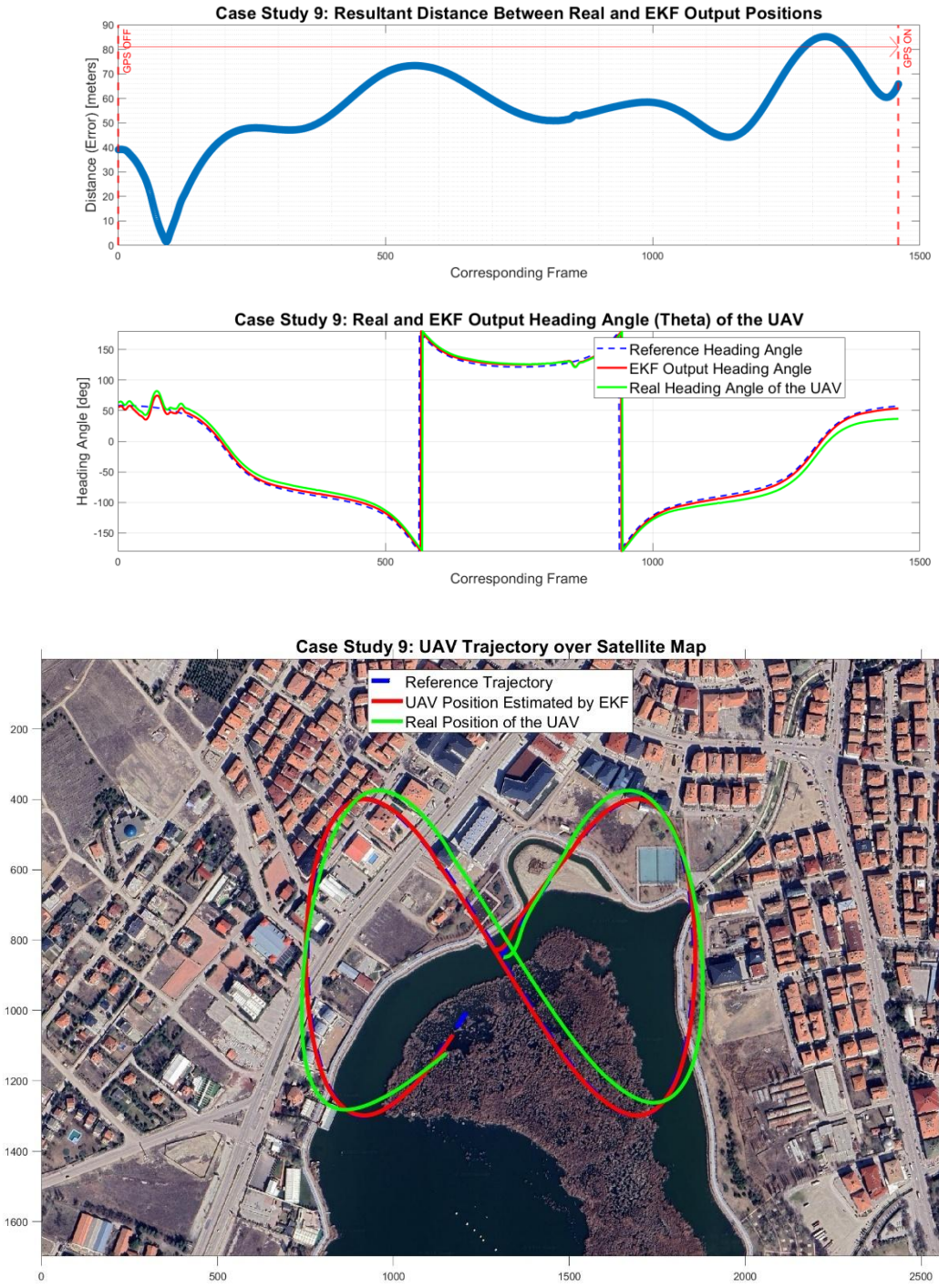

Figure A.7. Case study 9 (MPC mode Regulating, Scenario 2 -whole time vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

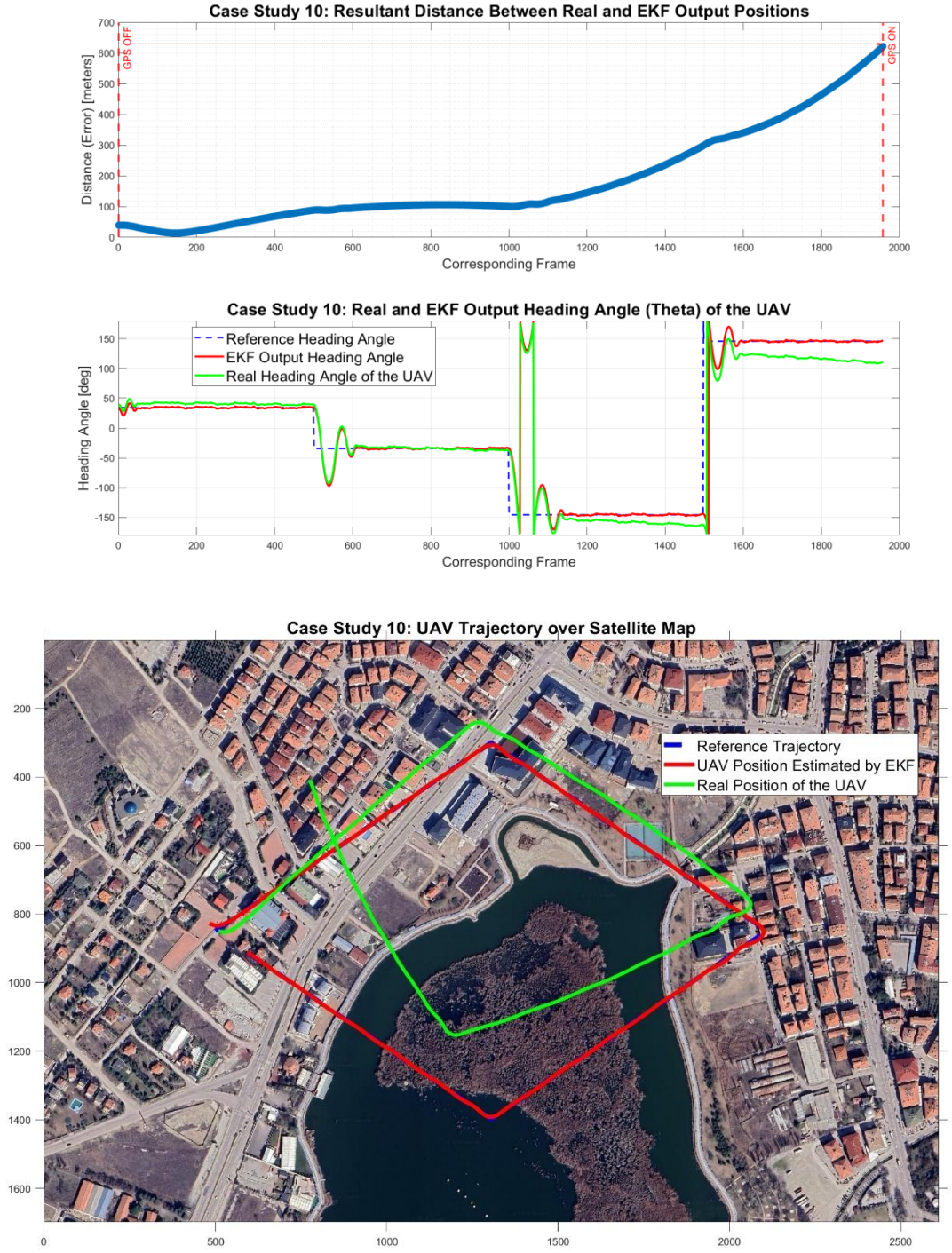

Figure A.8. Case study 10 (MPC mode Regulating, Scenario 2 -whole time vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

Figure A.9. Case study 11 (MPC mode Switching, Scenario 2 -whole time vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

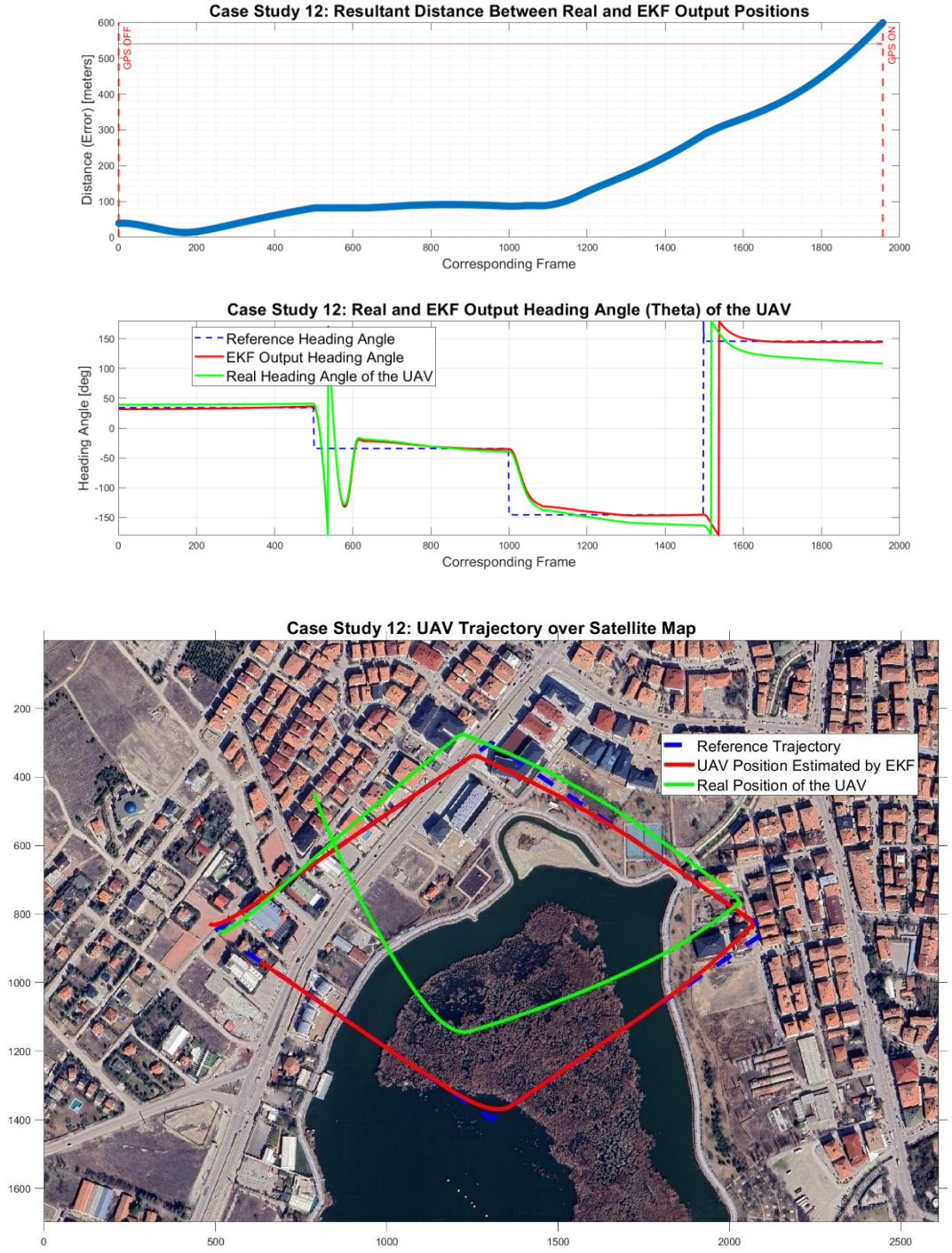

Figure A.10. Case study 12 (MPC mode Switching, Scenario 2 -whole time vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

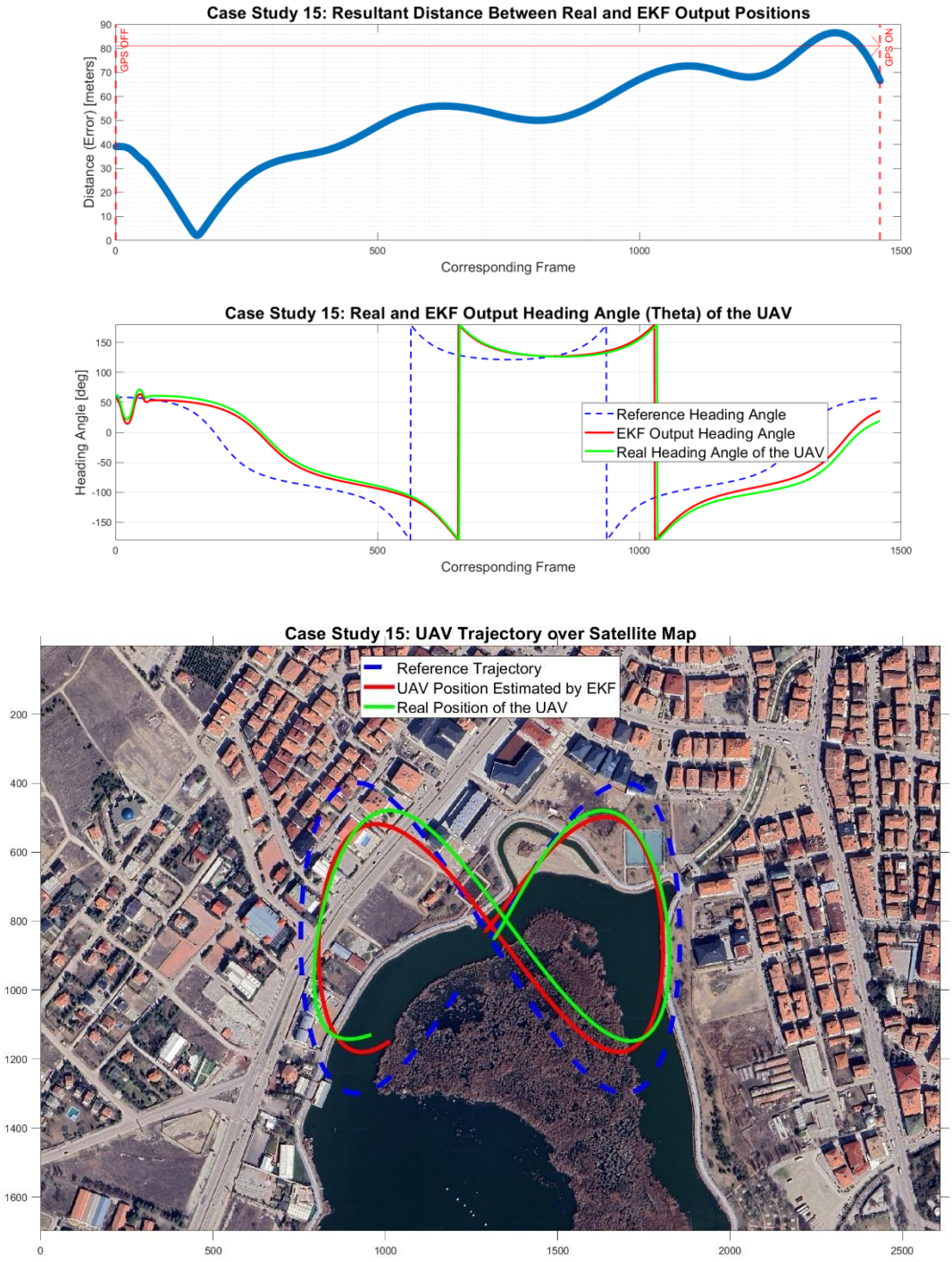

Figure A.11. Case study 15 (MPC mode Exponential, Scenario 2 -whole time vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

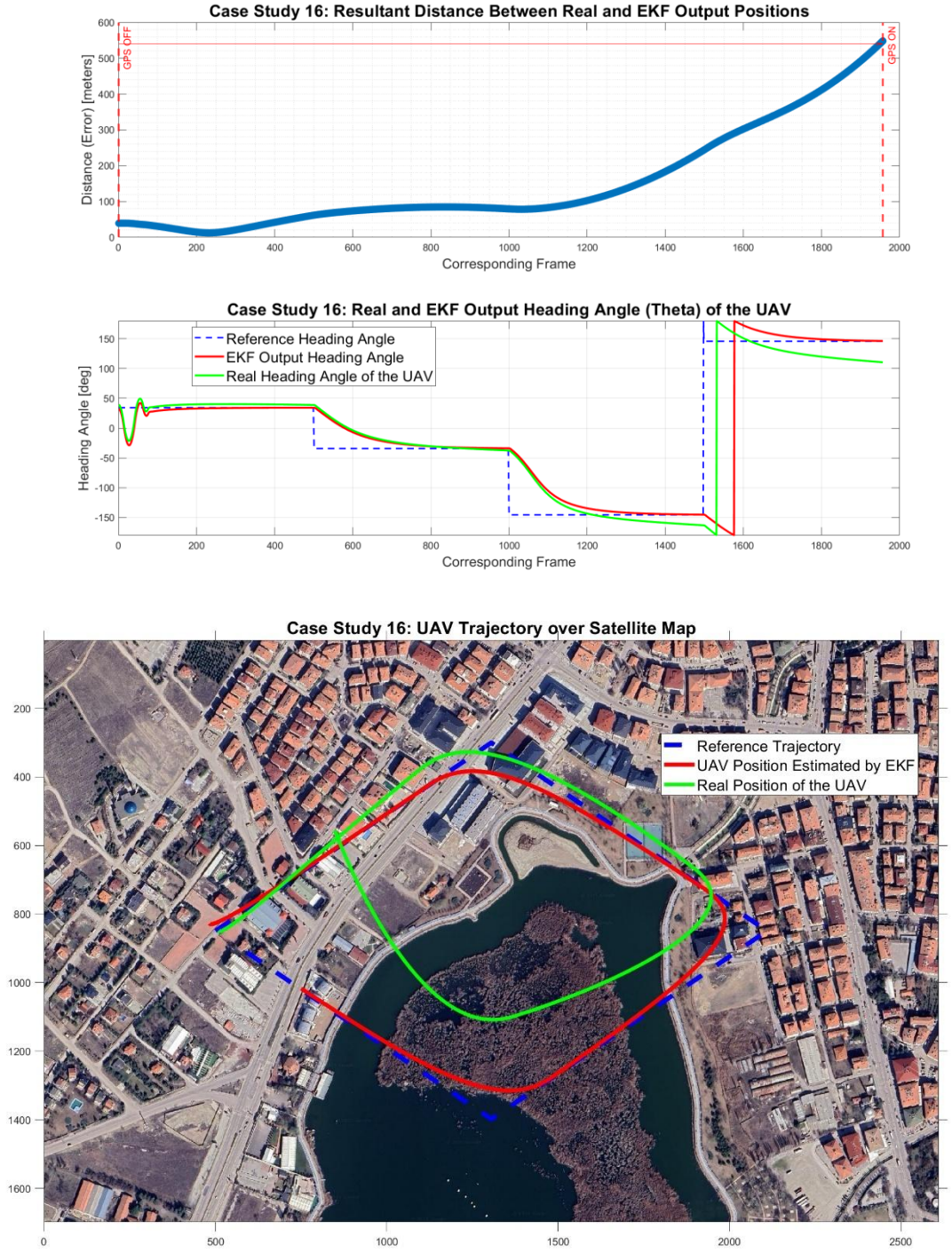

Figure A.12. Case study 16 (MPC mode Exponential, Scenario 2 -whole time vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

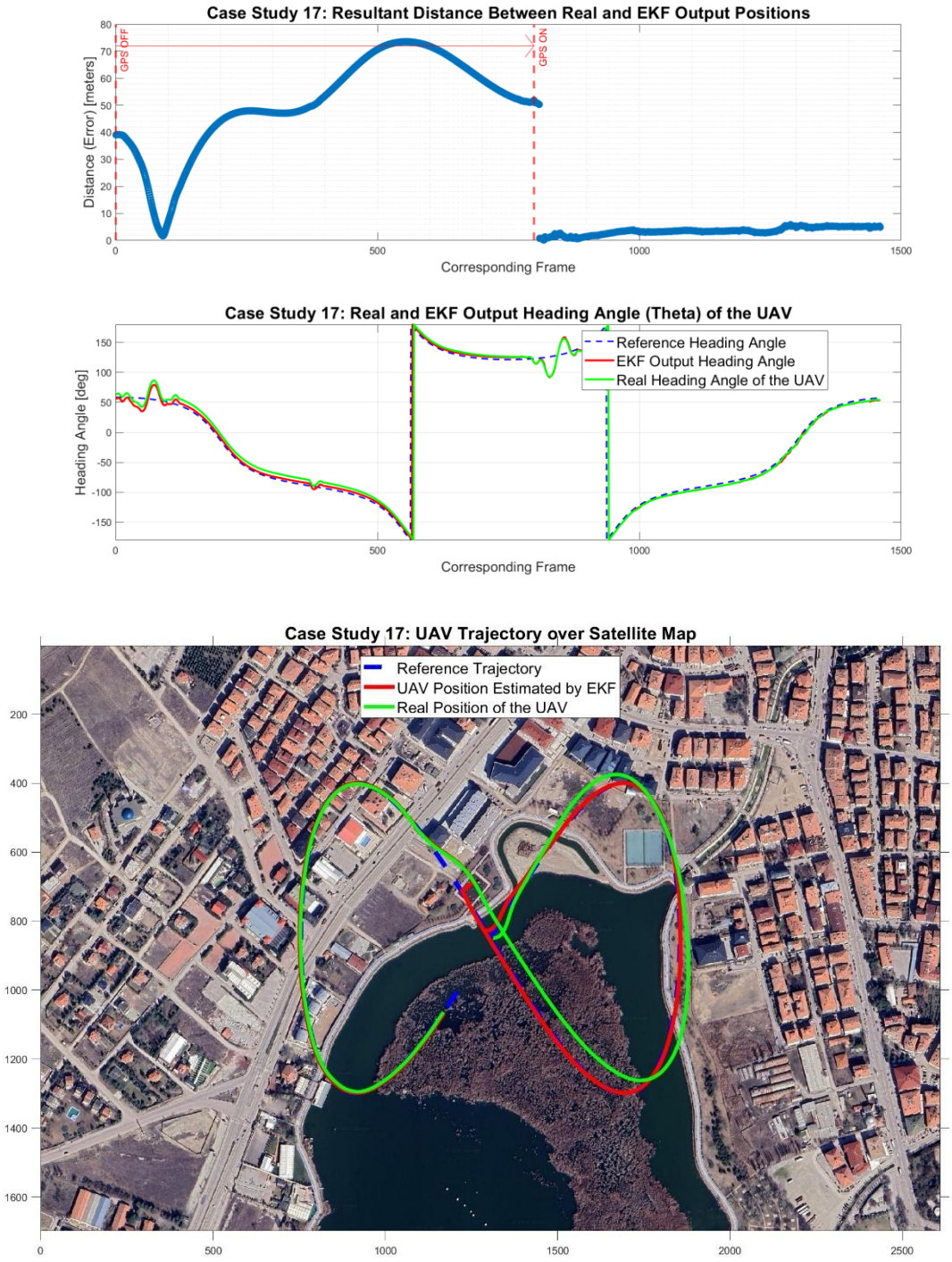

Figure A.13. Case study 17 (MPC mode Regulating, Scenario 3 -first section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

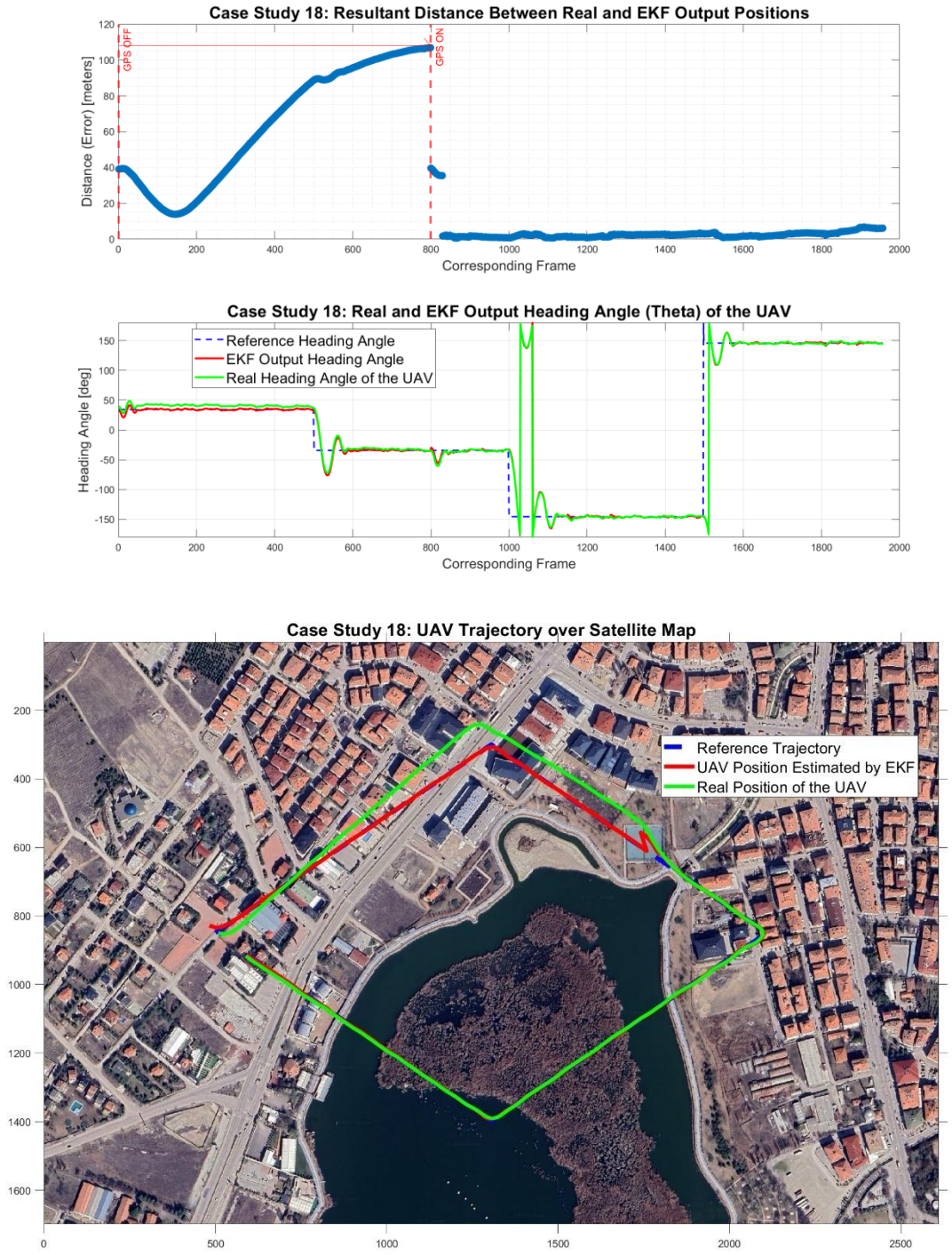

Figure A.14. Case study 18 (MPC mode Regulating, Scenario 3 -first section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

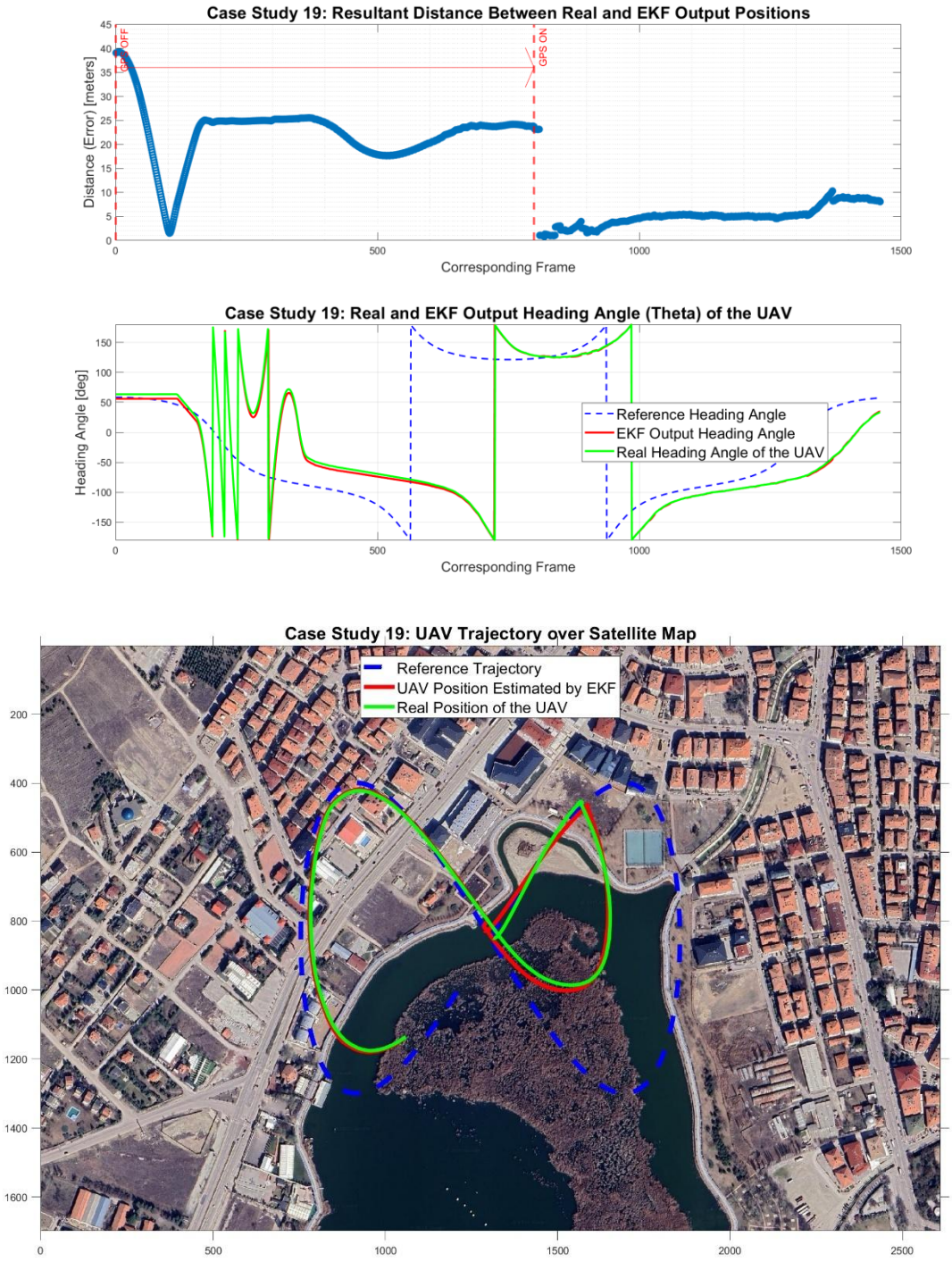

Figure A.15. Case study 19 (MPC mode Switching, Scenario 3 -first section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

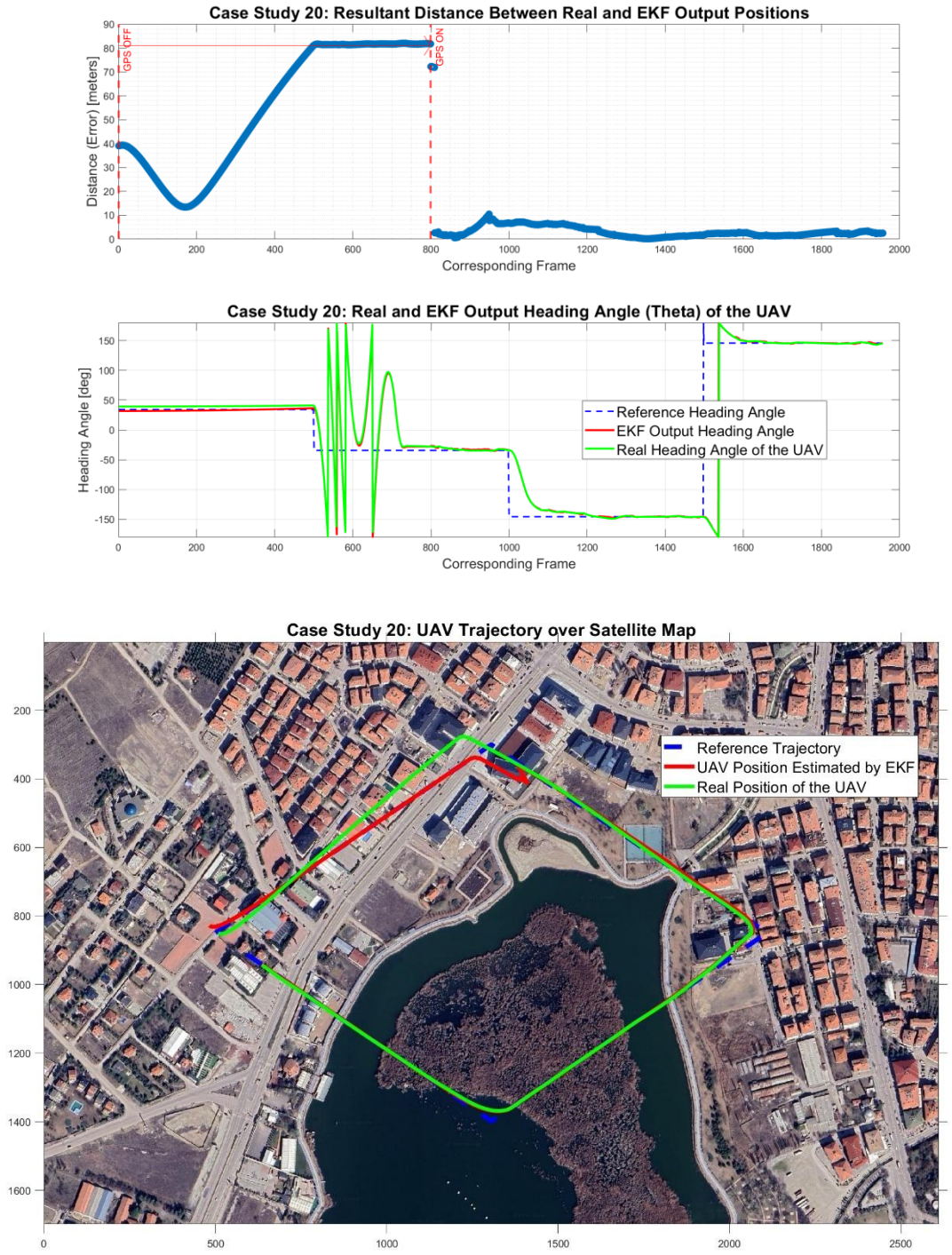

Figure A.16. Case study 20 (MPC mode Switching, Scenario 3 -first section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

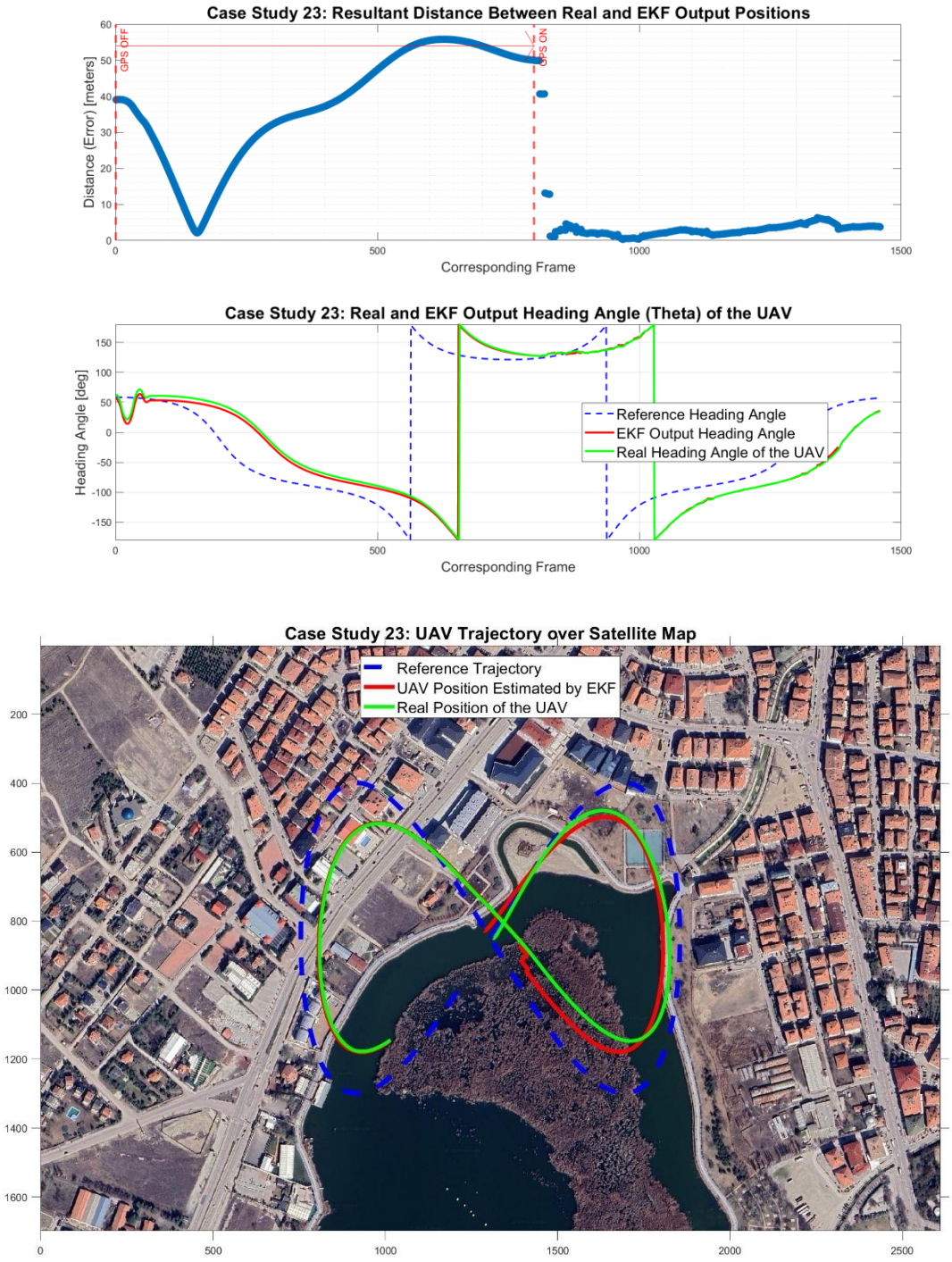

Figure A.17. Case study 23 (MPC mode Exponential, Scenario 3 -first section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

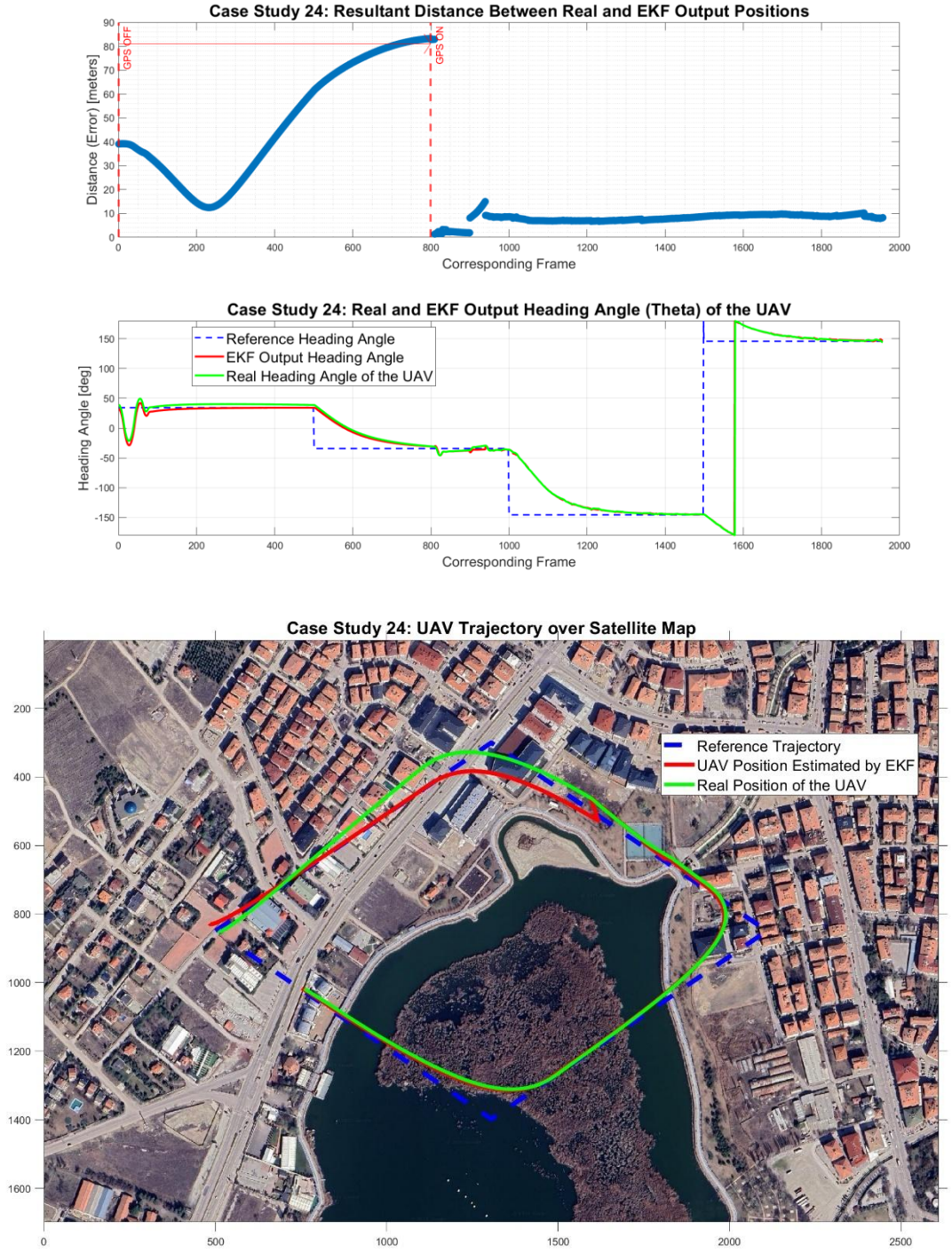

Figure A.18. Case study 24 (MPC mode Exponential, Scenario 3 -first section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

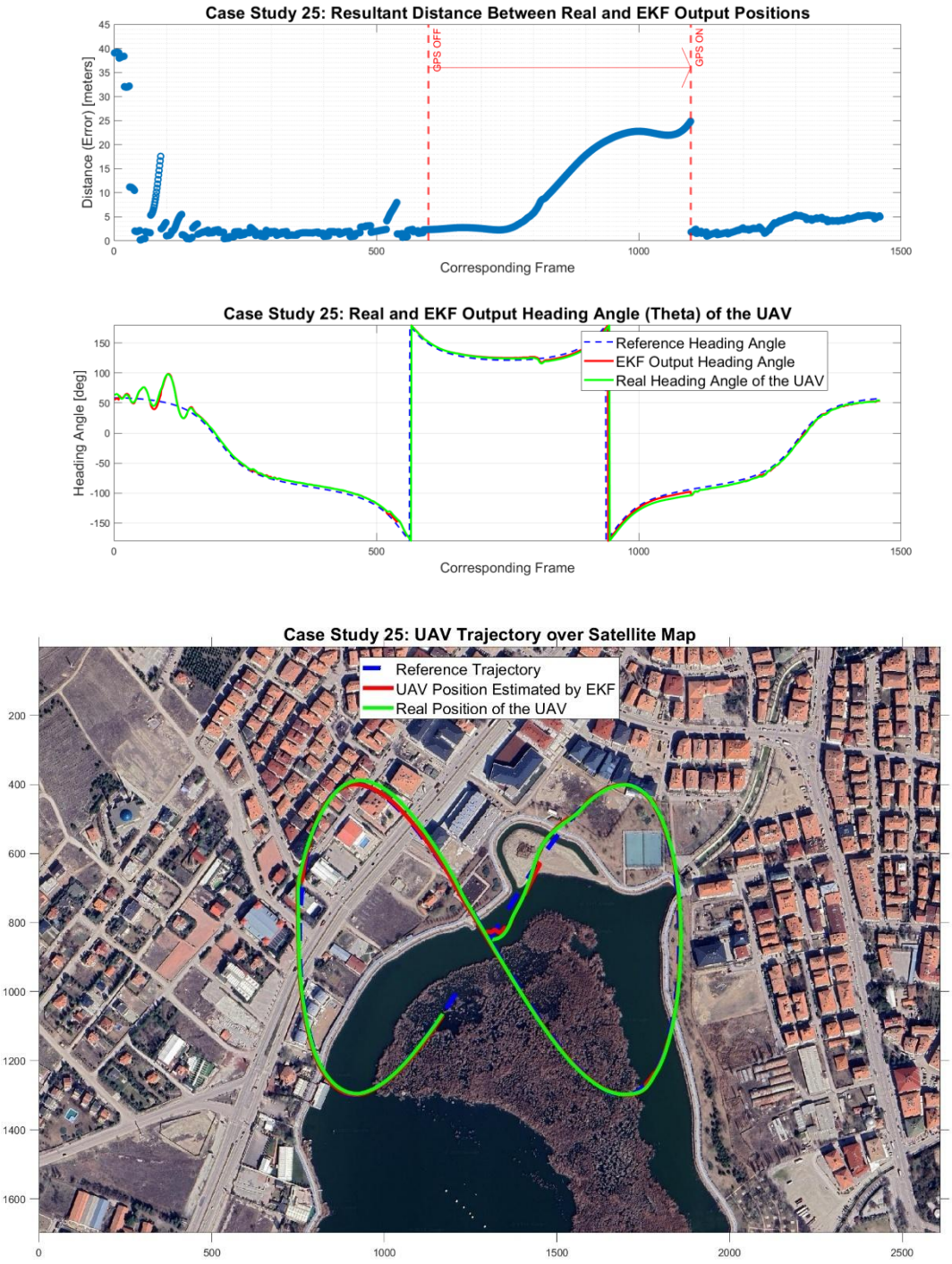

Figure A.19. Case study 25 (MPC mode Regulating, Scenario 4 -middle section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

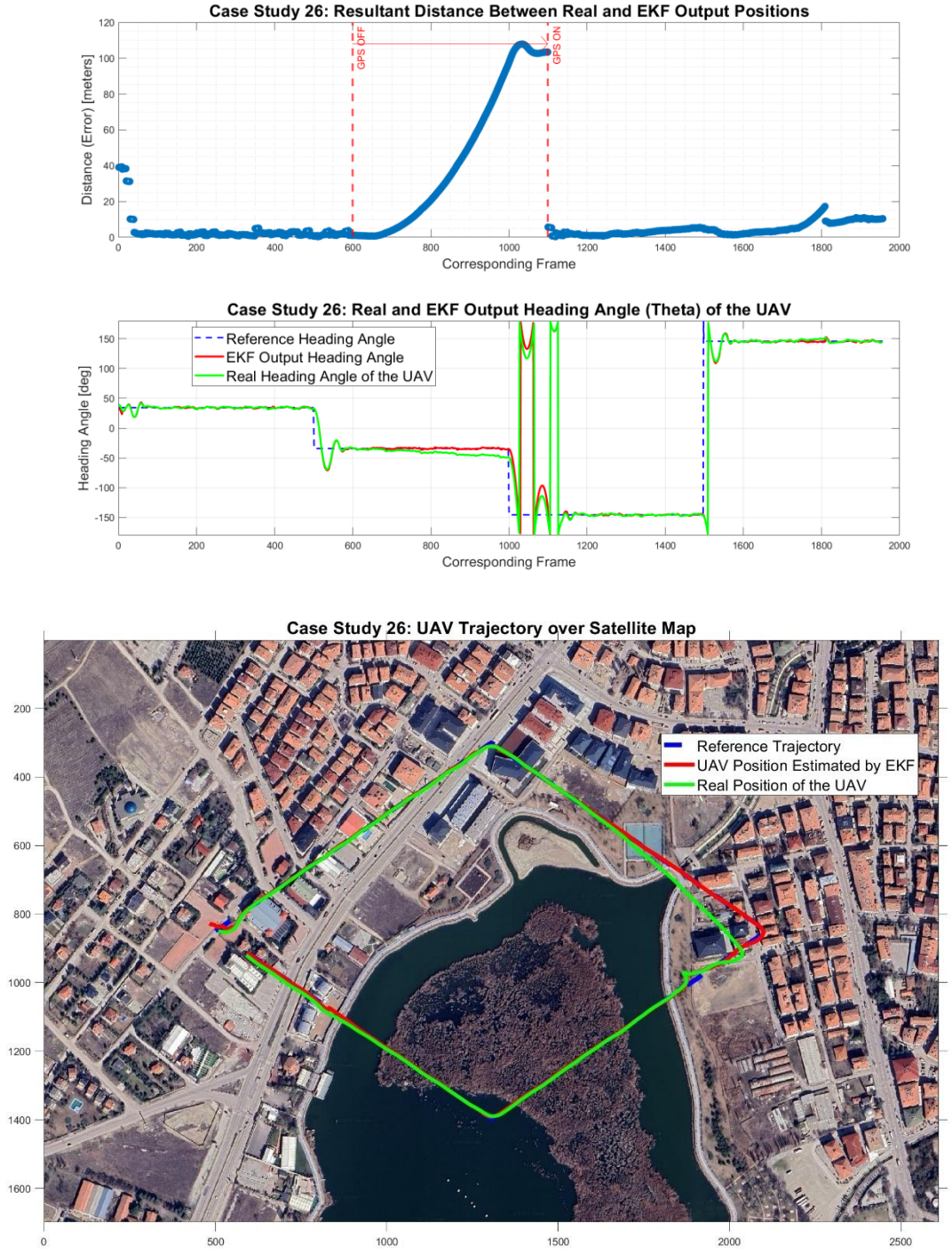

Figure A.20. Case study 26 (MPC mode Regulating, Scenario 4 -middle section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

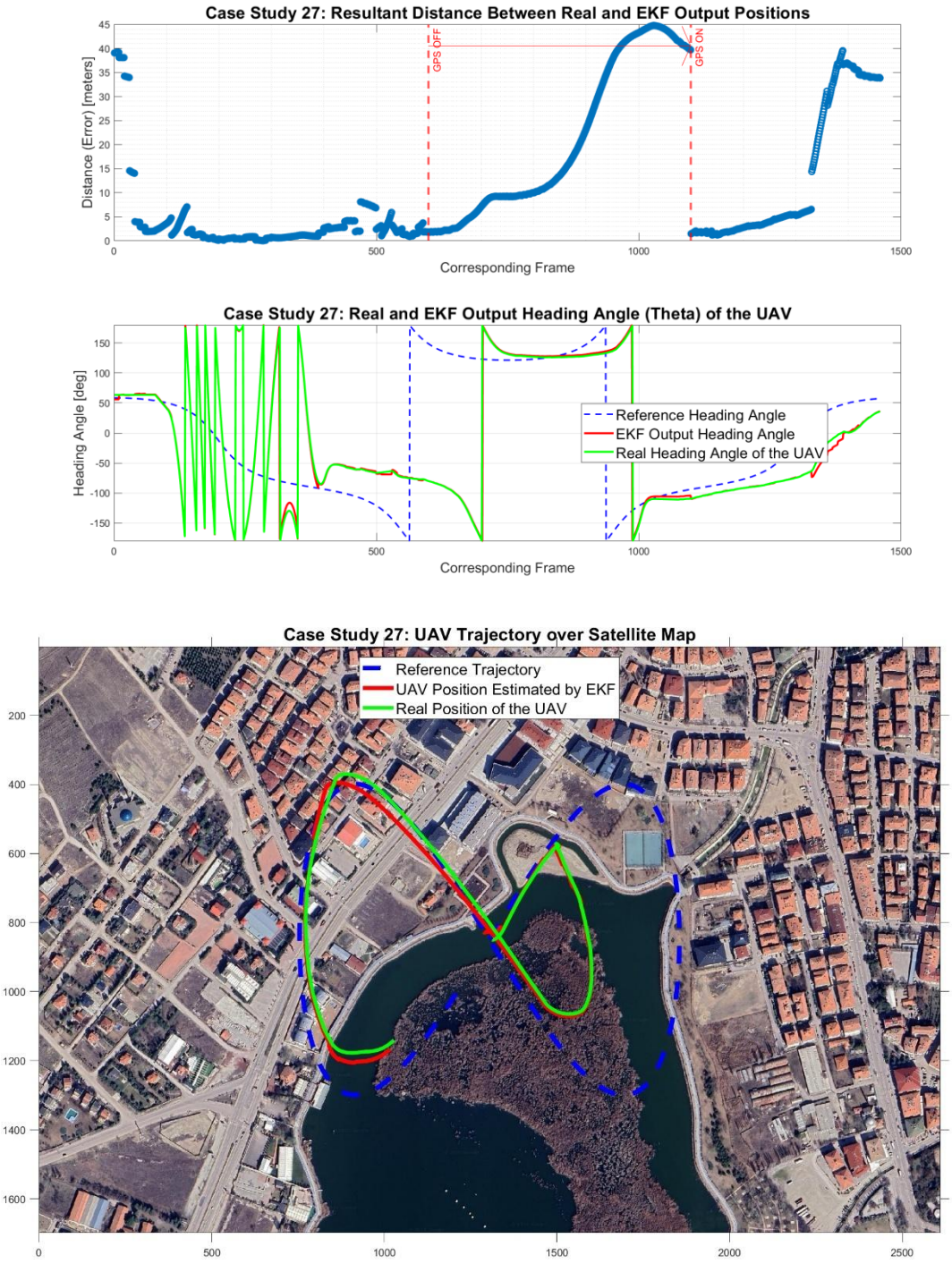

Figure A.21. Case study 27 (MPC mode Switching, Scenario 4 -middle section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

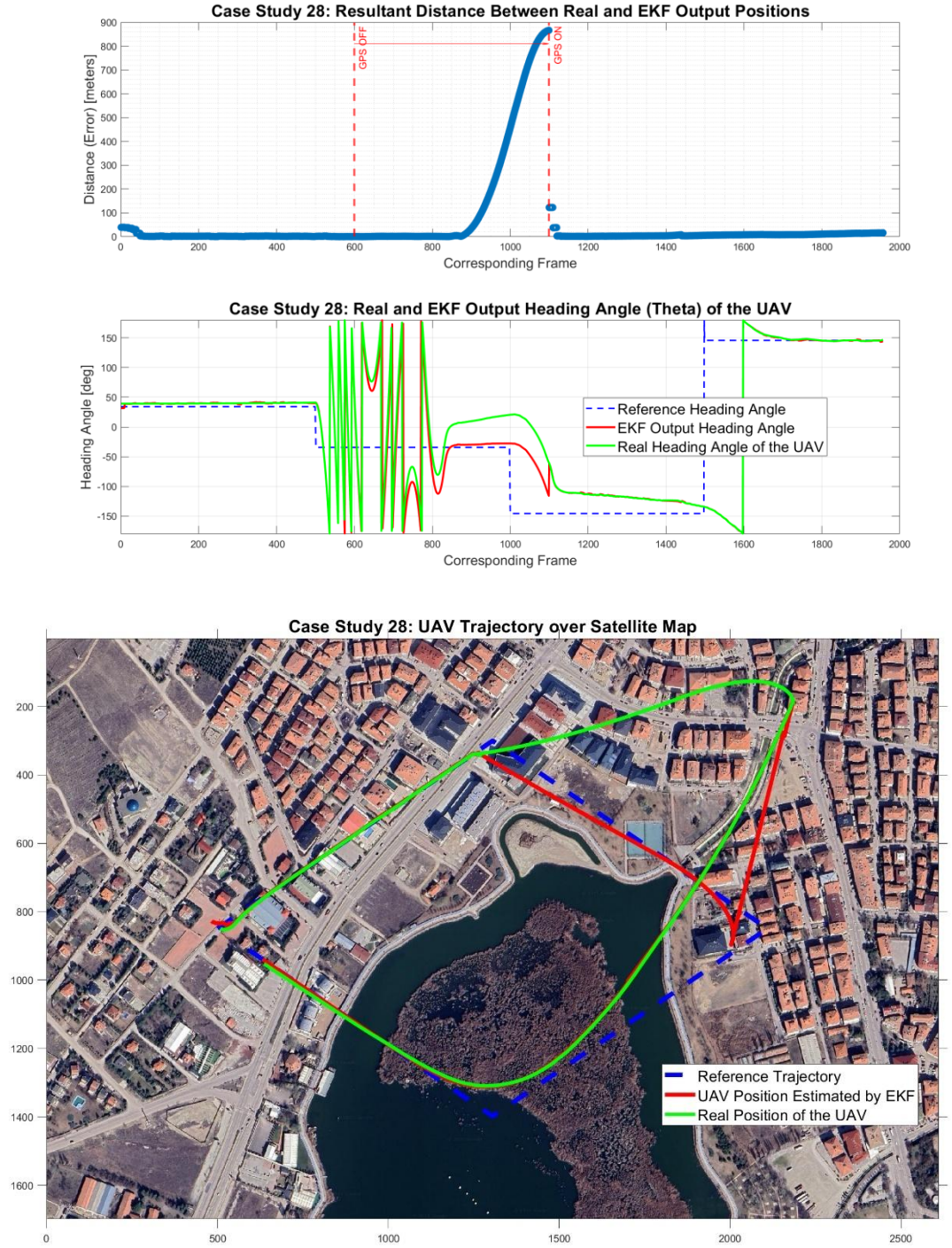

Figure A.22. Case study 28 (MPC mode Switching, Scenario 4 -middle section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

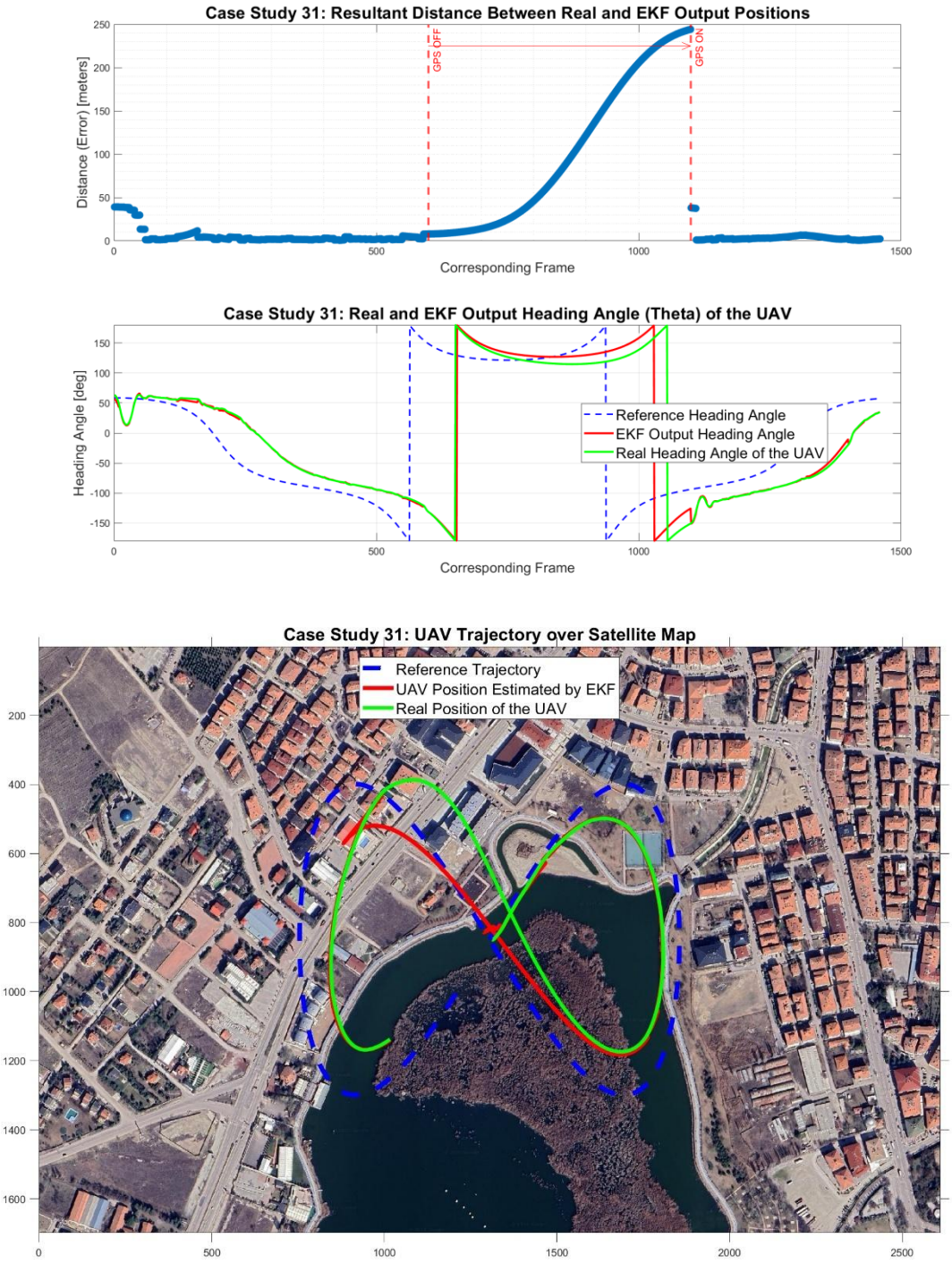

Figure A.23. Case study 31 (MPC mode Exponential, Scenario 4 -middle section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

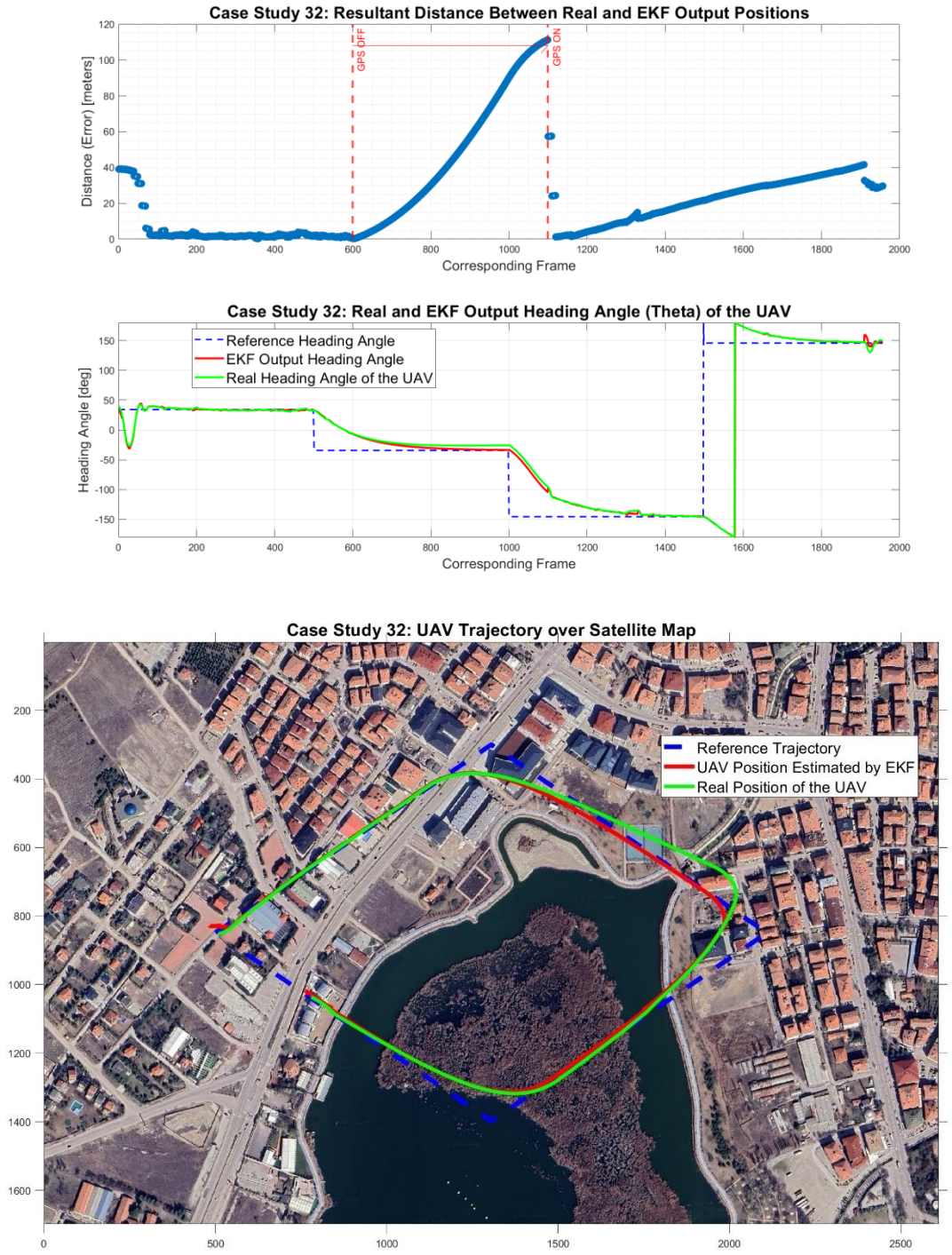

Figure A.24. Case study 32 (MPC mode Exponential, Scenario 4 -middle section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

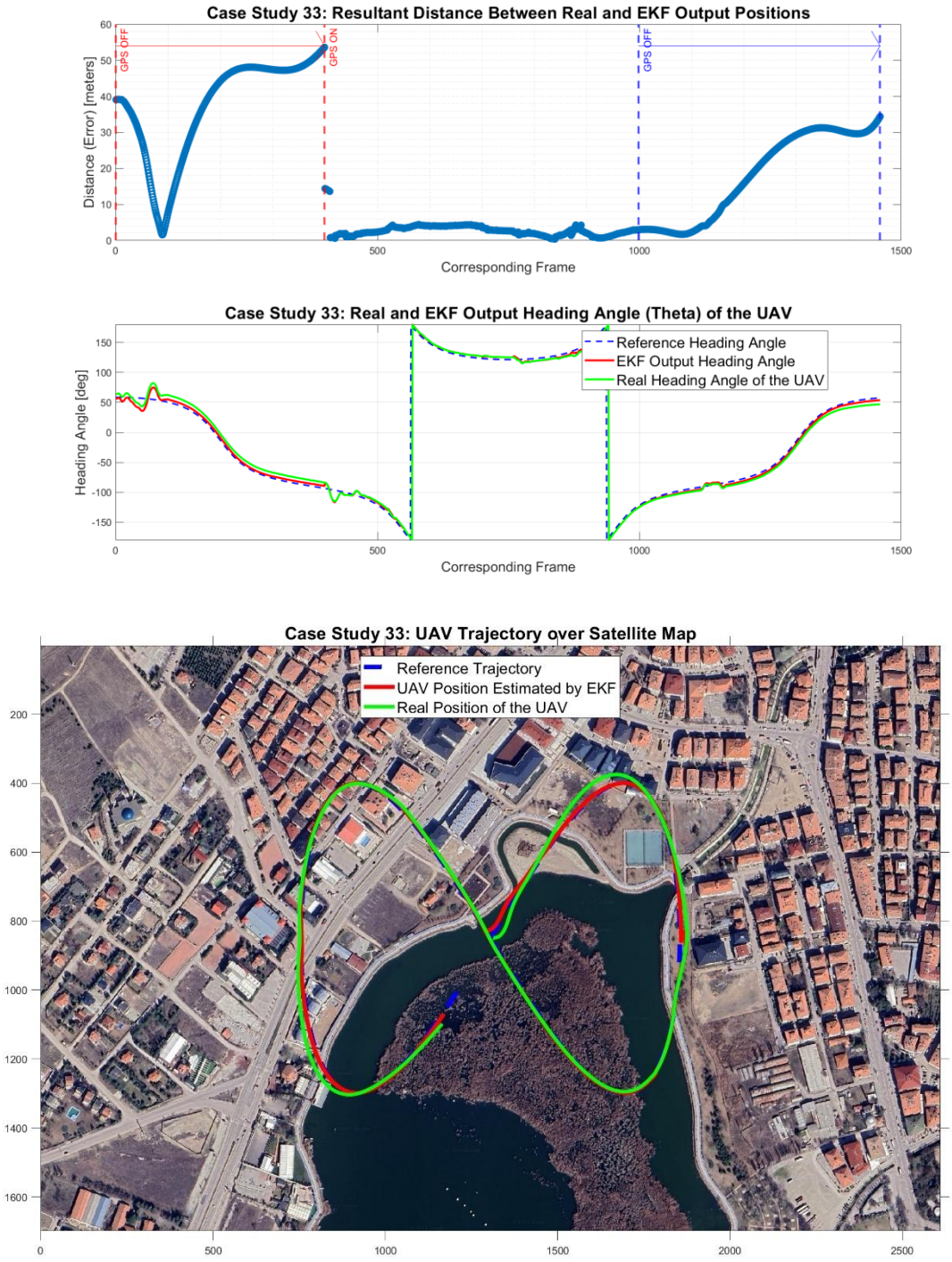

Figure A.25. Case study 33 (MPC mode Regulating, Scenario 5 -first and last section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

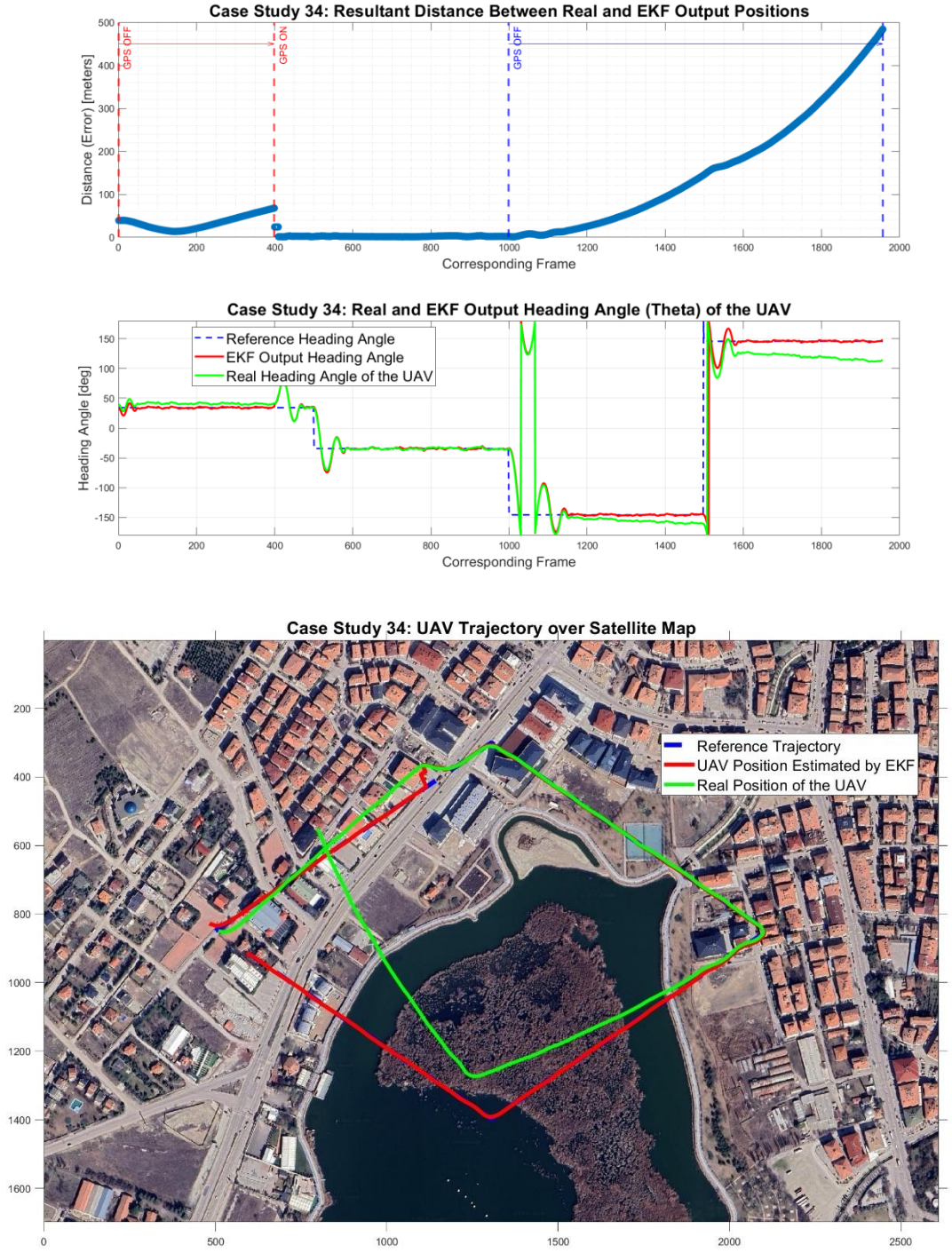

Figure A.26. Case study 34 (MPC mode Regulating, Scenario 5 -first and last section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

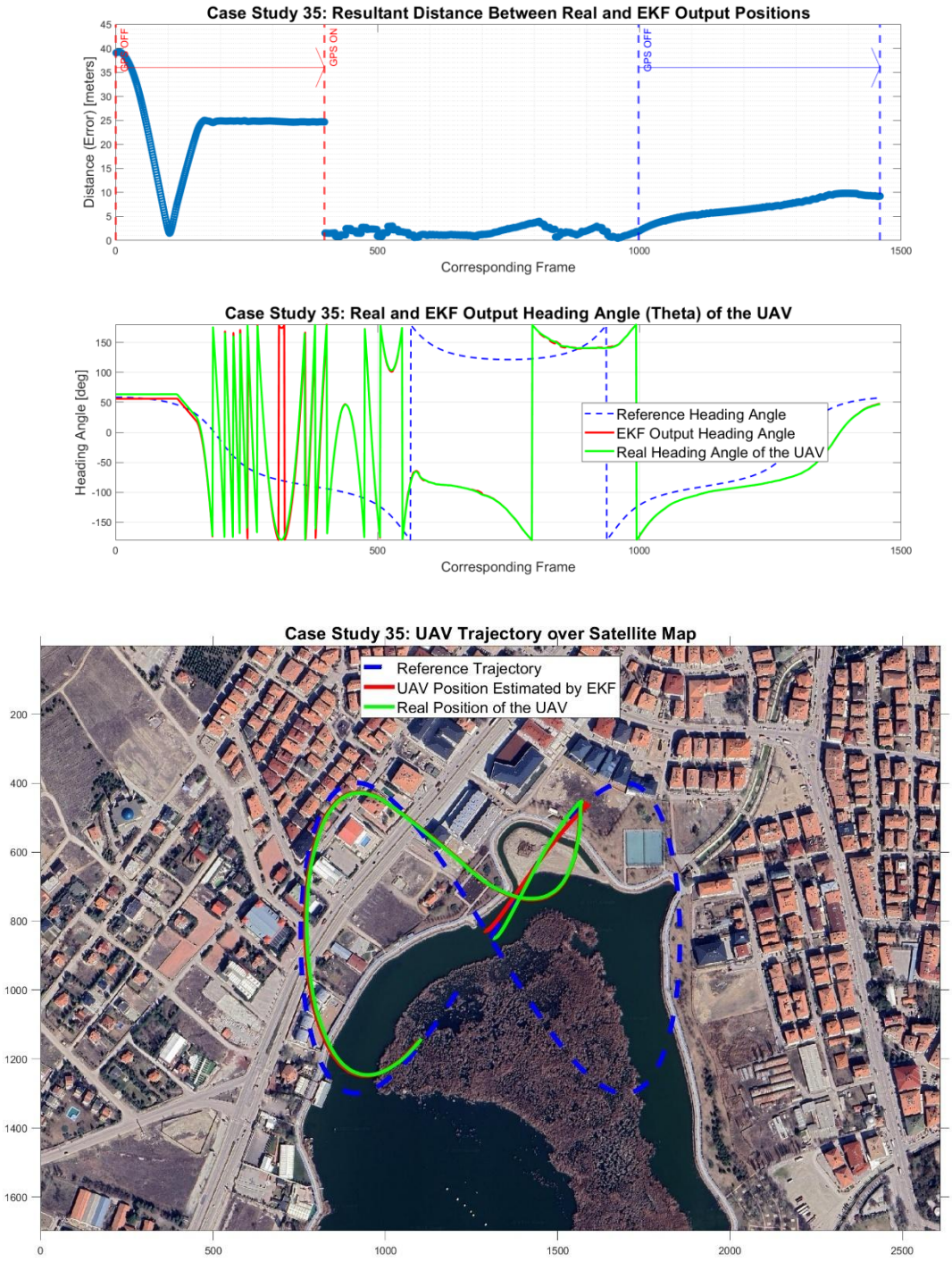

Figure A.27. Case study 35 (MPC mode Switching, Scenario 5 -first and last section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

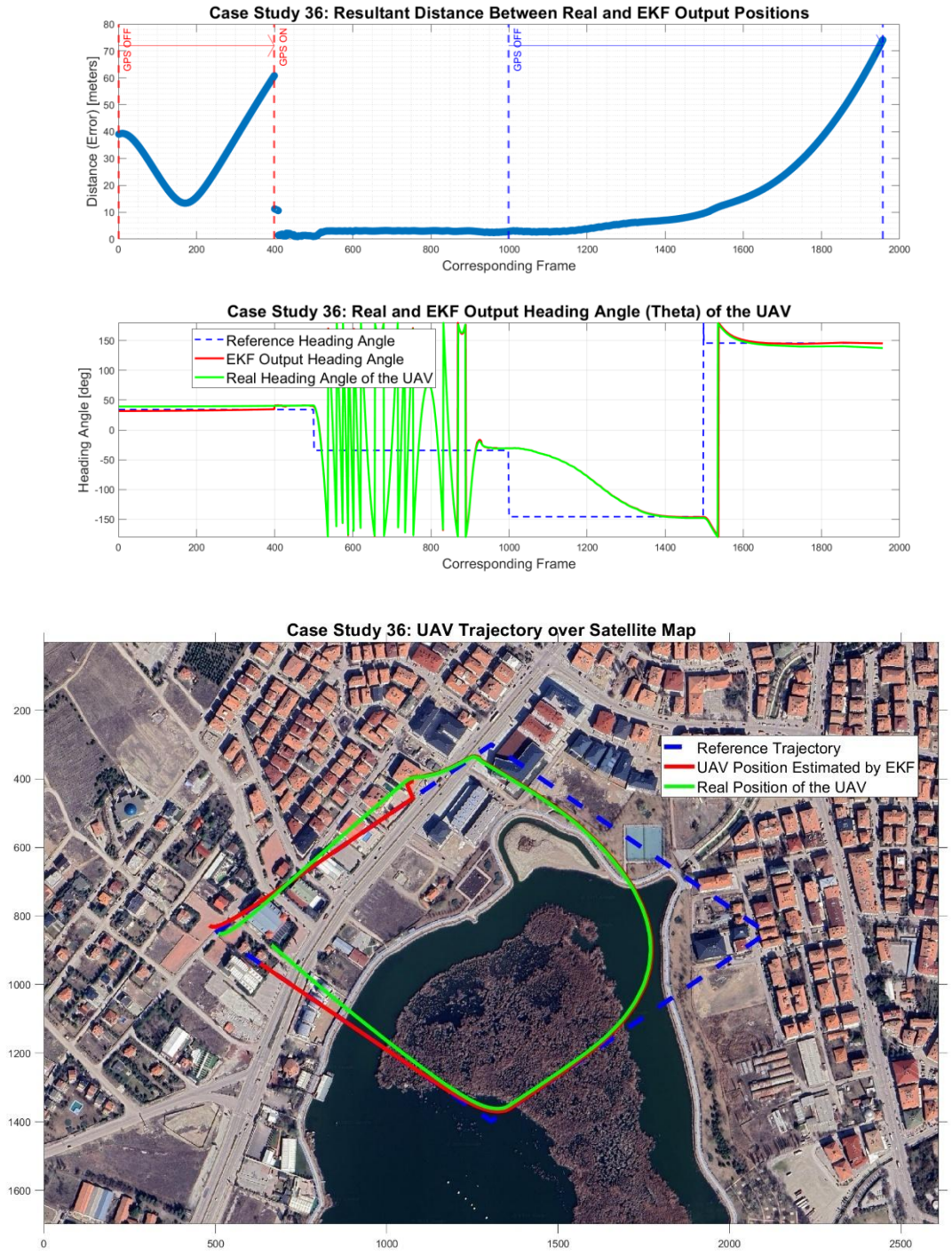

Figure A.28. Case study 36 (MPC mode Switching, Scenario 5 -first and last section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

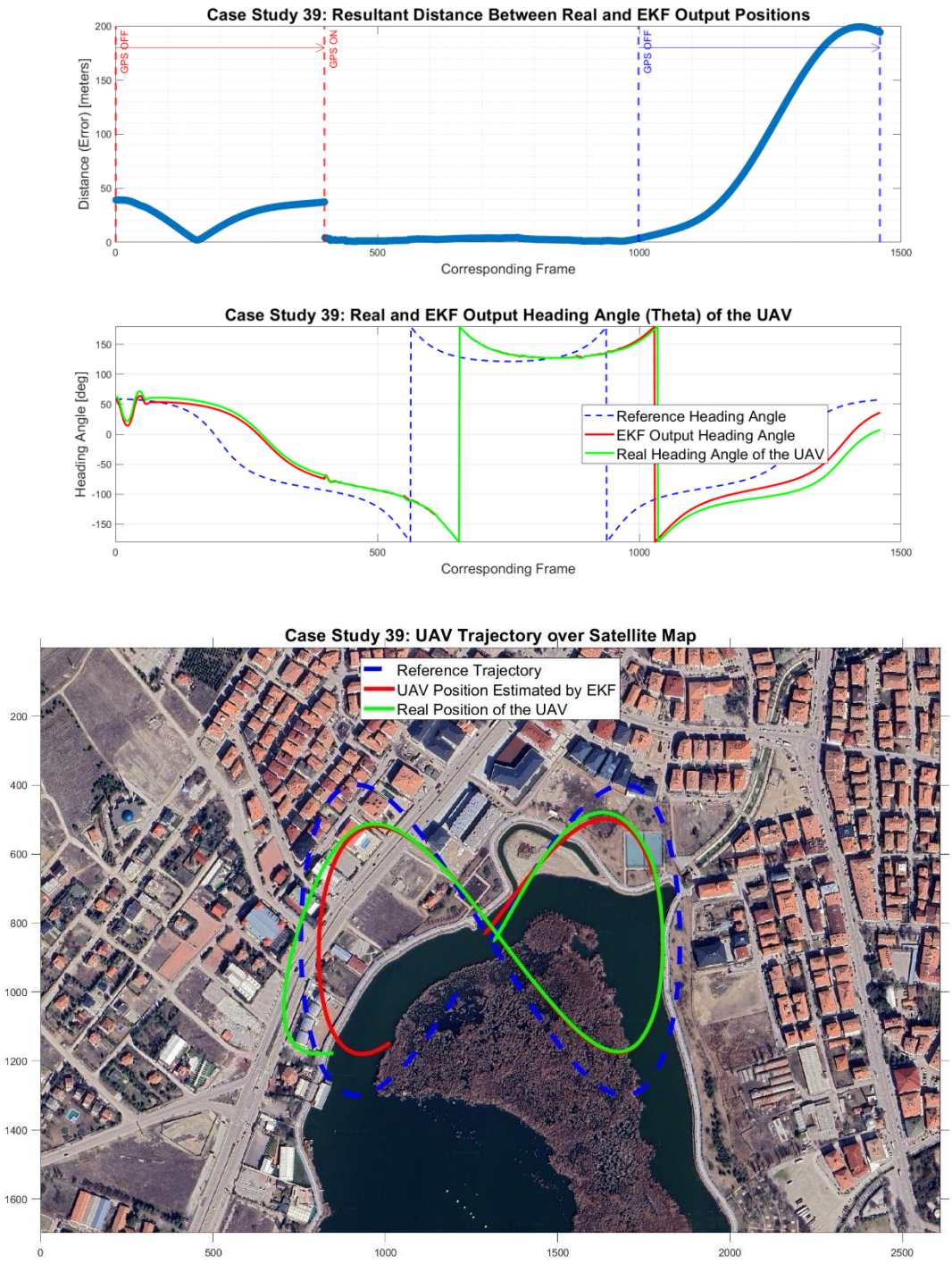

Figure A.29. Case study 39 (MPC mode Exponential, Scenario 5 -first and last section vGPS loss, 8-shape path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map

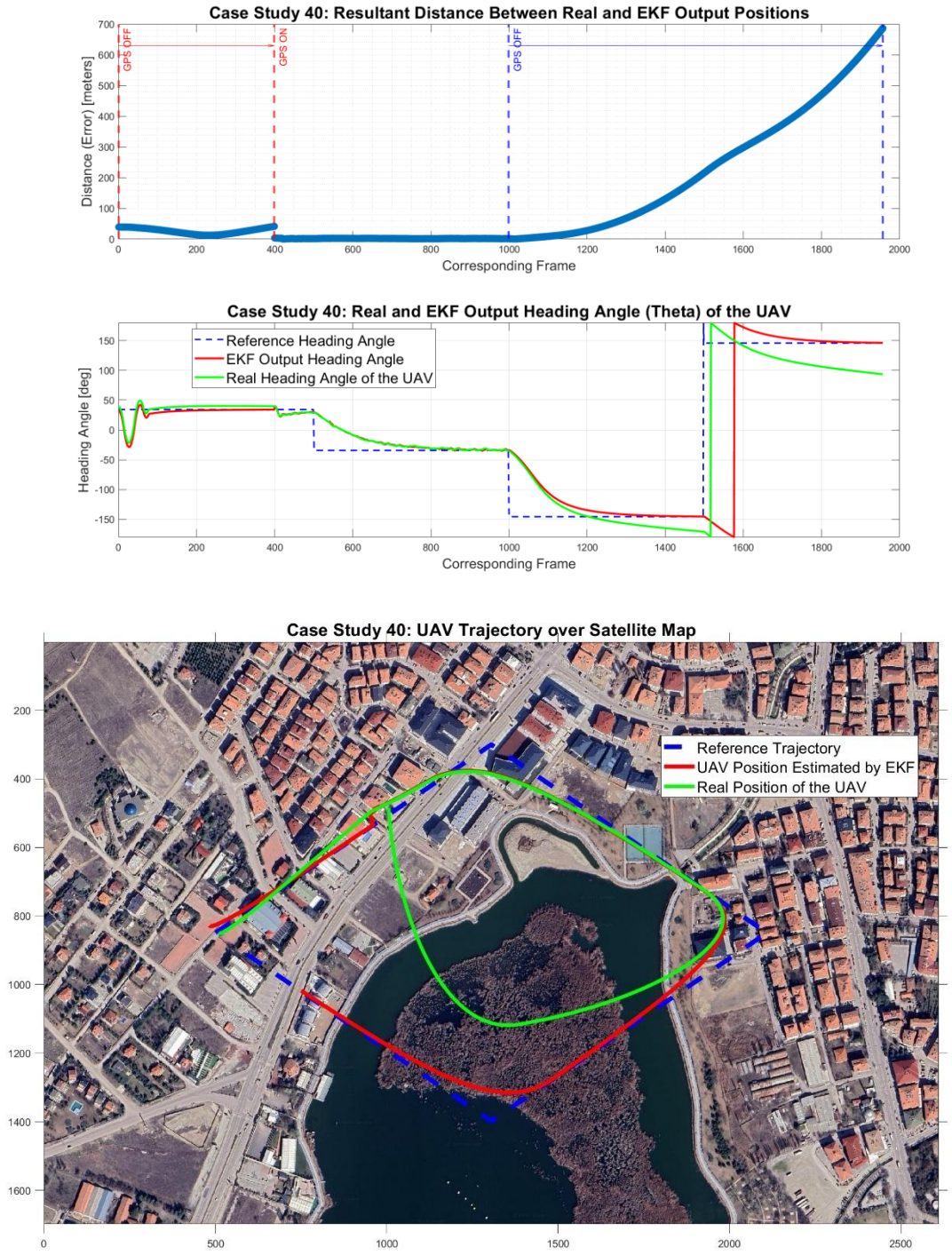

Figure A.30. Case study 40 (MPC mode Exponential, Scenario 5 -first and last section vGPS loss, diamond path): Position error between EKF and real UAV, heading angles of EKF output, real UAV and reference (trajectory), positions from EKF output, real UAV and reference (trajectory) over satellite map