



# Visual Pose Estimation of Rescue Unmanned Surface Vehicle From Unmanned Aerial System

Jan Dufek\* and Robin Murphy

Department of Computer Science and Engineering, Texas A&M University, College Station, TX, United States

This article addresses the problem of how to visually estimate the pose of a rescue unmanned surface vehicle (USV) using an unmanned aerial system (UAS) in marine mass casualty events. A UAS visually navigating the USV can help solve problems with teleoperation and manpower requirements. The solution has to estimate full pose (both position and orientation) and has to work in an outdoor environment from oblique view angle (up to  $85^\circ$  from nadir) at large distances (180 m) in real-time (5 Hz) and assume both moving UAS (up to  $22 \text{ m s}^{-1}$ ) and moving object (up to  $10 \text{ m s}^{-1}$ ). None of the 58 reviewed studies satisfied all those requirements. This article presents two algorithms for visual position estimation using the object's hue (thresholding and histogramming) and four techniques for visual orientation estimation using the object's shape while satisfying those requirements. Four physical experiments were performed to validate the feasibility and compare the thresholding and histogramming algorithms. The histogramming had statistically significantly lower position estimation error compared to thresholding for all four trials ( $p$ -value ranged from  $\sim 0$  to  $8.23263 \times 10^{-29}$ ), but it only had statistically significantly lower orientation estimation error for two of the trials ( $p$ -values  $3.51852 \times 10^{-39}$  and  $1.32762 \times 10^{-46}$ ). The mean position estimation error ranged from 7 to 43 px while the mean orientation estimation error ranged from 0.134 to 0.480 rad. The histogramming algorithm demonstrated feasibility for variations in environmental conditions and physical settings while requiring fewer parameters than thresholding. However, three problems were identified. The orientation estimation error was quite large for both algorithms, both algorithms required manual tuning before each trial, and both algorithms were not robust enough to recover from significant changes in illumination conditions. To reduce the orientation estimation error, inverse perspective warping will be necessary to reduce the perspective distortion. To eliminate the necessity for tuning and increase the robustness, a machine learning approach to pose estimation might ultimately be a better solution.

**Keywords:** visual pose estimation, visual localization, heterogenous multi-robot team, search and rescue robotics, field robotics, computer vision, marine robotics, aerial robotics

## OPEN ACCESS

### Edited by:

Wei Li,  
Nanjing University, China

### Reviewed by:

Muhammad Ahmad Kamran,  
Pusan National University,  
South Korea  
Önder Tutsoy,  
Adana Science and  
Technology University, Turkey

### \*Correspondence:

Jan Dufek  
dufek@tamu.edu

### Specialty section:

This article was submitted to  
Multi-Robot Systems,  
a section of the journal  
Frontiers in Robotics and AI

**Received:** 23 January 2019

**Accepted:** 08 May 2019

**Published:** 31 May 2019

### Citation:

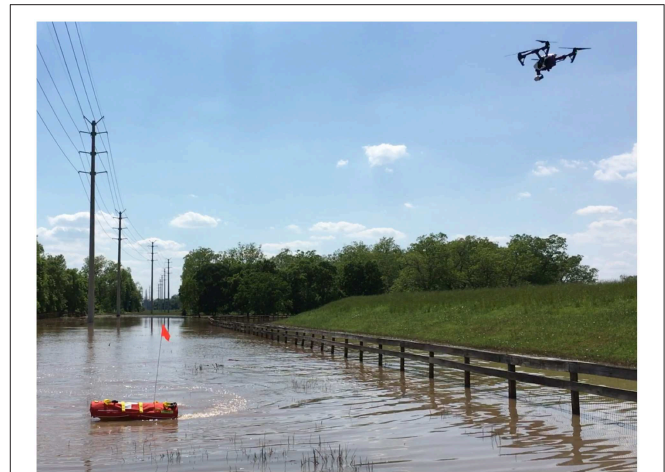
Dufek J and Murphy R (2019) Visual  
Pose Estimation of Rescue Unmanned  
Surface Vehicle From Unmanned  
Aerial System. *Front. Robot. AI* 6:42.  
doi: 10.3389/frobt.2019.00042

## 1. INTRODUCTION

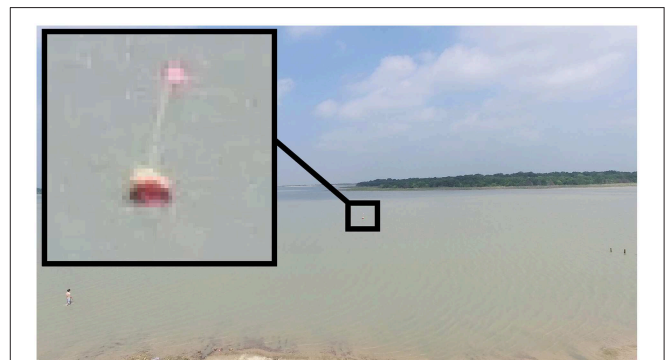
Using a UAS to visually navigate a rescue USV to victims can help responders during marine mass casualty events. First responders lack the manpower to effectively address marine mass casualty events during which it is not uncommon to have 80 victims in the water at the same time. While rescue USVs can help, teleoperation is problematic due to the lack of depth perception when USV is far away and the manpower requirements, and global positioning system (GPS) waypoint navigation is not precise enough as exact GPS coordinates of the victims are unknown and victims might drift with waves and currents. A UAS can visually navigate the USV to the victims by using visual feedback eliminating the need for teleoperation or GPS-based navigation as was shown in our previous work presented in Dufek et al. (2017), Karnan et al. (2017), Xiao et al. (2017), and Dufek and Murphy (2018). Such a heterogeneous multi-robot team of a UAS and a USV is illustrated in **Figure 1**.

Visual navigation requires visual pose estimation of the USV which can be generalized to visual pose estimation of any fast moving object using UAS leading to the following problem statement: Visually estimate the pose of a fast maneuvering object using moving UAS, relative to UAS's image frame of reference, in an outdoor environment from an oblique view angle at large distances in real-time. In this article, a USV is taken as an instance of this object. The solution must satisfy working environment requirements, vehicular requirements, physical settings requirements, and desired output requirements. The working environment requirements are that the solution must work in an outdoor water environment where the rescues are being performed. Vehicular requirements are that the solution must work for moving UAS and moving USV. Small inexpensive UAS that would be used in rescue application (e.g., DJI Inspire 1) can move at speeds of up to  $22 \text{ m s}^{-1}$ . Rescue- USVs (e.g., EMILY) can move at speeds of up to  $22 \text{ m s}^{-1}$  and make abrupt maneuvers with turn rate of up to  $180^\circ \text{ s}^{-1}$ . The physical settings requirements are that the solution must work at a large distance between UAS and USV and at an oblique view angle. The radius of marine mass casualty event rescue operations is typically 180 m. This distance causes the spatial resolution of the USV to be very low as illustrated in **Figure 2** implicating that fiducial markers encoding full pose (e.g., AprilTag) would not be visible. The UAS might be operated above shore while USV is in the water causing the oblique view angle of up to  $85^\circ$  from nadir. The desired output requirements are that the output should be full pose (both position and orientation) and it should be updated in real-time (at least 5 Hz) to enable the visual navigation. While it might be beneficial to use multiple UAS as well as to estimate the pose of multiple objects simultaneously, for the sake of simplicity, it is assumed in this paper that there is only a single UAS and it is estimating the pose of a single object.

The contributions of this article are addressing a problem of how can a moving small UAS visually estimate the position and orientation of a fast maneuvering object, relative to UAS's image frame of reference, in an outdoor environment from an oblique view angle at large distances in real-time, that previous work failed to fully address, analyzing this problem on an



**FIGURE 1** | DJI Inspire 1 (UAS) assisting EMILY (USV) using visual feedback.



**FIGURE 2** | The spatial resolution of the object in the UAS view might be very low.

example of a life-saving USV, proposing a method satisfying all the introduced requirements, and comparing two algorithms for solving this problem. First, this work addresses a problem defined above that none of the 58 reviewed studies fully covered. Second, it presents an analysis of this problem and justifies why some approaches would not work. Third, it presents a hue-based visual pose estimation method consisting of two algorithms for visual position estimation and four techniques for visual orientation estimation. Fourth, it compares two algorithms, thresholding and histogramming, for visual position estimation combined with one technique for orientation estimation in four physical experiments.

The rest of this article is organized as follows. Section 2 discusses related work in visual pose estimation of an object from UAS and identifies gaps and opportunities in research, most notably the lack of solution satisfying all the requirements discussed above. Section 3 presents two algorithms for position estimation (thresholding and histogramming) and four shape analysis techniques for orientation estimation. Section 4 details implementation of those algorithms on a laptop using C++ and OpenCV and describes platforms used for the implementation.

Section 5 introduces experimental methodology to determine pose estimation error relative to manually annotated ground truth and presents results from four physical trials comparing thresholding and histogramming. Section 6 discusses reasons histogramming algorithm is better than thresholding (it can better model the object under local illumination conditions), points out problems with the proposed algorithms (particularly large orientation estimation error, necessity of tuning, and low robustness to significant illumination changes), examines how the experiments could have been improved (most notably testing for robustness in addition to precision), and indicates machine learning would be suitable to explore for future work. Finally, section 7 summarizes the article.

## 2. RELATED WORK

Fifty-eight studies from the robotics and computer vision scientific literature were reviewed. While all the eight requirements introduced in section 1 were partially addressed by those studies, none satisfied all the requirements combined. The studies were classified to seven categories depending on what was the object for which the visual pose estimation using UAS was done: UGV (27 studies), car (9 studies), person (7 studies), landing platform (7 studies), standalone fiducial marker (3 studies), another UAS (2 studies), and other objects (3 studies). There were no studies on visual pose estimation of USV from UAS. All the categories except UGV category were limited in scope to only those studies where the visual pose estimation was done outdoors.

Each study was evaluated using eight criteria to validate if the particular study satisfied the requirements introduced in section 1. To validate the working environment requirements, one criterion was evaluated: (1) Were the experiments performed indoors or outdoors? To validate the vehicular requirements, two criteria were evaluated: (1) Was the UAS static or moving? (2) Was the object static or moving? To validate physical settings requirements, three criteria were evaluated: (1) Was the distance between UAS and the object short or long (distance is considered short for the purpose of this review if less than or equal 30 m)? (2) Was a fiducial marker used or not? (3) Was the view from UAS nadir or oblique? To validate desired output requirements, two criteria were evaluated: (1) Was just a position estimated or full pose (both position and orientation)? (2) Was the output generated real-time (i.e., updating at least at 5 Hz)? **Table 1** lists all the studies and for each, it shows which criteria were satisfied.

A UAS has been used to visually estimate the pose of UGV in 27 studies relying on fiducial markers and the nadir view or only tested indoors at a short distance. Majority of those studies (20 out of 27) relied on a fiducial marker mounted on the UGV and assumed the nadir view (Dixon et al., 2001; Cognetti et al., 2014; Aranda et al., 2015; Harik et al., 2015a,b, 2016, 2017; Hausman et al., 2015, 2016; Laiacker et al., 2015; Rosa et al., 2015; Byun et al., 2016; Cantelli et al., 2016; Santana et al., 2016; Wang et al., 2016, 2017; Araar et al., 2017; Battiato et al., 2017; Gomez-Avila et al., 2018; Harikumar et al., 2018). Four studies assumed an oblique view, but still relied on fiducial markers mounted on

the UGV (Rao et al., 2003, 2004, 2005, 2006). Only three studies estimated pose of UGV without any fiducial markers. Gao et al. (2014) proposed a method to track UGV from UAS, but only the position was estimated (not orientation) and the method was only tested indoors at a short distance. Chen et al. (2016) also estimated position only (no orientation) of UGV with UAS flying directly above at a short distance looking nadir in an indoor experiment. Hoang et al. (2017) proposed a method to track a UGV, but it was tested only indoors at a short distance (1.5 m).

A UAS has been used to visually estimate the pose of cars in nine studies estimating position only or requiring the nadir view (Siam and ElHelw, 2012; Siam et al., 2012; van Eekeren et al., 2015; Ma et al., 2016; Watanabe et al., 2016; Askar et al., 2017; Chen et al., 2017; Kim et al., 2017; Kaufmann et al., 2018). All those studies except Watanabe et al. (2016) only estimated the position and not orientation. The distance between the UAS and the car/cars was relatively long, but at the same time, cars are large objects so the object's spatial resolution was still relatively high. Four of those studies (Ma et al., 2016; Chen et al., 2017; Kim et al., 2017; Kaufmann et al., 2018) used static UAS and the proposed methods required known ground reference points in the environment. For three studies (Ma et al., 2016; Chen et al., 2017; Kim et al., 2017), pedestrians were tracked as well, but the methods were not real-time. Ma et al. (2016) and Chen et al. (2017) assumed the nadir view and constant altitude. For Watanabe et al. (2016), orientation was estimated, but the nadir view was assumed and the distance between UAS and the car was short. Van Eekeren et al. (2015) used very high resolution images (116 Mpx) from high altitude. While the view was oblique, the very high altitude caused the view to appear very close to nadir. This method also required a 3D reconstruction to get the target's height and only worked with cars moving faster than  $10 \text{ km h}^{-1}$ .

A UAS has been used to visually estimate the pose of a person in seven studies estimating position only on short distances (Lim and Sinha, 2015; Bian et al., 2016; Mendonça et al., 2016; Monajjemi et al., 2016; Cheng et al., 2017; Lee et al., 2018; Liu et al., 2018). In those studies, only the position was estimated (not orientation) and the person was close to the UAS. The method proposed in Lim and Sinha (2015) additionally required the person's height. For Mendonça et al. (2016) and Monajjemi et al. (2016), the target person was assumed to be static. For Mendonça et al. (2016), the nadir view from UAS was assumed.

A UAS has been used to visually estimate the pose of a landing platform during an autonomous landing of a UAS in seven studies relying on fiducial markers, the nadir view, and short distances (Medeiros et al., 2015; Kim et al., 2016; Lee et al., 2016; Cabrera-Ponce and Martinez-Carranza, 2017; Collins et al., 2017; Junaid et al., 2017; Patruno et al., 2018). All of those studies assumed the nadir view and used a fiducial marker on the landing platform. All the experiment were done on a short distance since for autonomous landing the visual pose estimation of a landing platform is only done in the final stages of landing. All the studies except two (Kim et al., 2016; Lee et al., 2016) used static landing platform. All studies except three (Medeiros et al., 2015; Collins et al., 2017; Patruno et al., 2018) estimated only position and not orientation.

**TABLE 1** | The requirements violations for the 58 reviewed studies.

References	Object	Outdoors	Moving UAS	Moving object	Oblique view	Large distance	Estimated distance	No fiducials	Full pose	Real-time	Total violations
Dixon et al., 2001	UGV	✗	✗	✓	✗	✗		✗	✓	✓	6
Rao et al., 2003	UGV	✗	✗	✓	✓	✗		✗	✓	✓	5
Rao et al., 2004	UGV	✗	✗	✓	✓	✗		✗	✓	✓	5
Rao et al., 2005	UGV	✗	✗	✓	✓	✗		✗	✓	✓	5
Rao et al., 2006	UGV	✗	✗	✓	✓	✗		✗	✓	✓	5
Cognetti et al., 2014	UGV	✗	✗	✓	✗	✗		✗	✓	✓	6
Gao et al., 2014	UGV	✗	✓	✓	✓	✗		✓	✗	✓	3
Aranda et al., 2015	UGV	✗	✗	✓	✗	✗		✗	✓	✓	6
Harik et al., 2015a	UGV	✗	✗	✓	✗	✗		✗	✓	✓	6
Harik et al., 2015b	UGV	✗	✗	✓	✗	✗		✗	✓	✓	6
Hausman et al., 2015	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Laiacker et al., 2015	UGV	✓	✓	✗	✗	✗	10 m	✗	✓	✓	4
Rosa et al., 2015	UGV	✗	✗	✓	✗	✗		✗	✓	✓	6
Byun et al., 2016	UGV	✗	✓	✗	✗	✗		✗	✗	✓	6
Cantelli et al., 2016	UGV	✓	✓	✓	✗	✗	5 m	✗	✓	✓	3
Chen et al., 2016	UGV	✗	✓	✓	✗	✗		✓	✗	✓	4
Harik et al., 2016	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Hausman et al., 2016	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Santana et al., 2016	UGV	✓	✓	✓	✗	✗	1 m	✗	✓	✓	3
Wang et al., 2016	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Araar et al., 2017	UGV	✗	✓	✗	✗	✗		✗	✓	✓	5
Battiato et al., 2017	UGV	✓	✓	✗	✗	✗	10 m	✗	✗	✓	5
Harik et al., 2017	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Hoang et al., 2017	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Wang et al., 2017	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Gomez-Avila et al., 2018	UGV	✗	✓	✓	✗	✗		✗	✓	✓	4
Harikumar et al., 2018	UGV	✓	✓	✓	✗	✗	10 m	✗	✓	✓	3
Siam et al., 2012	Car	✓	✓	✓	✓	✓	70 m	✓	✗	✓	1
Siam and ElHelw, 2012	Car	✓	✓	✓	✓	✓	70 m	✓	✗	✓	1
van Eekeren et al., 2015	Car	✓	✓	✓	✓	✓	500 m	✓	✗	✓	1
Ma et al., 2016	Car	✓	✗	✓	✗	✓	70 m	✓	✗	✗	4
Watanabe et al., 2016	Car	✓	✓	✓	✗	✗	30 m	✓	✓	✓	2
Askar et al., 2017	Car	✓	✓	✓	✓	✓	70 m	✓	✗	✓	1
Chen et al., 2017	Car	✓	✗	✓	✗	✓	70 m	✓	✗	✗	4
Kim et al., 2017	Car	✓	✗	✓	✓	✓	50 m	✓	✗	✗	3
Kaufmann et al., 2018	Car	✓	✗	✓	✓	✓	100 m	✓	✗	✓	2
Lim and Sinha, 2015	Person	✓	✓	✓	✓	✗	10 m	✓	✗	✓	2
Mendonça et al., 2016	Person	✓	✓	✗	✗	✗	10 m	✓	✗	✓	4
Bian et al., 2016	Person	✓	✓	✓	✓	✗	10 m	✓	✗	✓	2
Monajemi et al., 2016	Person	✓	✓	✗	✓	✗	30 m	✓	✗	✓	3
Cheng et al., 2017	Person	✓	✓	✓	✓	✗	5 m	✓	✗	✓	2
Lee et al., 2018	Person	✓	✓	✓	✓	✗	30 m	✓	✗	✓	2
Liu et al., 2018	Person	✓	✓	✓	✓	✗	14 m	✓	✗	✓	2

(Continued)

TABLE 1 | Continued

References	Object	Outdoors	Moving UAS	Moving object	Oblique view	Large distance	Estimated distance	No fiducials	Full pose	Real-time	Total violations
Medeiros et al., 2015	Landing platform	✓	✓	✗	✗	✗	10m	✗	✓	✓	4
Kim et al., 2016	Landing Platform	✓	✓	✓	✗	✗	10m	✗	✗	✓	4
Lee et al., 2016	Landing platform	✓	✓	✓	✗	✗	10m	✗	✗	✓	4
Cabrera-Ponce and Martinez-Carranza, 2017	Landing platform	✓	✓	✗	✗	✗	5m	✗	✗	✓	5
Collins et al., 2017	Landing platform	✓	✓	✗	✗	✗	22m	✗	✓	✓	4
Junaid et al., 2017	Landing platform	✓	✓	✗	✗	✗	5m	✗	✗	✓	5
Patruno et al., 2018	Landing platform	✓	✓	✗	✗	✗	5m	✗	✓	✓	4
Feng et al., 2007	Standalone Fiducial marker	✓	✗	✗	✗	✗	10m	✗	✗	✓	6
Cho et al., 2016	Standalone Fiducial marker	✓	✓	✗	✓	✓	200m	✗	✗	✓	3
Hinas et al., 2017	Standalone Fiducial marker	✓	✓	✗	✗	✗	30m	✗	✗	✓	5
Liu and Feng, 2018	UAS	✓	✓	✓	✓	✗	30m	✓	✗	✓	2
Wang et al., 2018	UAS	✓	✗	✗	✓	✗	5m	✓	✓	✗	4
Máthé et al., 2016	Railway semaphore	✓	✓	✗	✓	✗	5m	✓	✗	✓	3
Koo et al., 2017	Jellyfish	✓	✗	✗	✗	✗	10m	✓	✗	✓	5
Liu et al., 2017	General objects	✓	✓	✗	✗	✗	15m	✓	✗	✓	4
Violationsperrequirement		22	17	18	37	49		35	30	4	

The distance between the UAS and the object was considered large if more than 30m. The numerical distance is only listed for outdoor studies and it was estimated from particular study's figures if not explicitly reported. A study was considered real-time if the update rate was more than 5 Hz. The last column lists the total number of requirements violations for a particular study. The last row lists the total number of studies that violated a particular requirement. The studies are grouped by the object for which the visual pose estimation was done. The groups are ordered by group size. Inside a single group, studies are ordered by year and then by first author last name. Green tick indicates the corresponding study satisfies corresponding requirement and red cross otherwise.

A UAS has been used to visually estimate the pose of a standalone fiducial marker in three studies assuming static object and estimating position only (Feng et al., 2007; Cho et al., 2016; Hinas et al., 2017). Two studies (Feng et al., 2007; Hinas et al., 2017) assumed the nadir view and were tested on a short distance. In addition, Feng et al. (2007) assumed a static UAS. A UAS has been used to visually estimate the pose of another UAS in two studies estimating position only or not running in real-time and tested only on a short distance (Liu and Feng, 2018; Wang et al., 2018). Both methods did pose estimation on a short distance. The method proposed by Wang et al. (2018) was not real-time and both UASs were static. Liu and Feng (2018) estimated the position only and not the orientation.

Finally, a UAS has been used to visually estimate the pose of three kinds of other objects in three studies all assuming static object, estimating position only, and testing only on a short distance. Máthé et al. (2016) estimated pose of railway semaphores, Koo et al. (2017) estimated pose of jellyfish in the water, and Liu et al. (2017) estimated pose of general objects. All those studies estimated only position (not orientation) on a short distance and assumed static objects. All except Máthé et al. (2016) assumed the nadir view. All except Koo et al. (2017) assumed a constant distance between UAS and objects, making specific assumptions about object sizes. Koo et al. (2017) also assumed static UAS. Máthé et al. (2016) had view fixed in a single horizontal plane looking forward.

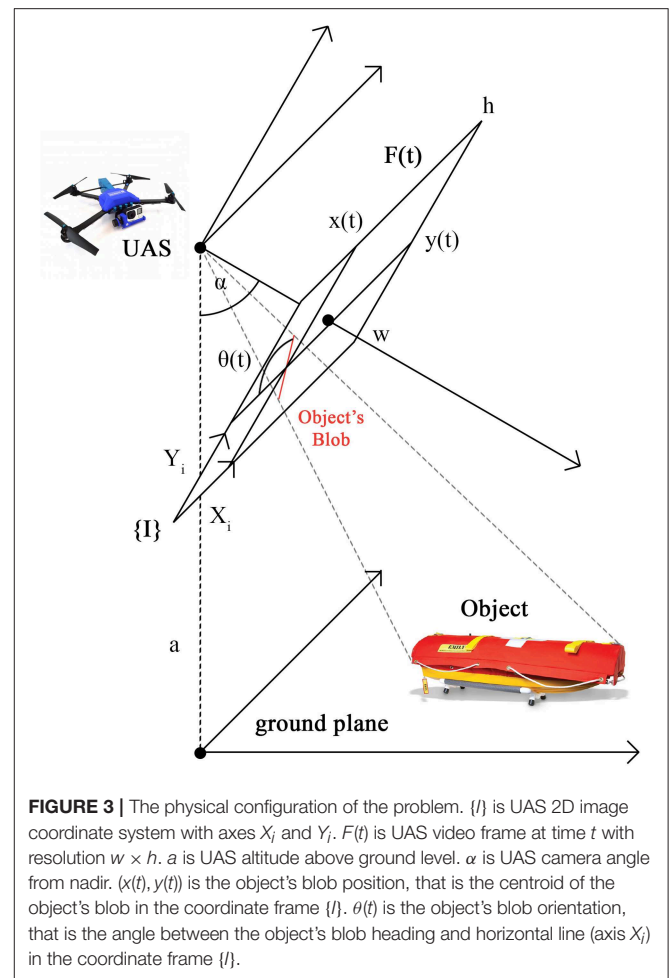
None of the 58 reviewed studies satisfied all the requirements introduced in section 1. The 58 studies had one or more of the requirement violations: indoor experiments only (22 studies), static UAS (17 studies), static object (18 studies), the nadir view only (37 studies), short distance less than 30 m between UAS and the object (49 studies), fiducial markers (35 studies), not estimating full pose including both position and orientation (30 studies), or not real-time with update at least 5 Hz (4 studies).

### 3. APPROACH

The review of the literature indicates that a method is needed for visual pose estimation, that would satisfy all the requirements: working outdoors, moving UAS, moving object, oblique view angle, large distance more than 30 m, no fiducial markers, estimating full pose including both position and orientation, and real-time update rate of more than 5 Hz. The approach taken in this article is based on the object's blob hue that is relatively invariant compared to the object's blob brightness, convexity, size, inertia ratio, features, or motion. Two algorithms for position estimation are presented. The first algorithm, thresholding, consists of seven steps applied for each video frame: blur, HSV conversion, value histogram equalization, thresholding, erosion, dilation, and contours detection. The second algorithm, histogramming, first takes user input to construct a hue histogram model of the object and then applies six steps on each video frame: blur, HSV conversion, value histogram equalization, hue histogram backprojection, thresholding on saturation and value, and CamShift algorithm for finding and tracking objects. Shape analysis is used for orientation estimation assuming the major axis of the object's blob indicates the object's blob heading. A total of four techniques for estimation of the principal axis of a blob are examined: line fitting, rectangle fitting, principal component analysis (PCA), and ellipse fitting.

The problem of visually estimating the pose of an object can be decomposed into two parts, position estimation and orientation estimation, where input for both are video frames from the UAS. The input is a sequence of  $n$  video frames  $F(t)$ ,  $t \in [1 \dots n]$  taken from a UAS with resolution  $w \times h$ ,  $w \in \mathbb{N}$ ,  $h \in \mathbb{N}$ . The UAS is flying at altitude  $a$  above ground level and the UAS camera angle is  $\alpha$  from nadir. A frame is defined as  $F(t) = ((R, G, B)_{u,v})$ ,  $u \in [1 \dots w]$ ,  $v \in [1 \dots h]$ ,  $R, G, B \in [0 \dots 255]$ . For the visual position estimation, the output is the coordinates of the centroid of the object's blob,  $\mathbf{x}(t) = (x(t), y(t))$ , in UAS 2D image coordinate system,  $\{I\}$ , at time  $t$ . For the visual orientation estimation, the output is an angle between the object's blob heading and horizontal line,  $\theta(t)$ , in UAS 2D image coordinate system,  $\{I\}$ , at time  $t$ . Then, the pose at time  $t$  is a tuple  $(\mathbf{x}(t), \theta(t))$ . The physical configuration of the problem is depicted in **Figure 3**.

The object's blob has to be first identified in the video frame, however, identification using the object's blob brightness, convexity, size, inertia ratio, features, or motion is problematic as those attributes are not invariant; however, object's blob hue is relatively invariant and can be exploited. The object is represented as a blob of pixels in the UAS video frame  $F(t)$ . To



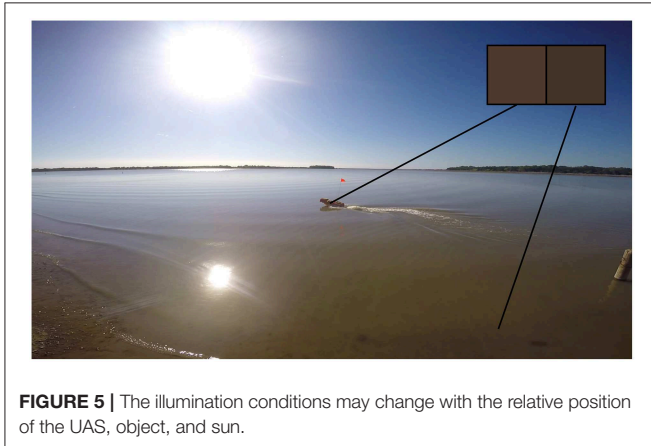
reliably identify this blob, some invariant property of this blob relative to changing  $t$  has to be identified.

The brightness of the blob representing the object in the video frames  $F(t)$  is not invariant in  $t$  because of the outdoor environmental conditions, particularly weather and sun. The brightness may change frequently depending on the weather and the relative pose of the object, UAS, and the sun. An example of weather affecting brightness are clouds that might temporarily cast shadows on the object. An example of the relative pose influence on the brightness is when the object that faces the UAS with its non-illuminated side changes its pose in a way that it now faces the UAS with its sun illuminated side as shown in **Figure 4**. In the figure, the brightness of a USV's blob changes as the USV completes a turn. The USV first displays its non-illuminated side making its blob look dark (left) and then after completing the turn, it displays the sun-illuminated side making its blob look bright (right). Another example is when the sun is in the field of view of the UAS's camera causing the white balance distortion in the video frames as can be seen in **Figure 5**. Consequently, the USV's color is very similar to the color of the surrounding water.

The convexity and concavity of the blob representing the object in the video frames  $F(t)$  is not invariant in  $t$  because of the outdoor environmental conditions, particularly water occlusions



**FIGURE 4** | Object's blob brightness may vary significantly in the video frames  $F(t)$  with varying  $t$  because of the outdoor environmental conditions.



**FIGURE 5** | The illumination conditions may change with the relative position of the UAS, object, and sun.

and shadows. The object might be temporarily partially occluded by water (waves or wake) or shadows causing the convexity and concavity of the blob to change over time.

The size of the blob representing the object in the video frames  $F(t)$  is not invariant in  $t$  due to the oblique view angle  $\alpha$  causing the distance from the UAS to change. The apparent size of the object in the image may change frequently depending on the distance of the object from the UAS and the relative pose of the object to the UAS. If the object moves away from the UAS, its size will decrease and vice versa as can be seen in **Figure 6**. In this figure, the relative size of the USV on the left is much larger than on the right because the USV is closer to the camera. Therefore, a priori assumptions about the object's blob size in video frames cannot be made.

The ratio of the minimum inertia to maximum inertia of the blob representing the object in the video frames  $F(t)$  is not invariant in  $t$  due to the oblique view angle  $\alpha$  causing the object's shape to change. The inertia ratio of the blob depends on the relative orientation of the object relative to the UAS. Taking a cylindrical object such as a USV as an example, if such object is facing toward the UAS, its blob will be circular (inertia ratio will be high), however, if it is turned sideways, its blob will be elliptical (inertia ratio will be low) as shown in **Figure 7**. On the left side of the figure, USV has a high inertia ratio comparing to the low inertia ratio on the right because it faces the camera.

Visual features of the blob representing the object in the video frames  $F(t)$  are not invariant in  $t$  due to the potentially

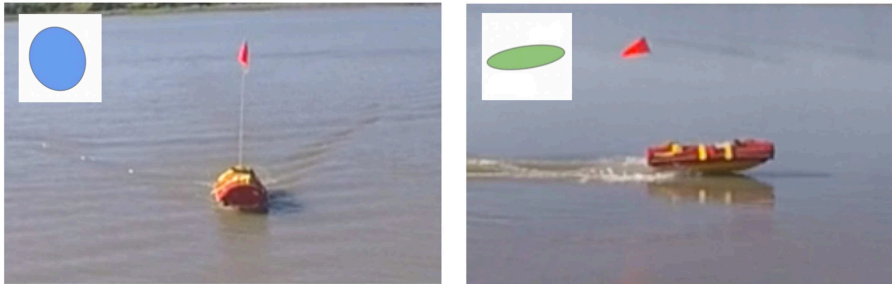
large distance between the object and the UAS. The object's blob might start relatively close to the UAS and then move very far (up to 180 m) causing the features to be lost with low spatial resolution. In the extreme case, the object's blob might be featureless due to the large distance between the object and the UAS causing the spatial resolution might be very low as illustrated in **Figure 2**. The figure shows DJI Phantom 3 Professional video feed with resolution  $2,132 \times 1,200$  px with USV being approximately 100 m away. As can be seen in the close-up view, the spatial resolution of USV is very low (about  $10 \text{ cm px}^{-1}$ ). The USV appears as a circle with the radius of only 5 px. The USV's image blob area is only about 80 px taking only 0.003 % of the image.

The motion of the blob representing the object in the video frames  $F(t)$  is not invariant in  $t$  due to the motion of the UAS. The UAS motion causes two problems. First, the motion of the object in the real world does not correspond to the motion of the object's blob in the video frames. For example, if both the object and the UAS are moving at the same speed in the same direction, the object's blob will be static in the video frames. Second, the UAS motion causes the background to move as well as shown in **Figure 8**. In the figure, two adjacent video frames from a moving UAS (on the left) were subtracted to reveal anything that moved in between those frames (on the right). It can be seen, that the USV (green circle) moving in the real world was not the only object that moved between the frames. The electric power pole (top red circle), the debris in the water (bottom red circle), the fence, and the parked cars all moved between the frames as well even though they were stationary in the real world. This background movement is not uniform due to the parallax effect. The parts of the background closer to the UAS will appear to move faster than the parts of the background further away.

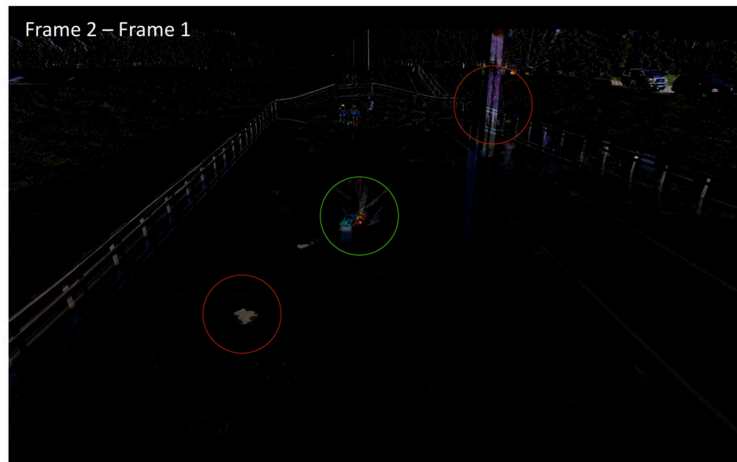
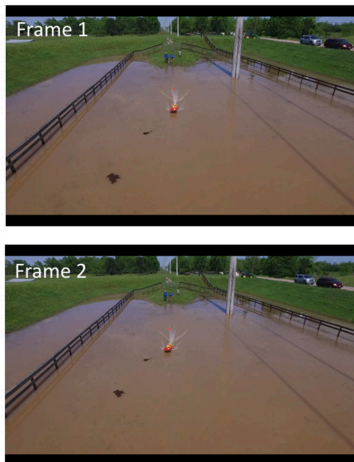
The hue of the blob representing the object in the video frames  $F(t)$ , unlike the brightness, convexity, size, inertia ratio, features, or motion, is relatively invariant in  $t$  and can be exploited. This assumes the object has a combination of colors that is unique in the UAS view. Those colors can then be exploited in identifying the object's blob. While the saturation and value of a particular color might change with illumination, hue should be relatively invariant. This hue-based approach satisfies all the requirements discussed in section 1. The assumption about the uniqueness of colors is reasonable for rescue USVs and might be reasonable for other objects as well.



**FIGURE 6** | Object's blob size may differ significantly in the video frames  $F(t)$  with varying  $t$  because of the oblique view angle.



**FIGURE 7** | The ratio of the minimum inertia to maximum inertia of the object's blob may differ significantly in the video frames  $F(t)$  with varying  $t$  because of the oblique view angle.



**FIGURE 8** | Moving UAS causes the stationary background to appear moving in the video frames.

There are two challenges with using the hue of the blob representing the object for the identification, non-uniformity of the object's color and difficulty to specify particular hue. First, the color of the object might not be uniform. For example, EMILY, a rescue USV, is mostly red but has many non-red parts on its surface. The object's color non-uniformity might cause that the object is represented by multiple blobs of different colors in the video frames. This might be a problem because the smaller color blobs might disappear and appear with changing spatial resolution as the object moves away or toward the UAS. This problem can be alleviated by using Gaussian blur convolution

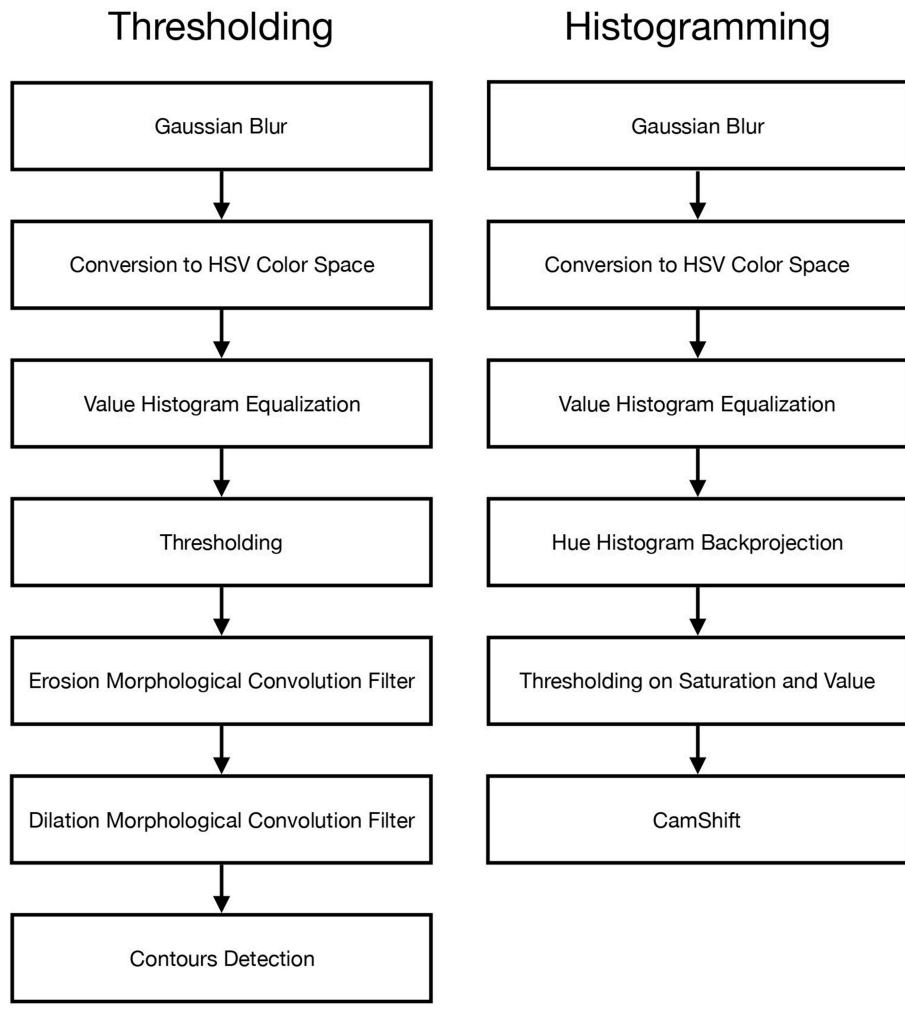
filter to diffuse the color of the smaller color blobs into the bigger color blobs as illustrated in **Figure 9**. The USV on the left side of the figure is composed of multiple color blobs. After the application of Gaussian blur convolution filter, smaller blobs are blended into bigger blobs as can be seen on the right side of the figure. The Gaussian filter can be applied by convolving the original image with a Gaussian kernel as follows:

$$g(i, j) = \sum_{k, l} f(i + k, j + l)G(k, l)$$





**FIGURE 9** | The color of the object might not be uniform which can be alleviated by Gaussian blur convolution filter.



**FIGURE 10** | Flowcharts for the thresholding and histogramming algorithms.

Where  $g$  is the new image,  $f$  is the original image and  $G(k, l)$  is the Gaussian kernel defined as follows:

$$G(k, l) = \frac{1}{2\pi\sigma^2} e^{-\frac{k^2+l^2}{2\sigma^2}}$$

Where  $k$  is the distance from the origin in the horizontal axis,  $l$  is the distance from the origin in the vertical axis, and  $\sigma$  is the standard deviation of the Gaussian distribution.

The second challenge is that specific hue is difficult to define in conventional RGB color space used by regular UAS

visual cameras. The RGB values might be different in all three coordinates and still represent the same hue. It is also problematic to recognize changes in the saturation and intensity of a particular hue. An object with the specific hue under the varying intensity of illumination will have different saturation and value leading to different RGB values. The solution to this problem is the conversion to HSV color space. The HSV color space specifies a color in terms of hue, saturation, and value. In this color space, a hue can be specified independently from its saturation and value, which both change with illumination changes. Given an RGB image, the HSV can be calculated as follows:

$$V = \max(R, G, B)$$

$$S = \begin{cases} \frac{V - \min(R, G, B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$H = \begin{cases} \frac{60(G - B)}{V - \min(R, G, B)} & \text{if } V = R \\ \frac{120 + 60(B - R)}{V - \min(R, G, B)} & \text{if } V = G \\ \frac{240 + 60(R - G)}{V - \min(R, G, B)} & \text{if } V = B \end{cases}$$

Where  $R$  is the red channel,  $G$  is the green channel,  $B$  is the blue channel,  $H$  is hue,  $S$  is saturation, and  $V$  is value. If  $H < 0$ , then  $H$  is adjusted to  $H = H + 360$ .

### 3.1. Two Position Estimation Algorithms

Two algorithms for position estimation are introduced. The first is thresholding based on HSV thresholds and contour detection. The second is histogramming based on hue histogram model of the object, its backprojection, and CamShift tracking algorithm. Input for both algorithms are the video frames  $F(t)$  from UAS and the output is the object's position  $x$ .

While thresholding and CamShift have been used for vision-based object tracking before, the basic sequence of steps has been extended to address the challenges discussed at the beginning of section 3 and requirements discussed in section 1. The basic steps for thresholding application to object tracking are the actual thresholding and contours detection. Some studies also used erosion and dilation after thresholding step (Rosin and Ellis, 1995; Intille et al., 1997; Rosin, 2009; Seenouvang et al., 2016). The basic steps for object tracking based on CamShift are the conversation to HSV color space, hue histogram backprojection, and CamShift itself (Chen et al., 2012; Kamate and Yilmazer, 2015). The additional steps discussed below and their order are proposed by the authors specifically for the problem in hand.

#### 3.1.1. Thresholding

The first, more naive, algorithm is based on thresholding of HSV values. Each input video frame  $F(t)$  from UAS goes through a series of seven steps: blur, HSV conversion, value histogram equalization, thresholding, erosion, dilation, and contours detection. This processed is schematized in the left flowchart in **Figure 10**. The final result of this series is the estimated  $x$  taken as the centroid of the largest area contour found in the last step.

The first and second step is Gaussian blur convolution filter and conversion to HSV color space as discussed above. The third step is histogram equalization on value plane of HSV to increase the global contrast in the video frame. The resulting image can be defined as:

$$g(i, j) = H'(f(i, j))$$

Where  $g$  is the new image,  $f$  is the original image, and  $H'$  is the integral of the normalized histogram  $H$  of the image  $f$ .  $H'$  can be calculated as follows:

$$H'(l) = \sum_{0 \leq k < l} H(k)$$

The fourth step is the application of binary thresholding to identify pixels lying within the specified range of HSV values. The result is a binary threshold map where true signifies that the corresponding pixel is within the specified range of HSV values, and false otherwise. An example of a binary threshold map can be seen on the left side of **Figure 11**. The binary threshold map is created as follows:

$$g(i, j) = T_{min} \leq f(i, j) \leq T_{max}$$

Where  $g$  is the resulting binary threshold map,  $f$  is the original image,  $T_{min}$  is the lower range, and  $T_{max}$  is the upper range.

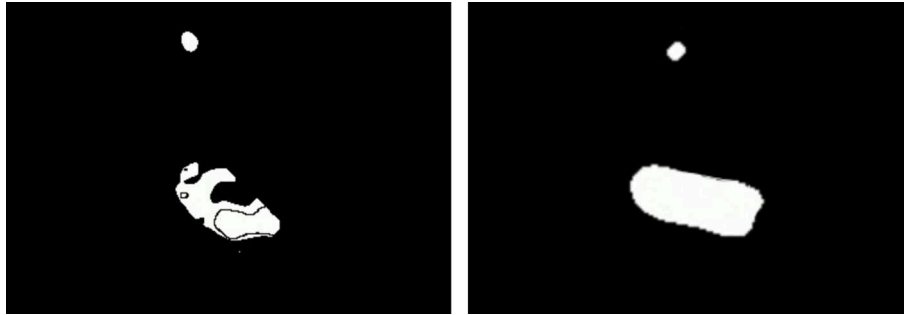
The fifth step is the application of erosion morphological convolution filter to filter out noise. The binary threshold map inherently contains noise. The erosion sets to false a specified number of true pixels that border with false pixels in the binary threshold map by using minimum function in the convolution kernel. It ultimately deletes very small dispersed clusters of true pixels and leaves just larger clusters. The resulting image can be defined as:

$$g(i, j) = \min_{\{(l, k) | \lambda(l, k) \neq 0\}} f(i + k, j + l)$$

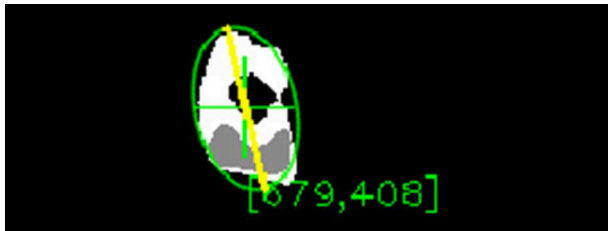
Where  $g$  is the new image,  $f$  is the original image, and  $\lambda$  is the structuring element defining the shape of the neighborhood defined as:

$$\lambda(i, j) = \begin{cases} 1 & \text{if } (i, j) \text{ is in the neighborhood} \\ 0 & \text{otherwise} \end{cases}$$

The sixth step is the application of dilation morphological convolution to amplify the remaining clusters, smooth shapes, and fill-in possible holes in the clusters. The dilation sets to true a specified number of false pixels that border with true pixels in the binary threshold map by using maximum function in the convolution kernel. The holes might occur due to non-uniformity of the object's surface and illumination effects in the environment. An example of a binary threshold map after the application of erosion and dilation can be seen on the right side of **Figure 11**. The resulting image can be defined as:



**FIGURE 11** | Erosion and dilation filters can be applied to the binary threshold map (Left) to filter out noise, smooth shapes, and fill-in holes (Right).



**FIGURE 12** | The backprojection of the object's histogram model and subsequent application of CamShift.

$$g(i, j) = \max_{\{(l, k) | \lambda(l, k) \neq 0\}} f(i + k, j + l)$$

Where  $g$  is the new image,  $f$  is the original image, and  $\lambda$  is the structuring element defining the shape of the neighborhood.

The final, seventh, step is determining contours in the binary threshold map to identify blobs using the method proposed in Suzuki and Be (1985). Contours are boundaries of self-contained clusters of true pixels in the binary threshold map. The contours are found by following borders between true and false pixels in the binary map. Each contour then represents a single blob. In the case there is more than a single blob, the one with the largest area is selected. Then the  $x$  is computed as the centroid of this largest blob.

### 3.1.2. Histogramming

The second algorithm is based on the construction of the hue histogram model and CamShift algorithm (Bradski, 1998). The main idea is to use a hue histogram instead of a simple hue range and then apply CamShift on backprojection of this hue histogram. This algorithm constructs hue histogram at the beginning using user's input and then takes each video frame through a series of six steps: blur, HSV conversion, value histogram equalization, hue histogram backprojection, thresholding on saturation and value, and CamShift algorithm for finding and tracking objects. This process is schematized in the right flowchart in **Figure 10**.

Before the visual position estimation can begin, a hue histogram model of the object must be constructed using a user's

selection of the object. This procedure is done only once in the beginning and not for every video frame. User input is required to select the object in the video frame. A hue histogram model is then constructed for the selected area. The hue histogram divides the entire hue range into a predefined number of non-overlapping bins of the same size. Each bin corresponds to a specific range of hue values. The model is constructed by counting for each bin how many pixels in the selected area are within the hue range of that bin. The histogram is only built for hue because the saturation and value of the object may change with variations in illumination. After the hue histogram is constructed, each input video frame  $F(t)$  from UAS goes through a series of six steps. The final result of this series is the estimated  $x$  taken as the centroid of the area found by CamShift.

The first three steps are the same as for the thresholding algorithm presented in section 3.1.1: Gaussian blur convolution filter, conversion to HSV color space, and value histogram equalization. The fourth step is the calculation of the hue histogram backprojection to identify how well each pixel in  $F(t)$  fits the histogram distribution using the method proposed in Swain and Ballard (1990). The result is a grayscale map of the same dimensions as  $F(t)$ , where each pixel's value signifies how much this pixel's hue is represented in the histogram as shown in **Figure 12**. This is calculated from the histogram by finding how many samples are in the histogram bin corresponding to the pixel's hue. The number of the samples is then normalized by dividing it by the number of samples in the bin with the most samples. The fifth step is the computation of binary threshold on saturation and value in order to filter out the pixels with low saturation and value. This binary threshold is computed on the original video frame independently of the backprojection. Only the pixels lying in the predefined range of saturation and value are set to true in this binary threshold map. This map is then joined with the backprojection map using logical and operation. This sets all the pixels that are false in the binary threshold map to 0 in the backprojection map while keeping everything else intact. This effectively filters out the pixels that do not belong to the specified saturation and value range from the backprojection. The sixth step is the application of CamShift algorithm (Bradski, 1998) for finding and tracking objects to find and track the area of the maximum pixel density. CamShift works on a principle of an imaginary window that iteratively slides toward the weighted

centroid of all the pixels in that window. The size and rotation of the window dynamically adapt in each step. The algorithm eventually converges to a local maximum density area and tracks it as can be seen in **Figure 12**. In the figure, the green cross represents the centroid of the window found by CamShift. Since the value of the pixels in the backprojection correspond to the probability that given pixel belongs to the object, CamShift finds the group of pixels with the highest local probability of belonging to the object. The centroid of the window is then taken as the position of the object,  $x$ .

### 3.2. Shape Analysis for Orientation Estimation

The orientation estimation is based on the analysis of the shape of the object's blob. It is assumed that the major axis of the object's blob indicates the object's blob heading. This assumption is reasonable for USVs as they usually have the major axis going from stern to bow facing the direction of movement. Four techniques for estimation of the principal axis of a blob were examined: line fitting, rectangle fitting, principal component analysis (PCA), and ellipse fitting. The ellipse fitting had the lowest orientation estimation error in preliminary experiments and therefore was the only one used in the final experiments presented in section 5.

The first technique was to fit a line through the blob using linear regression. The regression analysis method of least squares was used. It uses the  $M$ -estimator method to find the best fitting line by iteratively applying the weighted least-squares algorithm. It finds the line minimizing the sum of the squares of residuals,  $\sum_i \frac{r_i^2}{2}$ . A residual  $r_i$  is defined as the distance between the original point  $i$  and the point on the fitted line approximating the original point  $i$ . This technique is more suitable for fitting a line to multiple data points and did not work well for a blob, where all the points are immediately adjacent to each other.

The second technique was to fit a rectangle with the minimum possible area that would still enclose the entire blob. An algorithm proposed in Freeman and Shapira (1975) was used. This algorithm first determines the minimal-perimeter convex polygon enclosing the points (i.e., convex hull). Then, it selects the minimum area rectangle containing this polygon. The idea was that the rectangle's major axis should correspond to the object's major axis. Unfortunately, this technique was not very accurate. The rectangle would sometimes fit in a way that the object would be enclosed diagonally, so the orientation would be estimated incorrectly.

The third technique was principal component analysis taking the principal component as the major axis (Abdi and Williams, 2010). This technique finds the eigenvectors that are the principal components of data points. The eigenvector with the highest eigenvalue represents the major axis of the data. However, this technique suffered from a similar problem as line fitting. It is more suitable for multiple data points, but it did not work well for a single blob where all the points are immediately adjacent to each other.

The fourth technique was to fit an ellipse with the minimum possible area that would still enclose the entire blob. The algebraic

distance algorithm was used (Fitzgibbon and Fisher, 1995). This algorithm finds an ellipse enclosing the points while minimizing the least-squares of distances of the points to the ellipse. The major axis of the ellipse then approximates the major axis of the blob. Unlike the rectangle fitting, ellipse does not have the extra corners preventing the blob to fit diagonally. Therefore, this approach worked better than fitting a rectangle and overall the best from the four examined techniques. An example of the fitted ellipse and the estimated orientation can be seen in **Figure 12**. The green ellipse represents the ellipse fitted to the blob and the yellow line is the major axis of this ellipse approximating the blob orientation.

## 4. IMPLEMENTATION

The proposed approach was implemented to validate the feasibility and to compare the two algorithms for position estimation. Physical platforms used for implementation were a USV (EMILY), a UAS (either DJI Phantom 3 Professional or DJI Inspire 1), or a visual camera as a UAS substitute (GoPro HERO4 Black). The software was implemented on a macOS laptop computer in C++ using OpenCV library. The input was either prerecorded video files or UAS live video stream.

The physical robot platforms used for implementation were Hydronalix Emergency Integrated Lifesaving Lanyard (EMILY) as the USV, and DJI Phantom 3 Professional or DJI Inspire 1 as the UAS, or a GoPro HERO4 Black camera as a substitution for UAS. EMILY is a fast rescue USV covered with a red flotation device. It has maximum speed of  $22 \text{ m s}^{-1}$  and size of 120 cm length and 28 cm beam. It is designed to move through high surf, currents, and swift water. EMILY can be seen in **Figure 1**. DJI Phantom 3 Professional is a small (1.3 kg) inexpensive (\$800) quad-rotor UAS with 23 min flight time. It is equipped with a gimbaled visual camera that provides up to 4K resolution with  $94^\circ$  field of view and 20 mm focal length. The video is streamed over 2.4 GHz downlink to a ground controller in the resolution of 720p at 30 frames per second (fps) with 220 ms latency. The controller can be equipped with optional DJI HDMI Output Module to provide mini-HDMI video output. DJI Inspire 1 is a small (2.9 kg) inexpensive (\$2000) quad-rotor UAS with 18 min flight time. It is also equipped with a gimbaled visual camera that provides up to 4K resolution with  $94^\circ$  field of view and 20 mm focal length. The video is streamed over 2.4 GHz downlink to a ground controller in the resolution of 720p at 30 frames per second (fps) with 220 ms latency. The controller is equipped with mini-HDMI video output by default. DJI Inspire 1 can be seen in **Figure 1**. GoPro HERO4 Black is a non-gimbaled visual camera that provides up to 4 K resolution (at 30 fps) or up to 240 fps (at 720p resolution) with  $65\text{--}123^\circ$  field of view and 17–35 mm focal length. The video recordings are saved on a Micro SD card or can be streamed over 2.4 GHz Wi-Fi.

The software was implemented in C++ using *OpenCV* library and executed on a macOS laptop computer. The algorithms presented in section 3 were implemented using the following OpenCV functions with default parametrization except where specified otherwise: The Gaussian blur was implemented using

GaussianBlur function and kernel size was  $21 \times 21$  px. The conversion from RGB to HSV color space was implemented using cvtColor function. The value histogram equalization used equalizeHist function. For the thresholding algorithm, the thresholds were computed using inRange function, the erosion used erode function with kernel size  $2 \times 2$  px and was applied twice in a row, the dilation used dilate function with kernel size  $16 \times 16$  px and was applied twice in a row, and contour detection used findContours function with simple chain approximation mode (compressing horizontal, vertical, and diagonal segments and leaving only their endpoints). For the histogramming algorithm, the histogram was calculated using calcHist function with histogram size 16, hue histogram backprojection was computed using calcBackProject function, and CamShift was applied using CamShift function with termination criterion either 10 iterations or desired accuracy equal to 1, whichever came first. For orientation estimation algorithms, line fitting used fitLine function implementing the least-squares method as a distance in the M-estimator with the default values for sufficient accuracy, rectangle fitting used minAreaRect function, the principal component analysis was done using PCA class, and ellipse fitting used fitEllipse function. For both thresholding and histogramming algorithms, HSV, and saturation and value thresholds, respectively, were not fixed and were tuned for each trial using implemented graphical user interface (GUI). The GUI was also used to get user input for the construction of the hue histogram in the case of the histogramming algorithm. The code was executed a macOS laptop computer. The hardware configuration of the laptop was the following: processor 2.5 GHz Intel Core i7-4870HQ (Turbo up to 3.7 GHz), memory 16 GB 1.600 MHz DDR3, and no graphical processing unit acceleration.

The input was either video files recorded by UAS/camera onboard or live video feed streamed from UAS. The prerecorded input files were in either MP4 or MOV format. The live video feed stream was used once with DJI Phantom 3 Professional. Since this UAS was not equipped with the optional DJI HDMI Output Module, the video feed had to be streamed from a control tablet (Samsung Galaxy Tab S) connected via USB to the UAS controller. An Android application Screen Stream Mirroring was used to stream the screen of the tablet over the Internet using Real Time Streaming Protocol (RTSP). The stream parameters were set to resolution of 640p at 30 fps, 2048 kbit s<sup>-1</sup> bit rate, and H.264 encoding.

## 5. EXPERIMENTS

The goals of the experiments were to validate the feasibility of the proposed approach to visual pose estimation and to compare the thresholding and histogramming algorithms. Four physical trials were performed to measure the position and orientation estimation error of the thresholding and histogramming algorithms: Trial 1 in an outdoor environment using moving camera and an extremely oblique view angle  $85^\circ$ , Trial 2 in an outdoor environment with moving UAS flying at lower altitude (5–15 m), Trial 3 in an outdoor environment with static UAS flying at higher altitude (30 m), and Trial 4 in indoor

environment testing robustness under different viewpoints and live video stream input. A position and orientation error relative to ground truth were computed by comparing the position and orientation output with manually annotated ground truth position and orientation.

### 5.1. Experimental Methodology

The experiments tested the following hypothesis. *H*: The position and orientation estimation error for histogramming will be lower than the position and orientation estimation error for thresholding when compared to manually annotated ground truth. The materials used in the experiments were a USV (EMILY), a UAS (DJI Phantom 3 Professional, DJI Inspire 1, or GoPro HERO 4 Black as a substitution for UAS), and computing hardware (a laptop computer). Four physical trials were performed and the details about those trials are summarized in **Table 2**. The two metrics were a position and orientation estimation error relative to manually annotated ground truth.

Trial 1 was an outdoor trial in a lake intended to test an extremely oblique view angle ( $85^\circ$ ) and used elevated (2 m) ground-based camera instead of a UAS. The GoPro HERO4 Black camera was handheld and moved to keep the USV in the field of view, therefore the video was shaking and moving. The USV was teleoperated at a speed of up to  $4 \text{ m s}^{-1}$  in a way to frequently change its distance (up to 23 m) and orientation relative to the camera. The trial was performed late in the afternoon (16:30) with the sun being low over the horizon. The camera was facing in the west general direction so the sun was sometimes in the field of view of the camera causing challenges with the white balance of the resulting video. The sky was clear so there were no effects from clouds. The view from the camera looking at the USV during this trial can be seen in **Figures 4–7**.

Trial 2 was an outdoor trial that used a low flying (5–15 m), fast moving (up to  $9 \text{ m s}^{-1}$ ) UAS frequently changing viewpoints of a teleoperated USV. Both the USV and the UAS moved with frequent changes in direction and speed (USV speed was up to  $5 \text{ m s}^{-1}$ ). The UAS was sometimes following the USV and changed viewpoints frequently so the USV was viewed from different sides. The distance between the UAS and USV change frequently being up to 70 m. The view angle from nadir also changed frequently being up to  $78^\circ$ . The relative position of the UAS and the sun was also changing causing the USV to be visible from both the sun-illuminated side and non-illuminated side. The sky was clear so there were no effects from clouds. This trial was done during Fort Bend County, TX, 2016 floods so the USV was operated in flood water. The trial in progress can be seen in **Figure 1**.

Trial 3 was an outdoor trial in a lake that used a high flying (30 m), stationary UAS flying over the shore and looking at the USV at an oblique view angle of up to  $73^\circ$ . The UAS was flying directly above the shore while the USV was operated in GPS waypoints mode at speeds of up to  $2 \text{ m s}^{-1}$  far away in the water causing the distance between the UAS and the USV to be large (up to 100 m) and the view angle to be oblique (up to  $73^\circ$ ). The cloudy weather caused the USV to transition from sun-illuminated areas to shadows and back frequently. The site for the experiment was the same as during Trial 1 with the camera facing in the same

**TABLE 2 |** The trials varied in physical configuration and environmental conditions as can be seen from the parameters of the trials.

	Trial 1	Trial 2	Trial 3	Trial 4
Date	03/28/2016	04/23/2016	05/10/2016	07/05/2016
Time	16:30	15:30	11:00	13:00
Location	Lake Bryan, Bryan, Texas	Fort Bend County, Texas	Lake Bryan, Bryan, Texas	College Station, Texas
Outdoors	✓	✓	✓	✗
Water type	Lake	Flood	Lake	N/A
USV	Hydronalix EMILY	Hydronalix EMILY	Hydronalix EMILY	Hydronalix EMILY
USV control method	Teleoperation	Teleoperation	GPS Waypoints	N/A
USV speed	Up to 4 m s <sup>-1</sup>	Up to 5 m s <sup>-1</sup>	Up to 2 m s <sup>-1</sup>	0 m s <sup>-1</sup>
UAS/Camera	GoPro HERO4 Black	DJI Inspire 1	DJI Phantom 3 Professional	DJI Phantom 3 Professional
Video streaming	✗	✗	✗	✓
UAS speed	0 m s <sup>-1</sup>	Up to 9 m s <sup>-1</sup>	0 m s <sup>-1</sup>	Up to 1 m s <sup>-1</sup>
Altitude (a)	2 m	5 to 15 m	30 m	0 to 1 m
View angle (α)	Up to 85°	Up to 78°	Up to 73°	Up to 90°
Distance UAS to USV	Up to 23 m	Up to 70 m	Up to 100 m	Up to 7 m
Cloud cover	Clear	Clear	Mostly Cloudy	N/A
Wind	ENE 13 km h <sup>-1</sup>	N 7 km h <sup>-1</sup>	S 15 km h <sup>-1</sup>	N/A
Temperature	23 °C	27 °C	28 °C	23 °C
Precipitation	✗	✗	✗	N/A

Green tick indicates the study was outdoors and red cross indicates the trial was not outdoors. For video streaming, it indicates if the trial used video streaming. For precipitation, it indicates if the trial had any precipitation.

general direction, however, this time the trial was done in the late morning (11:00) causing different illumination. The view from the UAS looking at the USV during this trial can be seen in **Figure 2**.

Trial 4 was performed in an indoor environment testing the robustness under different viewpoints and using UAS live video stream as input. There was a constant illumination from ceiling fluorescent lamps. The USV was stationary, and the UAS was carried around at speed up to 1 m s<sup>-1</sup> to observe the USV from different angles and distances (up to 7 m). Because the UAS was carried, it was shaking and moving. The USV was placed on an elevated platform so the altitude difference between the USV and UAS was from 0 to 1 m causing the view angle to be up to 90°. The input video was streamed live at the resolution of 640p at 30 fps with 2048 kbit s<sup>-1</sup> bit rate from the control tablet of the UAS using RTSP (as described in section 4) to test the performance of the algorithms on a live video stream input. Since a USV might look different from different view angles, this trial tested if the algorithms work correctly for all the possible viewpoints.

The methodology for each trial was to capture the USV in operation using UAS/camera, take the video as input, adjust parameters to local conditions, run both algorithms (thresholding and histogramming), and compare the estimated pose with ground truth. In each trial, a UAS/camera was used to capture a video of a USV in operation. The video was either recorded onboard UAS/camera (Trial 1, Trial 2, and Trial 3) or live streamed (Trial 4). Before executing the algorithms, the parameters (HSV thresholds for thresholding, and hue and

saturation thresholds for histogramming) were adjusted to local conditions for each trial. Then, both algorithms for position estimation (thresholding and histogramming) with ellipse fitting algorithm for orientation estimation were executed on the video frames to get the estimated pose. This estimated pose was then compared with manually annotated ground truth pose to compute the error.

The metric used for comparison of the estimated pose with ground truth was composed from two parts, position estimation error and orientation estimation error. Both the position estimation error and the orientation estimation error were computed relative to manually annotated ground truth. The video frames were manually annotated for ground truth USV's blob centroid position  $\mathbf{x}_g(t)$  and ground truth USV's blob orientation (i.e., heading)  $\theta_g(t)$  both in  $\{I\}$  for frame  $F(t), \forall t \in [1..n]$ . The particular algorithm's output was the estimated position of USV's blob centroid  $\mathbf{x}_o(t)$  and the estimated orientation of USV's blob  $\theta_o(t)$  in  $\{I\}$  for frame  $F(t), \forall t \in [1..n]$ . Then for a frame  $F(t)$ , the position estimation error was computed as  $e_x(t) = \|\mathbf{x}_g(t) - \mathbf{x}_o(t)\|_2$  and the orientation estimation error was computed as  $e_\theta(t) = \frac{\pi}{2} - \left| \left| \theta_g(t) - \theta_o(t) \right| - \frac{\pi}{2} \right|$ . The final metric was mean, median, and standard deviation of  $e_x(t)$  and  $e_\theta(t)$  for  $t \in [1..n]$ .

## 5.2. Results

While histogramming had statistically significantly lower position estimation error compared to thresholding for all four trials ( $p$ -value ranged from  $\sim 0$  to  $8.23263 \times 10^{-29}$ ), it

only had statistically significantly lower orientation estimation error for Trial 1 and Trial 3 ( $p$ -values  $3.51852 \times 10^{-39}$  and  $1.32762 \times 10^{-46}$ , respectively). The mean position estimation error for histogramming was quite low ranging from 7 px to 43 px. The mean orientation estimation error, on the other hand, was quite large being in the best case 0.134 rad (histogramming during Trial 1) and in the worst case even 0.480 rad (thresholding during Trial 3). **Table 3** summarizes mean, median, and standard deviation of the position estimation error ( $e_x$ ) and orientation estimation error ( $e_\theta$ ) for both histogramming and thresholding for all four trials.

The main hypothesis,  $H$ , was divided into two hypotheses,  $H^x$  for position estimation error metric ( $e_x$ ) and  $H^\theta$  for orientation estimation error metric ( $e_\theta$ ), and tested for statistical significance using a  $t$ -test. The hypothesis for position estimation error was  $H^x$ : The position estimation error for histogramming ( $e_x^{(H)}$ ) will be lower than the position estimation error for thresholding ( $e_x^{(T)}$ ). The hypothesis for orientation estimation error was  $H^\theta$ : The orientation estimation error for histogramming ( $e_\theta^{(H)}$ ) will be lower than the orientation estimation error for thresholding ( $e_\theta^{(T)}$ ). First, two-sample two-tailed  $f$ -test for equal variances with 5 % significance level indicated unequal variances for all the cases. Then, two-sample one-tailed (left)  $t$ -test assuming unequal variances (using Satterthwaite's approximation) was used to test the two hypothesis for all four trials leading to a total of eight  $t$ -tests. The  $t$ -values were computed as follows:

$$t = \frac{\bar{A} - \bar{B}}{\sqrt{\frac{s_A^2}{N_A} + \frac{s_B^2}{N_B}}}$$

Where  $\bar{A}$  and  $\bar{B}$  are the sample means,  $s_A$  and  $s_B$  are the sample standard deviations, and  $N_A$  and  $N_B$  are the sample sizes. **Table 3** summarizes  $t$ -values and  $p$ -values for each hypothesis  $H^x$  and  $H^\theta$  for all four trials.

The  $t$ -test confirmed the statistical significance of the hypothesis  $H^x$  for position estimation error metric ( $e_x$ ) on 5 % significance level for all four trials meaning histogramming had statistically significantly lower position estimation error ( $e_x$ ) compared to thresholding for all four trials. The null hypothesis was  $H_0^x : e_x^{(H)} = e_x^{(T)}$  and the alternate hypothesis was  $H_a^x : e_x^{(H)} < e_x^{(T)}$ , where  $e_x^{(H)}$  and  $e_x^{(T)}$  were mean position estimation error of histogramming and thresholding, respectively. The resulting  $p$ -values were  $2.24164 \times 10^{-72}$ ,  $8.23263 \times 10^{-29}$ ,  $\sim 0$ ,  $6.02253 \times 10^{-92}$  for Trials 1–4, respectively.

The  $t$ -test confirmed the statistical significance of the hypothesis  $H^\theta$  for orientation estimation error metric ( $e_\theta$ ) on 5 % significance level for Trial 1 and Trial 3, but failed to confirm the statistical significance of the hypothesis for Trial 2 and Trial 4. Therefore, histogramming had statistically significantly lower mean orientation estimation error ( $e_\theta$ ) compared to thresholding for Trial 1 and Trial 3, but the results for Trial 2 and Trial 4 were inconclusive. The null hypothesis was  $H_0^\theta : e_\theta^{(H)} = e_\theta^{(T)}$  and the alternate hypothesis  $H_a^\theta : e_\theta^{(H)} < e_\theta^{(T)}$ , where  $e_\theta^{(H)}$  and  $e_\theta^{(T)}$

were mean orientation estimation error of histogramming and thresholding, respectively. The resulting  $p$ -values were  $3.51852 \times 10^{-39}$ ,  $6.97037 \times 10^{-1}$ ,  $1.32762 \times 10^{-46}$ , and  $9.99999 \times 10^{-1}$  for Trials 1–4, respectively.

The histogramming had a lower mean, median, and standard deviation of the position estimation error ( $e_x$ ) compared with thresholding. The mean of  $e_x$  for histogramming was quite low ranging from 7 to 43 px with median ranging from 6 to 27 px. On the other hand, the mean position estimation error for thresholding was relatively high ranging 15 to 440 px with median ranging from 9 to 51 px. The histogramming had much lower standard deviation of  $e_x$  (4 to 44 px) compared to thresholding (51 to 434 px).

The mean orientation estimation error ( $e_\theta$ ) was relatively high for both algorithms ranging from 0.134 rad for histogramming in Trial 1 to 0.480 rad for thresholding in Trial 3. The median of  $e_\theta$  was ranging from 0.037 rad for thresholding in Trial 4 to 0.271 rad for thresholding in Trial 3. The standard deviation was also relatively large ranging from 0.163 rad for histogramming in Trial 1 to 0.485 rad for thresholding in Trial 1.

The update rate was higher for histogramming than thresholding for all the Trials. The update rate value was dependent on the video resolution. It was highest for Trial 4 (resolution  $1,024 \times 640$ ) being 11 Hz for thresholding and 14 Hz for histogramming. It was lowest for Trial 3 (resolution  $2,132 \times 1,200$ ) being 2 Hz for thresholding and 3 Hz for histogramming.

## 6. DISCUSSION

The histogramming algorithm had statistically significantly lower position estimation error than thresholding when compared with manually annotated ground truth ( $p$ -value ranged from  $\sim 0$  to  $8.23263 \times 10^{-29}$ ), showed the feasibility to variations in environmental conditions and physical settings in three outdoor trials and one indoor trial, and required one less parameter than thresholding. There were, however, three problems. The orientation estimation error was quite large for both algorithms (0.134 to 0.480 rad), both algorithms required manual tuning, and both algorithms were not robust enough to recover from significant changes in illumination conditions. The experiments could have been improved to measure the robustness of the algorithms in addition to their precision by introducing a metric counting the number of times particular algorithm completely lost track of the object. Future work will focus on adapting machine learning methods used for 6D pose estimation to address the issues with the robustness and the need for manual tuning.

### 6.1. Histogramming Algorithm Is Better Than Thresholding for Position Estimation

The histogramming algorithm had a lower position estimation error than thresholding due to its capability to create the object's model in local illumination conditions and to capture more

**TABLE 3 |** The results of the four experiments listing the position estimation error, the orientation estimation error, and the update rate for both thresholding and histogramming algorithms per each trial.

		<b>Trial 1</b>	<b>Trial 2</b>	<b>Trial 3</b>	<b>Trial 4</b>
Location		Lake Bryan, Bryan, Texas	Fort Bend County, Texas	Lake Bryan, Bryan, Texas	Laboratory
Video resolution		1,920 × 1,080	1,920 × 1,080	2,132 × 1,200	1,024 × 640
Observations		480	5,934	6,315	2,249
<b>Thresholds</b>					
	Hue	[0, 10] ∪ [160, 180]	[0, 10] ∪ [160, 180]	[0, 10] ∪ [160, 180]	[0, 10]
Thresholding	Sat.	[58, 255]	[120, 255]	[30, 255]	[167, 255]
	Value	[81, 255]	[100, 255]	[10, 255]	[50, 255]
Histogramming	Sat.	[52, 255]	[120, 255]	[30, 255]	[130, 255]
	Value	[10, 255]	[100, 255]	[10, 255]	[10, 255]
<b>Position estimation error (<math>e_x</math>)</b>					
Thresholding	Mean	440 px	15 px	311 px	100 px
	Median	51 px	9 px	10 px	47 px
	SD	434 px	51 px	385 px	126 px
Histogramming	Mean	15 px	8 px	7 px	43 px
	Median	13 px	6 px	6 px	27 px
	SD	9 px	5 px	4 px	44 px
T-Value for $H^x$		-21.458303	-11.13177414	-63.23634755	-21.01028455
P-Value for $H^x$		2.24164 × 10 <sup>-72</sup>	8.23263 × 10 <sup>-29</sup>	~ 0	6.02253 × 10 <sup>-92</sup>
<b>Orientation estimation error (<math>e_\theta</math>)</b>					
Thresholding	Mean	0.462 rad	0.178 rad	0.480 rad	0.141 rad
	Median	0.243 rad	0.099 rad	0.271 rad	0.037 rad
	SD	0.485 rad	0.260 rad	0.477 rad	0.251 rad
Histogramming	Mean	0.134 rad	0.180 rad	0.365 rad	0.183 rad
	Median	0.076 rad	0.119 rad	0.147 rad	0.044 rad
	SD	0.163 rad	0.202 rad	0.427 rad	0.325 rad
T-value for $H^\theta$		-14.055826	0.515911	-14.345444	4.854452
P-value for $H^\theta$		3.51852 × 10 <sup>-39</sup>	6.97037 × 10 <sup>-1</sup>	1.32762 × 10 <sup>-46</sup>	9.99999 × 10 <sup>-1</sup>
<b>Update rate</b>					
Thresholding		5 Hz	6 Hz	2 Hz	11 Hz
Histogramming		7 Hz	8 Hz	3 Hz	14 Hz

Both the position estimation error and orientation estimation error are listed in terms of mean, median, and standard deviation, and t-value and p-value for hypothesis  $H^x$  and  $H^\theta$ , respectively. The table also contains location of each trial, resolution of the input video, number of frames in the video (observations), and parameters (thresholds) of each algorithm that were used for particular trial (Sat. is an abbreviation for saturation). Green and red are used to compare thresholding and histogramming. Red indicates higher error and green indicates lower error. For p-value, green indicates statistical significance of corresponding hypothesis and red otherwise.

information about the hue distribution of the object. It showed the feasibility to varying configurations and environmental conditions in physical trials. It also required less parameter tuning than thresholding and had a higher update rate.

The histogramming algorithm had statistically significantly lower position estimation error than thresholding when compared with manually annotated ground truth for two reasons: it uses a hue model of the object created in local

illumination conditions and it can capture more information about the hue distribution of the object than thresholding. First, the histogramming algorithm creates a hue histogram model of the object in local illumination conditions. In the case of the thresholding algorithm, the hue threshold has to be set manually and is not computed from the actual appearance of the object in specific illumination conditions. There are four environmental conditions that can change the appearance of hue: time of day,



time of year, weather, and surrounding environment colors (water and the surrounding environment all reflect light onto the object). Second, a hue histogram can capture more information about the hue properties of the object compared to thresholding. The thresholding algorithm only models the hue of the object as a single range of acceptable hue values. On the other hand, the histogramming algorithm works with the entire histogram of hue specifying a hue distribution of the object of interest.

The histogramming algorithm showed feasibility for varying configurations and environmental conditions in physical trials. Despite the variations in trials, the mean position estimation error for histogramming was still relatively low ranging from 7 to 43 px. The three outdoor field trials were performed at three times of year (March, April, May), three times of day (11:00, 15:30, 16:30), in two water types (lake and flood water), using two control modes of the USV (teleoperation and GPS waypoints), with USV speed ranging from 2 to 5  $\text{ms}^{-1}$ , with three UASs/camera platforms (DJI Phantom 3 Professional, DJI Inspire 1, GoPro HERO4 Black), with UAS speeds of up to 9  $\text{m s}^{-1}$ , at altitudes ranging from 2 to 30 m, at view angles ranging from 73 to 85°, at distance between UAS/camera and UAS of up to 100 m, with two different weather conditions (clear and mostly cloudy sky), at three different relative positions of UASs/cameras (an extremely oblique view from static camera yawing to follow the USV, UAS following the USV, stable UAS above shore), and with both the sun-illuminated side and non-illuminated side of the USV being visible. In addition to the three outdoor trials, one indoor trial tested the position estimation from different viewpoints with the view angle of up to 90° from nadir.

The histogramming algorithm had one less parameter than thresholding and therefore required less manual tuning. The thresholding required to manually set all three HSV thresholds. The histogramming algorithm only required to set saturation and value thresholds.

The histogramming algorithm had a higher update rate than thresholding in all the trials. This indicates a higher efficiency of the histogramming algorithm compared to thresholding. Nevertheless, the update rate for both algorithms was enough to enable visual navigation.

## 6.2. Limitations of Proposed Method

A total of three problems with the proposed method were identified. First, the orientation error was quite high due to the oblique view angle. Second, both algorithms required manual tuning before each mission. Third, the algorithms were not robust to significant illumination changes.

The orientation estimation error was quite large for both thresholding and histogramming ranging from 0.134 to 0.480 rad because of perspective distortion caused by the oblique view. Because of the oblique view angle, the shape of the object is not constant and may vary depending on the relative pose of the object to the camera. In the extreme case (object faces the camera and is it is far away), even a cylindrical object might have a circular profile having no significant major axis. This is illustrated in **Figure 2** where a USV's blob is circular making it hard to estimate the orientation by shape analysis. For this reason, an

inverse perspective warping will probably be necessary to reduce the perspective effects introduced with an oblique view angle.

Both thresholding and histogramming required manual tuning for local conditions requiring to adjust three and two parameters, respectively. The thresholding required setting HSV thresholds and histogramming required setting saturation and value thresholds and selecting the object before the visual pose estimation could start. If those parameters were set incorrectly, it would likely lead to a complete failure. For example, if the lower saturation threshold was set higher than the object's saturation, the object would be filtered out. On the other hand, if the lower saturation threshold was set too low, it would fail to filter out parts of the environment with similar hue, but lower saturation.

While histogramming algorithm had relatively low position estimation error (mean ranging from 7 to 43 px), when it loses track of the object due to significant illumination changes, it is very unlikely to recover by itself and human input is required to reselect the object to rebuild the histogram model. The reason is that the algorithm cannot adapt to significant changes in illumination conditions during a single mission. The illumination conditions may change very abruptly with changes in the relative position of all UAS, the object, and sun. For example, the camera may face the sun distorting white balance in the entire view as can be seen in **Figure 5**. This causes the object's color to be very similar to the color of water violating the assumption that the object's color is unique in the environment. Another example is presented in **Figure 4** illustrating how significantly can illumination conditions change during a single run. A method based on machine learning might lead to better robustness to illumination changes and might require less manual tuning before each mission.

## 6.3. Improving Experiments and Machine Learning for Future Work

The experiments could have been improved by introducing an additional metric that would reflect robustness to changing illumination conditions. Additionally, not all the experiments used UAS and not all were outdoors as one trial used a camera instead of UAS and one trial was indoors. The future work will focus on exploring machine learning methods to increase robustness to changing illumination conditions and reduce the need for manual parameter tuning.

An additional metric could have been introduced to test for robustness to changing illumination by reflecting how many times did a particular algorithm completely fail and required a manual change of parameters to recover. Since the parameters of the algorithms were tuned before each trial, the results reflected the precision of the algorithms rather than their general robustness to changing conditions. To test for robustness, each experiment would be started with default parameters and the metric would count how many times those parameters had to be changed to keep the track of the object.

The experiments could have been improved by using a UAS in all the trials and by performing all the trials outdoors. Only two of the four trials were performed outdoors with a UAS and moving USV. Trial 1, while outdoors, used a visual camera instead of

UAS. However, this provided a good challenge since the view angle was extremely oblique (about  $85^\circ$ ) due to very low altitude (2 m). The camera was not stabilized in any way and was hand-held and moving to keep the USV in the field of view. Trial 4 was performed indoors and the USV was not moving. However, this enabled to cover different viewpoints around the USV. The UAS was carried around the USV in the way to cover all the possible view angles.

The aim of future work will be to explore machine learning methods for visual pose estimation with a goal to increase robustness to illumination changes and eliminate the need for manual tuning. There is a large body of work on visual tracking of objects, however, those methods generally do not estimate orientation. There is very active research on visual 6D pose estimation of objects, however, those methods are usually used for robotic manipulation and are done indoors at a very short distance (e.g., objects placed on a table and camera on a manipulator). Most of those methods use depth information from RGB-D cameras, but some use only monocular RGB cameras. The state of the art methods using monocular RGB cameras are Do et al. (2018), Li et al. (2018), Tremblay et al. (2018). The goal of the future work will be to explore if those methods can be adapted to the problem in hand. Machine learning methods might improve robustness while eliminating the need for manual parameter tuning.

## 7. SUMMARY

This article focused on the problem of visual pose estimation of USV from UAS with the following requirements:

- The operating environment is an outdoor water environment.
- Both UAS and USVs are moving (up to 22 and  $10 \text{ m s}^{-1}$ , respectively).
- The UAS is looking at the scene at an oblique view angle (up to  $85^\circ$  from nadir) from large distance (up to 180 m).
- The output has to be full pose (both position and orientation) provided in real-time (more than 5 Hz).

The presented approach was based on the object's hue invariance. Two algorithms for position estimation were presented: thresholding and histogramming. Shape analysis was used for orientation estimation assuming the major axis of the object's blob indicates the object's blob heading. Four techniques for estimation of the principal axis of a blob were examined: line fitting, rectangle fitting, principal component analysis (PCA), and ellipse fitting.

Four physical experiments were performed to validate the feasibility of the proposed approach for visual pose estimation

and to compare the thresholding and histogramming algorithms. A position and orientation error relative to ground truth were computed.

The histogramming algorithm had a lower position estimation error, showed feasibility for varying environmental conditions and physical settings, and required fewer parameters than thresholding. However, three problems were identified. The orientation estimation error was quite large for both algorithms, both algorithms required manual tuning, and both algorithms were not robust enough to recover from significant changes in illumination conditions.

To conclude, the histogramming algorithm with ellipse fitting for orientation estimation should be good enough if enhanced with an inverse perspective warping to reduce the orientation estimation error and if no significant changes in illumination are expected as demonstrated by the low position estimation error (7 to 43 px). If there are significant lighting changes, manual input might be necessary to recreate the hue histogram. To increase the robustness to significant lighting changes without the necessity to recreate the hue histogram, a machine learning approach might ultimately be a better solution.

## AUTHOR CONTRIBUTIONS

JD is responsible for the literature review, approach, implementation, experiments, and writing this manuscript. RM is the principal investigator for the project and provided guidance in all the parts and revised each draft of this manuscript.

## FUNDING

This work was supported in part by National Science Foundation (NSF) grant OISE-1637214 RAPID: Using an Unmanned Aerial Vehicle and Increased Autonomy to Improve an Unmanned Marine Vehicle Lifeguard Assistant Robot.

## ACKNOWLEDGMENTS

This journal article expands upon a conference paper (Dufek and Murphy, 2016) published in 2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR).

The authors would like to thank Xuesu Xiao for assisting with Trial 1, Trial 2, and Trial 3, and Tim Woodbury, Rebecca Schofield, Matt Hegarty, and Jesus Orozco for assisting with Trial 1 and Trial 3. The authors would also like to thank Justin Adams for providing and piloting DJI Inspire 1 during Trial 2.

## REFERENCES

- Abdi, H., and Williams, L. J. (2010). Principal component analysis. *Wiley Interdisc. Rev.* 2, 433–459. doi: 10.1002/wics.101
- Araar, O., Aouf, N., and Vitanov, I. (2017). Vision based autonomous landing of multirotor UAV on moving platform. *J. Intell. Robot. Syst.* 85, 369–384. doi: 10.1007/s10846-016-0399-z
- Aranda, M., López-Nicolás, G., Sagués, C., and Mezouar, Y. (2015). Formation control of mobile robots using multiple aerial cameras. *IEEE Trans. Robot.* 31, 1064–1071. doi: 10.1109/TRO.2015.2452777
- Askar, W., Elmowafy, O., Youssif, A., and Elnashar, G. (2017). "Optimized UAV object tracking framework based on integrated particle filter with ego-motion transformation matrix," in *21st International Conference on Circuits, Systems, Communications and Computers (CSCC)*, Vol.

- 125 of *MATEC Web of Conferences* (Crete: EDP Sciences), 04027. doi: 10.1051/mateconf/201712504027
- Battiatto, S., Cantelli, L., D'Urso, F., Farinella, G. M., Guarnera, L., Guastella, D., et al. (2017). "A system for autonomous landing of a uav on a moving vehicle," in *Image Analysis and Processing - ICIAP 2017*, eds S. Battiatto, G. Gallo, R. Schettini, and F. Stanco (Cham: Springer International Publishing), 129–139.
- Bian, C., Yang, Z., Zhang, T., and Xiong, H. (2016). "Pedestrian tracking from an unmanned aerial vehicle," in *2016 IEEE 13th International Conference on Signal Processing (ICSP)* (Chengdu), 1067–1071. doi: 10.1109/ICSP.2016.7877993
- Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface. *Intel Technol. J.* 2, 214–219.
- Byun, Y., Song, J., Song, W., and Kang, B. (2016). Conceptual study of a smart docking system for VTOL-UAV. *J. Aerospace Eng.* 29:04015053. doi: 10.1061/(ASCE)AS.1943-5525.0000508
- Cabrera-Ponce, A. A., and Martinez-Carranza, J. (2017). "A vision-based approach for autonomous landing," in *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)* (Linköping), 126–131. doi: 10.1109/REDUAS.2017.8101655
- Cantelli, L., Laudani, P., Melita, C. D., and Muscato, G. (2016). *UAV/UGV Cooperation to Improve Navigation Capabilities of a Mobile Robot in Unstructured Environments*. London, UK: World Scientific.
- Chen, P., Zeng, W., Yu, G., and Wang, Y. (2017). Surrogate safety analysis of pedestrian-vehicle conflict at intersections using unmanned aerial vehicle videos. *J. Adv. Transport.* 2017:5202150. doi: 10.1155/2017/5202150
- Chen, S., Guo, S., and Li, Y. (2016). "Real-time tracking a ground moving target in complex indoor and outdoor environments with UAV," in *2016 IEEE International Conference on Information and Automation (ICIA)* (Zhejiang), 362–367.
- Chen, X., Li, X., Wu, H., and Qiu, T. (2012). "Real-time object tracking via CamShift-based robust framework," in *2012 IEEE International Conference on Information Science and Technology* (Hubei), 527–530.
- Cheng, H., Lin, L., Zheng, Z., Guan, Y., and Liu, Z. (2017). "An autonomous vision-based target tracking system for rotorcraft unmanned aerial vehicles," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vancouver, BC), 1732–1738.
- Cho, S., Jung, Y., and Shim, D. H. (2016). "Flight tests of the vision-based target sensing and approaching," in *30th Congress of the International Council of the Aeronautical Sciences (ICAS)* (Daejeon: International Council of the Aeronautical Sciences), 1–7.
- Cognetti, M., Oriolo, G., Peliti, P., Rosa, L., and Stegagno, P. (2014). "Cooperative control of a heterogeneous multi-robot system based on relative localization," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Chicago, IL), 350–356.
- Collins, G., Clause, A., and Twining, D. (2017). "Enabling technologies for autonomous offshore inspections by heterogeneous unmanned teams," in *OCEANS 2017 - Aberdeen* (Aberdeen), 1–5.
- Dixon, W. E., Dawson, D. M., Zergeroglu, E., and Behal, A. (2001). Adaptive tracking control of a wheeled mobile robot via an uncalibrated camera system. *IEEE Trans. Syst. Man Cybern. Part B* 31, 341–352. doi: 10.1109/3477.931519
- Do, T.-T., Pham, T., Cai, M., and Reid, I. (2018). "LieNet: Real-time monocular object instance 6D pose estimation," in *2018 British Machine Vision Conference (BMVC)* (Newcastle upon Tyne).
- Dufek, J., and Murphy, R. (2016). "Visual pose estimation of USV from UAV to assist drowning victims recovery," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (Lausanne), 147–153.
- Dufek, J., and Murphy, R. (2018). "Theoretical limitations of visual navigation of lifesaving USV using small UAS," in *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)* (Philadelphia, PA), 1–7.
- Dufek, J., Xiao, X., and Murphy, R. (2017). "Visual pose stabilization of tethered small unmanned aerial system to assist drowning victim recovery," in *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)* (Shanghai), 116–122.
- Feng, L., Ben, M. C., and Yew, L. K. (2007). "Integration and implementation of a low-cost and vision-based UAV tracking system," in *2007 Chinese Control Conference* (Hunan), 731–736.
- Fitzgibbon, A. W., and Fisher, R. B. (1995). "A buyer's guide to conic fitting," in *Proceedings of the 6th British Conference on Machine Vision (Vol. 2)*, *BMVC '95*, (Birmingham, UK: BMVA Press), 513–522.
- Freeman, H., and Shapira, R. (1975). Determining the minimum-area enclosing rectangle for an arbitrary closed curve. *Commun. ACM* 18, 409–413. doi: 10.1145/360881.360919
- Gao, Q., Zeng, Z. C., and Hu, D. (2014). "Long-term tracking method on ground moving target of UAV," in *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference* (Yantai), 2429–2432.
- Gomez-Avila, J., Lopez-Franco, C., Alanis, A. Y., Arana-Daniel, N., and Lopez-Franco, M. (2018). Ground vehicle tracking with a quadrotor using image based visual servoing. *IFAC-PapersOnLine* 51, 344–349. doi: 10.1016/j.ifacol.2018.07.302
- Harik, E. H. C., Guérin, F., Guinand, F., Brethé, J., and Pelvillain, H. (2015a). "UAV-UGV cooperation for objects transportation in an industrial area," in *2015 IEEE International Conference on Industrial Technology (ICIT)* (Seville), 547–552.
- Harik, E. H. C., Guérin, F., Guinand, F., Brethé, J., and Pelvillain, H. (2016). "Towards an autonomous warehouse inventory scheme," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (Athens), 1–8.
- Harik, E. H. C., Guérin, F., Guinand, F., Brethé, J., Pelvillain, H., and Zentout, A. (2015b). "Vision based target tracking using an unmanned aerial vehicle," in *2015 IEEE International Workshop on Advanced Robotics and its Social Impacts (ARSO)* (Lyon), 1–6.
- Harik, E. H. C., Guérin, F., Guinand, F., Brethé, J.-F., Pelvillain, H., and Parédé, J.-Y. (2017). Fuzzy logic controller for predictive vision-based target tracking with an unmanned aerial vehicle. *Adv. Robot.* 31, 368–381. doi: 10.1080/01691864.2016.1271500
- Harikumar, K., Bera, T., Bardhan, R., and Sundaram, S. (2018). "State estimation of an agile target using discrete sliding mode observer," in *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)* (Thessaloniki), 75–79.
- Hausman, K., Müller, J., Hariharan, A., Ayanian, N., and Sukhatme, G. S. (2015). Cooperative multi-robot control for target tracking with onboard sensing. *Int. J. Robot. Res.* 34, 1660–1677. doi: 10.1177/0278364915602321
- Hausman, K., Müller, J., Hariharan, A., Ayanian, N., and Sukhatme, G. S. (2016). *Cooperative Control for Target Tracking with Onboard Sensing*. Cham: Springer International Publishing.
- Hinas, A., Roberts, J. M., and Gonzalez, F. (2017). Vision-based target finding and inspection of a ground target using a multirotor UAV system. *Sensors* 17:2929. doi: 10.3390/s17122929
- Hoang, T., Bayasgalan, E., Wang, Z., Tschepnakis, G., and Panagou, D. (2017). "Vision-based target tracking and autonomous landing of a quadrotor on a ground vehicle," in *2017 American Control Conference (ACC)* (Seattle, WA), 5580–5585.
- Intille, S. S., Davis, J. W., and Bobick, A. F. (1997). "Real-time closed-world tracking," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (San Juan), 697–703.
- Junaid, A. B., Konoiko, A., Zweiri, Y., Sahinkaya, M. N., and Seneviratne, L. (2017). Autonomous wireless self-charging for multi-rotor unmanned aerial vehicles. *Energies* 10:803. doi: 10.3390/en10060803
- Kamate, S., and Yilmazer, N. (2015). Application of object detection and tracking techniques for unmanned aerial vehicles. *Proc. Comput. Sci.* 61, 436–441. doi: 10.1016/j.procs.2015.09.183
- Karnan, H., Biswas, A., Dhulipala, P. V., Dufek, J., and Murphy, R. (2017). Visual servoing of unmanned surface vehicle from small tethered unmanned aerial vehicle. *arXiv:1710.02932*.
- Kaufmann, S., Kerner, B. S., Rehborn, H., Koller, M., and Klenov, S. L. (2018). Aerial observations of moving synchronized flow patterns in over-saturated city traffic. *Transport. Res. Part C Emerg. Technol.* 86, 393–406. doi: 10.1016/j.trc.2017.11.024
- Kim, D., Yin, K., Liu, M., Lee, S., and Kamat, V. R. (2017). "Feasibility of a drone-based on-site proximity detection in an outdoor construction site," in *ASCE - American Society of Civil Engineers* (Seattle, WA), 392–400.
- Kim, J., Jung, Y., Lee, D., and Shim, D. H. (2016). Landing control on a mobile platform for multi-copters using an omnidirectional image sensor. *J. Intell. Robot. Syst.* 84, 529–541.
- Koo, J., Jung, S., and Myung, H. (2017). "A jellyfish distribution management system using an unmanned aerial vehicle and unmanned surface vehicles," in *2017 IEEE Underwater Technology (UT)* (Busan), 1–5.

- Laiacker, M., Schwarzbach, M., and Kondak, K. (2015). "Automatic aerial retrieval of a mobile robot using optical target tracking and localization," in *2015 IEEE Aerospace Conference* (Big Sky, MT), 1–7.
- Lee, H., Jung, S., and Shim, D. H. (2016). "Vision-based UAV landing on the moving vehicle," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (Washington, DC), 1–7.
- Lee, J. H., Millard, J. D., Lusk, P. C., and Beard, R. W. (2018). "Autonomous target following with monocular camera on UAS using recursive-RANSAC tracker," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)* (Dallas, TX), 1070–1074.
- Li, Y., Wang, G., Ji, X., Xiang, Y., and Fox, D. (2018). "DeepIM: Deep iterative matching for 6D pose estimation," in *Computer Vision – ECCV 2018*, eds V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss (Cham: Springer International Publishing), 695–711.
- Lim, H., and Sinha, S. N. (2015). "Monocular localization of a moving person onboard a quadrotor MAV," in *2015 IEEE International Conference on Robotics and Automation (ICRA)* (Seattle, WA), 2182–2189.
- Liu, S., and Feng, Y. (2018). Real-time fast moving object tracking in severely degraded videos captured by unmanned aerial vehicle. *Int. J. Adv. Robot. Syst.* 15:1729881418759108. doi: 10.1177/1729881418759108
- Liu, Y., Wang, Q., Hu, H., and He, Y. (2018). A novel real-time moving target tracking and path planning system for a quadrotor UAV in unknown unstructured outdoor scenes. *IEEE Trans. Syst. Man Cybern.* 48, 1–11. doi: 10.1109/TSMC.2018.2808471
- Liu, Y., Wang, Q., Zhuang, Y., and Hu, H. (2017). A novel trail detection and scene understanding framework for a quadrotor UAV with monocular vision. *IEEE Sens. J.* 17, 6778–6787. doi: 10.1109/JSEN.2017.2746184
- Ma, Y., Wu, X., Yu, G., Xu, Y., and Wang, Y. (2016). Pedestrian detection and tracking from low-resolution unmanned aerial vehicle thermal imagery. *Sensors* 16:446. doi: 10.3390/s16040446
- Máthé, K., Buşoni, L., Barabás, L., Iuga, C., Miclea, L., and Braband, J. (2016). "Vision-based control of a quadrotor for an object inspection scenario," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (Washington, DC), 849–857.
- Medeiros, F. L. L., Gomes, V. C. F., d. Aquino, M. R. C., Geraldo, D., Honorato, M. E. L., and Dias, L. H. M. (2015). "A computer vision system for guidance of VTOL UAVs autonomous landing," in *2015 Brazilian Conference on Intelligent Systems (BRACIS)* (Natal), 333–338.
- Mendonça, R., Marques, M. M., Marques, F., Lourenço, A., Pinto, E., Santana, P., et al. (2016). "A cooperative multi-robot team for the surveillance of shipwreck survivors at sea," in *OCEANS 2016 MTS/IEEE Monterey* (Monterey, CA), 1–6.
- Monajjemi, M., Mohaimenianpour, S., and Vaughan, R. (2016). "UAV, come to me: End-to-end, multi-scale situated HRI with an uninstrumented human and a distant UAV," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Daejeon), 4410–4417.
- Patrino, C., Nitti, M., Petitti, A., Stella, E., and D'Orazio, T. (2018). A vision-based approach for unmanned aerial vehicle landing. *J. Intell. Robot. Syst.* 92, 1–20. doi: 10.1007/s10846-018-0933-2
- Rao, R., Kumar, V., and Taylor, C. (2003). "Visual servoing of a UGV from a UAV using differential flatness," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)* (Cat. No.03CH37453) (Las Vegas, NV), 743–748.
- Rao, R., Taylor, C., and Kumar, V. (2006). "Experiments in robot control from uncalibrated overhead imagery," in *Experimental Robotics IX*, eds M. H. Ang and O. Khatib (Berlin; Heidelberg: Springer Berlin Heidelberg), 491–500.
- Rao, R., Taylor, C. J., and Kumar, V. (2004). "Calibrating an air-ground control system from motion correspondences," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004* (Washington, DC), II.
- Rao, R. S., Kumar, V., and Taylor, C. J. (2005). "Planning and control of mobile robots in image space from overhead cameras," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation* (Barcelona), 2185–2190.
- Rosa, L., Cognetti, M., Nicastrò, A., Alvarez, P., and Oriolo, G. (2015). Multi-task cooperative control in a heterogeneous ground-air robot team. *IFAC-PapersOnLine* 48, 53–58. doi: 10.1016/j.ifacol.2015.06.463
- Rosin, P. L. (2009). A simple method for detecting salient regions. *Pattern Recogn.* 42, 2363–2371. doi: 10.1016/j.patcog.2009.04.021
- Rosin, P. L., and Ellis, T. (1995). "Image difference threshold strategies and shadow detection," in *Proceedings of the 1995 British Conference on Machine Vision (Vol. 1)* (Surrey: BMVA Press), 347–356.
- Santana, L. V., Brandão, A. S., and Sarcinelli-Filho, M. (2016). "Heterogeneous leader-follower formation based on kinematic models," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (Washington, DC), 342–346.
- Seenoung, N., Watchareeruetai, U., Nuthong, C., Khongsomboon, K., and Ohnishi, N. (2016). "A computer vision based vehicle detection and counting system," in *2016 8th International Conference on Knowledge and Smart Technology (KST)* (Chiangmai), 224–227.
- Siam, M., and ElHelw, M. (2012). "Robust autonomous visual detection and tracking of moving targets in UAV imagery," in *2012 IEEE 11th International Conference on Signal Processing*, Vol. 2 (Beijing), 1060–1066.
- Siam, M., ElSayed, R., and ElHelw, M. (2012). "On-board multiple target detection and tracking on camera-equipped aerial vehicles," in *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)* (Guangzhou), 2399–2405.
- Suzuki, S., and Be, K. A. (1985). Topological structural analysis of digitized binary images by border following. *Comput. Vis. Graph. Image Process.* 30, 32–46. doi: 10.1016/0734-189X(85)90016-7
- Swain, M. J., and Ballard, D. H. (1990). "Indexing via color histograms," in *[1990] Proceedings Third International Conference on Computer Vision*, 390–393.
- Tremblay, J., To, T., Sundaralingam, B., Xiang, Y., Fox, D., and Birchfield, S. (2018). Deep object pose estimation for semantic robotic grasping of household objects. *arXiv [Preprint] arXiv:1809.10790*.
- van Eekeren, A. W. M., van Huis, J. R., Eendebak, P. T., and Baan, J. (2015). "Vehicle tracking in wide area motion imagery from an airborne platform," in *Electro-Optical and Infrared Systems: Technology and Applications XII; and Quantum Information Science and Technology* (Toulouse: International Society for Optics and Photonics, SPIE).
- Wang, J., Sadler, C., Montoya, C. F., and Liu, J. C. L. (2016). "Optimizing ground vehicle tracking using unmanned aerial vehicle and embedded apriltag design," in *2016 International Conference on Computational Science and Computational Intelligence (CSCI)* (Las Vegas, NV), 739–744.
- Wang, K., Phang, S. K., Ke, Y., Chen, X., Gong, K., and Chen, B. M. (2017). "Vision-aided tracking of a moving ground vehicle with a hybrid UAV," in *2017 13th IEEE International Conference on Control Automation (ICCA)* (Ohrid), 28–33.
- Wang, X., Deng, Y., and Duan, H. (2018). Edge-based target detection for unmanned aerial vehicles using competitive bird swarm algorithm. *Aerospace Sci. Technol.* 78, 708–720. doi: 10.1016/j.ast.2018.04.047
- Watanabe, Y., Manecy, A., Amiez, A., Lesire, C., and Grand, C. (2016). "Non-cooperative ground vehicle tracking and interception by multi-RPA collaboration," in *30th Congress of the International Council of the Aeronautical Sciences (ICAS)* (Daejeon).
- Xiao, X., Dufek, J., Woodbury, T., and Murphy, R. (2017). "UAV assisted USV visual navigation for marine mass casualty incident response," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Vancouver, BC), 6105–6110.

**Conflict of Interest Statement:** The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2019 Dufek and Murphy. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.