Received: 19 March 2017 | Accepted: 30 January 2018

DOI: 10.1002/rob.21780

REGULAR ARTICLE

WILEY

Hierarchical fiducial marker design for pose estimation in large-scale scenarios

Hao Wang[1] | Zongying Shi[1] | Geng Lu[1] | Yisheng Zhong[2]

[1]Department of Automation, Tsinghua University, Bejing, People's Republic of China

[2]Department of Automation, and NLIST, Tsinghua University, Beijing, People's Republic of China

Correspondence
Zongying Shi, Department of Automation, Tsinghua University, Bejing, People's Republic of China.
Email: szy@mail.tsinghua.edu.cn

Funding information
National Natural Science Foundation of China, Grant/Award Numbers: 61374034, 61210012

Abstract

In this paper, a hierarchical fiducial marker, called HArCo, is designed to guarantee a smooth pose estimation for large-scale applications. HArCo markers have a visually identifiable structure on multiple scales, so they can be used for consistent pose estimation across a range of altitudes. Experimental results are presented to validate the performance of the proposed methodologies; oscillating platform landing experiments were conducted to show the ability of HArCo to be used in real landing tasks on the deck of a ship.

KEYWORDS

aerial robotics, position estimation

1 | INTRODUCTION

Camera pose estimation has been a key focus of visual-based applications, such as augmented reality (AR) and virtual reality (VR) applications, three-dimensional (3D) reconstruction, path planning, and others (Baratoff, Neubeck, & Regenbrecht, 2002; Kato & Billinghurst, 2002; Wagner, Reitmayr, Mulloni, Drummond, & Schmalstieg, 2010). To obtain camera poses, key points in each frame need to be extracted for calculating image projections. Some approaches, such as structure from motion and simultaneous localization and mapping (SLAM), seek natural features as key points; however, in some situations in which pose estimation is needed with high reliability and accuracy, and it is not inconvenient to preplace markers, fiducial marker methods may be more efficient and effective.

More specifically, fiducial markers have the following three advantages, compared with other visual pose estimation methods, that make them attractive in many practical applications. First, the environment captured by the camera may be highly dynamic, but the corners provided by the markers can always reflect movement of the interested area. Second, preplaced markers can provide sufficient correspondences for pose estimation in low-textured environments. Finally, the lighting conditions have little effect on the performance of the marker methods. Previous research (Garrido-Jurado, Munoz-Salinas, Madrid-Cuevas, & Medina-Carnicer, 2016; Lee, Park, & Sung, 2008) can be consulted for more information and comparisons of pose estimation methods with and without markers. Though many SLAM methods try to identify the real pose of the camera given noisy correspondences, low-textured areas, and low-light conditions (Kim & Kim, 2017; Lee, Lee, & Kim, 2006; Yang, Song, Kaess, & Scherer, 2016), accuracy and reliability are still not sufficient for some extreme situations, such as a solid-colored house, on the deck of a ship, or a daylong mission.

Although marker methods are useful in many practical scenarios, they have some drawbacks. One is the problem of the marker being out of view, which we want to solve in this paper. Traditional marker-based pose estimation systems are effective when the camera is within a limited space in proximity to the preplaced marker, but may lose detection when the camera is too close or too far away from the marker. Furthermore, there is a trade-off between the minimum and maximum distances from the camera to the marker. If the marker is bigger, the maximum distance may be larger, but the minimum distance will also increase. So these methods are not appropriate in some large-scale scenarios.

Out-of-view problems are especially critical in large-scale visual-based pose estimation tasks, such as AR or VR in a large scene or the landing of unmanned aerial vehicles (UAVs) on a moving truck or the deck of a ship. In such scenarios, when the camera moves toward the marker, the image of the marker becomes increasingly larger in the camera's view, and the camera may lose track when it is too close.

This paper presents a hierarchical fiducial marker system named HArCo that offers solutions to the out-of-view problem. The design, generation, and detection methods of HArCo are described in detail. First, we propose a general method for building hierarchical structures in square markers to make them HArCo markers. Then, as hierarchical structures create submarkers, the generation method of the set of markers, which is used to identify the markers, is modified from that of the original marker system. This set of markers is also called a *dictionary*. Finally, we propose a solution to fuse poses estimated from all the markers detected, so as to obtain a unified pose estimation.

The remainder of the paper is organized as follows: Section 2 presents the literature review. In Section 3, the HArCo system is introduced with its design, generation, and pose estimation approaches. The performance of the proposed methods is evaluated in Section 4, and three pose estimation applications, including two quadrotor landing experiments, are also presented. Finally, Section 5 draws some conclusions.

2 | RELATED WORK

Marker systems are different in the following two aspects: the design of the marker system and the strategies to make it robust.

There are various designs for the markers, aimed at enhancing the robustness of a single marker or at the easiness of detecting them. The simplest proposals consist of using some particular points as markers, such as visually detectable balls (Masselli & Zell, 2012), infrared beacons (Dorfmüller & Wirth, 1998; Wenzel, Masselli, & Zell, 2012), or retroreflective spheres (Ribo, Pinz, & Fuhrmann, 2001). Although those specially designed markers are easily separated from the background, their identification is difficult to obtain or it involves a complex process. One straightforward solution to this problem is template-based methods, using either direct template matching (Martínez, Mondragón, Campoy, Sánchez-López, & Olivares-Méndez, 2013) or feature matching (Arora et al., 2013). These methods are often combined with adaptive tracking approaches to handle varying conditions (Nguyen, Worring, & Van Den Boomgaard, 2001). In some specific tasks, unique patterns, which are common in the domain, may be utilized as fiducial markers. For example, the detection and pose estimation using circle markers are studied extensively in helicopter landing tasks (Sanchez-Lopez, Saripalli, Campoy, Pestana, & Fu, 2013; Yang, Scherer, & Zell, 2013). Circles or ellipses can provide enough information for 6-degree-of-freedom (6-DoF) pose estimation, as presented in Li, Wu, Shi, and Zhong (2013).

To make the marker distinctive enough so as not to be confused with others, a commonly adopted strategy is to encode the identifications in the pattern of the markers. Cybercode (Rekimoto & Ayatsuka, 2000) and VisualCode (Rohs & Gfeller, 2004) use squares for marker detection and identification, whereas the ReacTIVision amoeba (Kaltenbrunner & Bencina, 2007) uses circles. Intersense (Naimark & Foxlin, 2002) uses circular borders for high accuracy of the centroid location of the fiducials.

Among the fiducial marker systems proposed in the literature, those based on square markers have gained popularity, for the pose estimation can be obtained from the four corners whereas the information for identification is encoded in the inner region. Although arbitrary patterns can be placed in the inner region, as the popular marker system ARToolkit does (Daniel & Dieter, 2007), using binary codes is a more efficient way, so that pose estimation can be obtained at a frequency close to that of the image acquisition (Fiala, 2010; Garrido-Jurado, Muñoz-Salinas, Madrid-Cuevas, & Marín-Jiménez, 2014).

In a real scenario, the markers may be partially occluded or under bad lighting conditions. The strategies for making a fiducial marker system robust in such situations have been studied extensively. The ARTag system (Fiala, 2004) uses an edge-based border detection method and checksum bits for error correction, so that the marker can be identified when partially occluded. Another way to solve the occlusion problem is using multiple markers. One of the most popular marker systems that does this is the ArUco system (Garrido-Jurado et al., 2014). In the paper, the authors used an array of markers for pose estimation, so that the markers as a whole can endure occlusions of some sort. Another advantage of multiple markers is that each marker can provide an estimation of the real camera pose, and statistical methods can be applied to achieve a more robust output. The ARToolKit Plus (Daniel & Dieter, 2007) uses an automatic thresholding method for marker tracking in various lighting conditions and provides a vignetting compensation feature for correct thresholding of the images.

There are other ways to improve the robustness of the marker system. In Konomura and Hori (2016), the authors build a noncoplanar marker and show that the system can provide a more robust pose estimation. As basic image-processing techniques may suffer from poor image conditions, such as camera defocus and motion blur, Medina-Carnicer (2017) uses a support vector machine for marker identification and presents more robust pose estimation results.

The generation of the dictionary, which is the set of markers to be used, can also influence the robustness of the marker system. On the one hand, the generated markers should be sufficiently different when seen from the four orientations to clearly define a direction for each marker. On the other hand, every two markers in the dictionary should be sufficiently different from each other to reduce the intermarker confusion rate. BinARyID (Fröhlich, Blach, & van Liere, 2007) proposes a method to avoid rotation ambiguities, which is among the earliest work regarding autonomous dictionary generation. In Garrido-Jurado et al. (2016), the authors improve on their previous work on the ArUco system by presenting two mixed integer linear programming methods to either guarantee the optimality of the dictionary or obtain a suboptimal dictionary within restricted time.

Most of the previous work, especially that on AR or VR applications, assumes that the markers are used in a limited space such as indoor environments. However, as fiducial markers are widely used in all kinds of visual tasks, there are applications that require an accurate and robust pose estimation in large-scale scenarios. For example, in a UAV landing task, the pose estimation should always be available throughout the landing procedure, but many existing marker-based systems may fail when the UAV is close to the ground. One study (Chaves, Wolcott, & Eustice, 2015) used separated markers for solving this problem, but this means an adjustment to the landing position to keep the markers in view, and the UAV may lose the target when transitioning between the markers. Another study (Andziulis, Drungilas, Glazko, & Kiseliovas, 2015) built a hierarchical marker whose idea was similar to ours, but the number of markers is limited, and the identification of different markers is difficult.

In this paper, an approach to strengthen AR markers is proposed, so that the out-of-view problem can be fixed and pose estimation can be reliably obtained within a sufficiently large range. The problem of the marker being out of view is solved by adding a hierarchical structure to the AR markers. That is, some parts of a marker are designed to involve

other markers. As a result, pose estimation can be performed within a much greater range if it is designed properly. And even when the marker is occluded, its child markers are still available for pose estimation. Although the idea of implementing hierarchical markers for pose estimation is explained based on ArUco (Garrido-Jurado et al., 2014) in this paper, other AR marker systems that use black and white cells to encode the identifications of the markers can also be used, such as Matrix (Rekimoto, 1998) and IGD (Zhang, Fronz, & Navab, 2002). This is achieved by applying the strategies presented in this paper to designate a white cell as a sublayer marker.

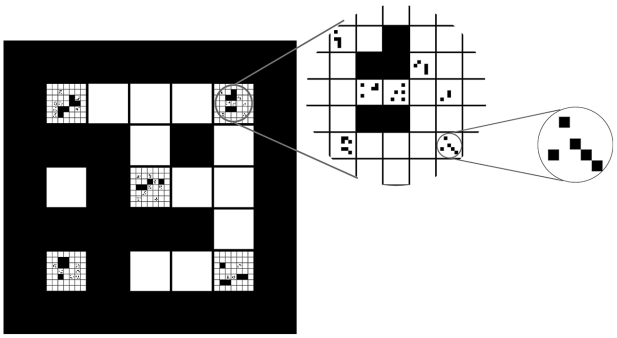

FIGURE 1 An example of a three-layer HArCo marker

3 | HIERARCHICAL FIDUCIAL MARKER SYSTEM

In an AR application, markers are denoted in two ways: appearances and codes. In the camera's view, the marker is a visual planar picture which can be detected by the visual system, while associated with a unique code for recognition purposes. In this paper, the markers described in ArUco (Garrido-Jurado et al., 2014) are adopted as appearances in the hierarchical marker system. The code of each marker records information for pose estimation, which will be discussed in Section 3.3.

Using the definition in ArUco, a marker is an $(n+2)\times(n+2)$ grid of white and black cells; for example, $n=5$ is used in this paper. The inner $n\times n$ cells can be denoted by an $n\times n$ 0-1 matrix by encoding black cells as 0, and white cells as 1. Each row of the matrix is expressed as a word, containing n bits. We represent the marker by a binary code, which is given by reshaping the $n\times n$ 0-1 matrix into an n^2 bits 0-1 code in row-first form, namely:

$$m = (b_0, b_1, \ldots, b_{n^2-1} \mid b_i \in \{0,1\}). \tag{1}$$

For convenience, the corresponding decimal number of m is used for denoting its code, namely, a decimal code.

In translating the binary image into a binary code, the average value of pixels in each cell is used to determine the corresponding bit b_j. So a small change in the number of white pixels does not change the value of the whole cell. This property leads to the basis of our hierarchical marker system. Under this property, a white cell can be replaced by a properly designed sublayer marker without changing the corresponding bit. The problem is that markers should have clear edges for detection, but the black external cells of the markers may blend into the other black cells if they are put directly into the inner area of other markers. The solution is to change the black external cells to be white, and the edges of their borders are drawn in black to make them detectable.

The hierarchical structure in a HArCo marker system is set up as follows: Top-layer markers are the same as those described in ArUco, called parent markers, whereas all other layer markers are child markers. Note that if we have a marker that has more than two layers, all the markers except those on the top layer are child markers. Child markers have all-white external cells, different from parent markers' all-black external cells. The import of child markers has no impact on the original marker system for two reasons. On the one hand, a child marker can be carefully designed to make the average value of the pixels in it close to 1 and will be recognized as a white cell in upper layers. On the other hand, we can check the external cells to separate child markers from the parent markers; thus, there is no impact on the number of parent markers.

3.1 | System overview

An example of a three-layer HArCo marker is shown in Figure 1.

The thin black lines are added to provide edges used for detection of sublayer markers, whose width is manually set. The effect of these black lines will be discussed in Section 3.2. External cells of the parent marker are all black, whereas those of the child markers are all white. In this way, the dictionaries of child and parent markers are separated. Although black cells may also contain child markers, we choose to only add child markers to white cells to make the marker spaces of parent and child markers clearly separated. The inner $n\times n$ cells of a child marker are encoded in the same way as parent markers.

This hierarchical structure of HArCo has a better transition property compared with that of multiple separate markers, as shown in Figure 2. For a successful transition from upper layer markers to sublayer markers, their detectable areas should overlap as much as possible; otherwise, the camera may have a higher probability of losing track during transitioning. As for HArCo markers, the detectable area of sublayer markers is fully contained within that of upper layer markers, so that it is more robust in the transition areas.

Though only two-layer and three-layer examples are shown in this paper, as in Figures 1 and 3, respectively, a marker with any number of layers can be easily produced by overlaying child markers. The maximum number of layers is analyzed as follows.

3.2 | Choosing layers

To better fit a specific application, the number of layers may be adjusted according to different scenarios. First, we should figure out exactly how many layers are needed to support the task. This question will be answered in Section 3.2.1. Although we can start the generation immediately after this number is set, whether the markers can be clearly detected in the real scene is not certain, due to the added black pixels of sublayers. As the number of added black pixels in each layer

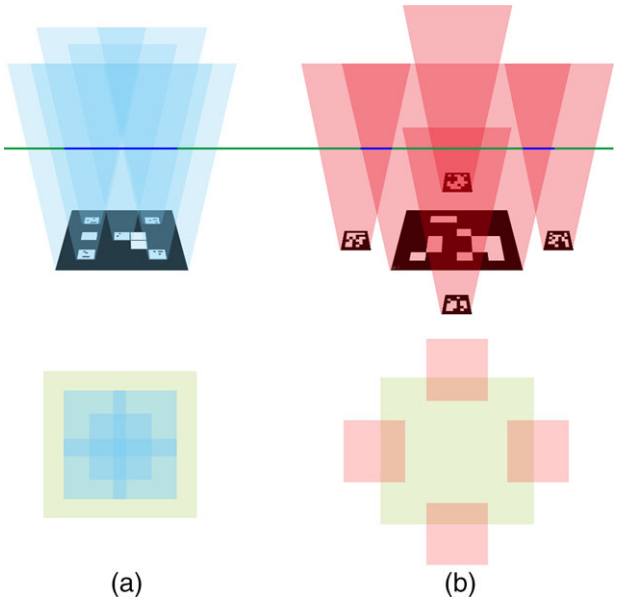

FIGURE 2 The comparison of overlapping detectable areas of HArCo and multiple separate markers

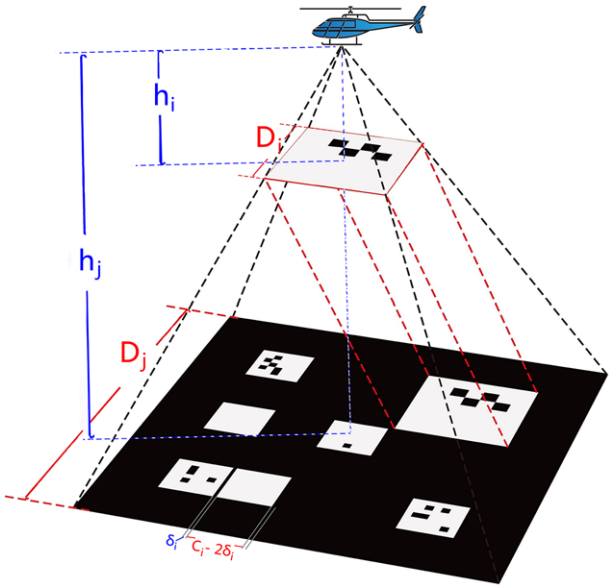

FIGURE 3 The relationship of the distances and the sizes of the markers

can be controlled, the question becomes the maximum number of layers we can get under some settings of the markers. This problem will be discussed in Section 3.2.2. The maximum number of layers should be larger than the number of layers we need, which can be done by adjusting the settings of the markers.

3.2.1 | The number of layers

The number of layers is proportional to the minimum distance from the marker to the camera, but a deeper layer structure may degrade the robustness of detection. So choosing the minimum number of layers that can fit our need may better guarantee the performance of the marker. Here we use the pinhole camera model to illustrate this problem, as shown in Figure 3.

The side length s of the marker in the camera's view can be denoted by

$$s = f\frac{D}{h}, \tag{2}$$

where f is the focal length of the camera, D is the side length of the marker, and h is the distance from the marker to the camera.

The relationship between side lengths and distances of the markers in different layers is expressed by $\frac{D_i}{D_j} = \frac{h_i}{h_j}$, where D_i is the side length of a marker in layer i, and h_i is the distance between the marker in layer i and the camera. The real sizes of markers in different layers change according to the equation $D_i = D_j \prod_{k=j}^{i} \theta_k$, where θ_k is the side length ratio of the markers in layer k to the markers in layer $k-1$. When making the sizes of markers in the camera's view the same, as shown in Figure 3, the distances from the markers to the camera are related through the following equation:

$$h_i = h_j \frac{D_i}{D_j} = h_j \prod_{k=j}^{i} \theta_k. \tag{3}$$

Let h_{min} and h_{max} denote the minimum and maximum distance from the marker to the camera, respectively, and assume $\theta_k = \theta, \forall k > 0$. The number of layers L can be calculated by

$$L = \lceil \log_\theta \frac{h_{\min}}{h_{\max}} \rceil. \tag{4}$$

As an example, set $h_{\min} = 0.1$ m, $h_{\max} = 5$ m, and $\theta = \frac{1}{7}$. The number of layers should be 2.

3.2.2 | Maximum number of layers

When the layers get deeper, other child markers are added, and the number of black pixels in the white cells of a parent marker gets larger. So there is a limit to the number of layers we can get. In this section, the maximum number of layers under a specified set of parameters is analyzed.

Denote the cell length of each child marker in layer i by C_i, the number of bits in each child marker by n^2, and the line width of an added grid in layer i by $2\delta_i$, as shown in Figure 3. Assume that the ratio of 0 bits in a child marker, namely the percentage of black area, is below α, and the ratio of cells that contain child markers is below β, where $\alpha, \beta \in (0, 1)$. α and β define the maximum number of black cells a child marker can have and the maximum number of child markers a marker can have, respectively, which should be strictly observed when generating the HArCo markers.

From the definitions above, a lower bound of white area in a child marker is $(C_i - 2\delta_i)^2(1-\alpha)$, and the lower bound of the ratio of the remaining white area in layer i is given by b_i:

$$b_i = \frac{n^2(1-\alpha)(C_i - 2\delta_i)^2}{n^2 C_i^2} = (1-\alpha)\left(1 - 2\frac{\delta_i}{C_i}\right)^2. \tag{5}$$

When we have an L-layer marker, the lower bound of the ratio of the remaining white area in the parent marker is given by

$$a_L = [(1-\beta) + \beta b_L]a_{L-1} = (1 - \beta(1-b_L))a_{L-1}, \tag{6}$$

where $a_1 = b_1$. Be aware that here we are dealing with child markers; hence a_0 has no definition. Assume that markers have the same $\frac{\delta_i}{C_i} = \gamma$ in each layer, then

$$a_L = (1-\beta+\beta(1-\alpha)(1-2\gamma)^2)a_{L-1} = (1-\beta+\beta(1-\alpha) \times (1-2\gamma)^2)^{L-1} b_1. \quad (7)$$

So the relationship between the maximum number of layers L_u and the ratio of the remaining white area of the parent marker is shown in the next equation:

$$L_u = 1 + \left\lfloor \log_{1-\beta+\beta(1-\alpha)(1-2\gamma)^2} \frac{a_{L_u}}{(1-\alpha)(1-2\gamma)^2} \right\rfloor. \quad (8)$$

As an example, if we set $\alpha = \frac{6}{49}, \beta = \frac{5}{49}, \gamma = \frac{1}{32}$, then $L_u = 1 + \lfloor \log_{0.9819} \frac{a_{L_u}}{0.8228} \rfloor$. If we need $a_{L_u} > 0.7$, the maximum number of layers we can get under those parameters is 8. Note that this is an estimated lower bound. If the task is to better preserve the detectability of the parent markers, one can choose a stricter parameter set.

3.3 | Generation of the marker dictionary

There may be a group of different HArCo markers placed at different positions to build a large positioning system, for example, two markers on the fore and stern of a ship. To better identify the markers, two dictionaries are generated to store the codes of the parent markers and the child markers separately. This is done by the following two principles: First, the markers in each dictionary should be clearly separated, or sufficiently different, from others in the same dictionary, namely the distances between every two markers should be as large as possible. Second, the direction of each marker should be clearly identified, namely the distances between all the poses of the marker should be large enough. The distances for the two principles will be discussed in Section 3.3.2.

The usual way to generate a marker dictionary is to randomly sample some codes and pick out good ones from them, as in ArUco (Garrido-Jurado et al., 2014). However, as the markers in a hierarchical system have some constraints, random generation and selection is not appropriate. First, as not to change the upper layer marker, codes of child markers should have as few 0 bits as possible. Second, because a child marker is a cell in its upper layer, this bit of the upper layer's code must be fixed as 1. Owing to these properties of the marker system, a mutation algorithm is proposed for generating the dictionaries in Section 3.3.3.

3.3.1 | Algorithm overview

We start from two nonempty dictionaries, parent dictionary $\mathcal{D}_F$ and child dictionary $\mathcal{D}_C$:

$$\mathcal{D}_F = \left\{ m_1^1, \ldots, m_{N_1}^1 \right\}, \quad (9)$$

$$\mathcal{D}_C = \left\{ m_1^2, \ldots, m_{N_1}^2, \ldots, m_1^L \ldots, m_{N_L}^L \right\},$$

where L is the number of layers and N_i is the number of markers in layer i. The initial dictionaries are obtained by the following procedure:

- First, set L, α, and β to meet the demand, according to the analysis in Section 3.2 ((1) in Algorithm 1).
- Second, from top to bottom, determine which parts of each layer are child markers according to the demand, and set the associated bits of these areas to 1. The positions are recorded in R_l, where

$$R_l = \left\{ (s,k) \middle| \, b_k \text{ is a child marker}, b_k \in m_s^l, s = 1, \ldots, N_l \right\}, \quad l = 1, \ldots, L-1. \quad (10)$$

The positions of the child markers define the number of markers, N_i. For example, in a marker with $n = 5$, we can set the 0 , 4 , 12 , 20, and 24 bits of the code of the markers in each layer fixed to 1, so $N_i = 5^i$. Other bits in each marker are randomly generated ((2) in Algorithm 1).

To obtain an applicable dictionary, two factors should be considered as previously indicated. First, each marker should be distinguishable when seen from its four directions. Second, every two markers in the dictionary should be as dissimilar as possible. These two factors, in this paper, are denoted by a distance requirement τ, which will be defined in the next section. For each marker that is similar to other markers in the dictionary or itself from the other three directions (has small distances with them), a mutation process will be conducted on it to adjust its code, until all the markers in the dictionary meet the distance requirement. To guarantee the convergence, τ is initially set by hand and reduced by one after a number of unproductive iterations N_{loop}. As the number of markers in $\mathcal{D}$ is fixed to the required number, this algorithm ends when all the markers meet the minimum distance condition τ, or τ is reduced to zero ((3) in Algorithm 1). The pseudocode of the above procedure is shown in Algorithm 1

3.3.2 | Distance calculation

The distance between two markers m_i and m_j is defined by the Hamming distance H, which is the number of different bits in the codes of the two markers. For example, in a 2×2 marker system, the number of different bits in $m_1 = 0101$ and $m_2 = 1111$ is 2, so the Hamming distance of the two markers is 2. Since markers are printed as binary grids of $n \times n$ bits that can be observed under rotation, there are four hamming distances when comparing two markers. So the distance between two markers is defined by

$$\mathrm{D}(m_i, m_j) = \begin{cases} \min\limits_{k\in\{0,1,2,3\}} \mathrm{H}(m_i, \mathrm{R}_k(m_j)), & i \neq j, \\ \min\limits_{k\in\{1,2,3\}} \mathrm{H}(m_i, \mathrm{R}_k(m_j)), & i = j, \end{cases} \quad (11)$$

where R_k is a function, which rotates the marker by $k \times 90$ deg in a clockwise direction. The case $i = j$ is called inner distance, which is used for clearly distinguishing marker orientations. As discussed above, a dictionary of a HArCo system can be divided into two parts: parent marker dictionary $\mathcal{D}_F$ and child marker dictionary $\mathcal{D}_C$, where $\mathcal{D}_F \cup \mathcal{D}_C = \mathcal{D}$ and $\mathcal{D}_F \cap \mathcal{D}_C = 0$. The distance of a marker m_i to the dictionary $\mathcal{D}_k$ containing it is defined as follows ($k = F$ or C):

ALGORITHM 1 Dictionary generation process

```
# (1) Initialization, generate markers in the two dictionaries
D_F ← ∅
D_C ← ∅
L ← L^0    # Set the number of layers
α ← α^0    # Set the maximum number of black cells a child marker can have
β ← β^0    # Set the maximum number of child markers a marker can have
N_1 ← N_1^0 # Set the number of parent markers
# (2) Generate initial dictionaries
for l = 1 to L − 1 do
  R_l ← R_l^0    # Set the positions of child markers in each layer
  for i = 1 to N_l do
    m_i ← random(n^2)   # Randomly generate the bits of each marker
    for (i, k) in R_l do
      m_i(b_k) ← 1    # Fix the bit to 1 if it contains a child marker
    end for
      # Separate markers into two dictionaries
    if l = 1 then
      D_F = D_F ∪ m_i
    else
      D_C = D_C ∪ m_i
    end if
    N_{l+1} = size(R_l)
  end for
end for
  # No constraint for the markers in the last layer
for i = 1 to N_L do
  m_i ← random(n^2)
  D_C = D_C ∪ m_s
end for
# (3) Iteration, change the markers to meet the demands
for j in {F, C} do
  τ ← τ^0    # Initialize the distance requirement
  ρ ← 0
  T ← 0
  while T < τ do
    for m in D_j do
      if D(m, D_j) < τ then
        mutate(m)
      end if
    end for
    T ← min_{m=1,...,size(D_j)} D(m, D_j)
    ρ ← ρ + 1
    if ρ = N_loop then
      τ ← τ − 1    # Decrease target distance
      ρ ← 0
    end if
  end while
end for
```

$$\mathrm{D}(m_i, \mathcal{D}_k) = \min_{m_j \in \mathcal{D}_k, j \neq i} \mathrm{D}(m_i, m_j), \quad m_i \in \mathcal{D}_k,\ k = F \text{ or } C \tag{12}$$

At each iteration of the algorithm, the markers that cannot meet the minimum distance requirement τ mutate to produce new markers. The process of mutation is described in the next section. The symbol τ is the threshold of the distance of a marker to the dictionary.

3.3.3 | Mutation

When a marker's distance to the dictionary is lower than a threshold τ, it is not accepted as a final marker. To change the marker, denoted by $m_i = \{b_p^i\}, p = 0, \ldots, n^2 - 1$, or in the word case $m_i = \{w_q^i\}, q = 0, \ldots, n - 1$, it should mutate bit by bit, in serial fashion. This check is done for all the markers in each iteration.

The idea for mutating bit b_p^i is straightforward. If b_p^i is closer to the average value of $w_q^i s$ in all the other markers, where $b_p^i \in w_q^i$ and the same subscript q means they are at the same position, it should have a larger probability of mutating.

For bit b_p^i in marker m_i to be changed, check the word w_q^i that b_p^i is in. The average value of w_q^i is defined by

$$\text{mean}(w_q^i) = \frac{1}{n}\sum_{k=0}^{n-1} b_k^i, \quad b_k^i \in w_q^i. \tag{13}$$

Then we define the bit rate in the dictionary of w_q^i by

$$\text{rate}(w_q^i) = \begin{cases} \frac{1}{|D_F|} \sum\limits_{w'^i_q \in m_j \in D_F} \text{mean}(w'^i_q), & m_i \in \mathcal{D}_F, \\ \frac{1}{|D_C|} \sum\limits_{w'^i_q \in m_j \in D_C} \text{mean}(w'^i_q), & m_i \in \mathcal{D}_C, \end{cases} \tag{14}$$

where $w_q^i \in m_i$, $|D_k|$ means the size of $D_k (k = F, C)$, and the same subscript q of w and w' means they are at the same position in m_i and m_j. This means that, if $m_i \in \mathcal{D}_F$ or $\mathcal{D}_C$, the mean value of the words at the same position in all the other markers in $\mathcal{D}_F$ or $\mathcal{D}_C$ defines $\text{rate}(w_q^i)$. Then the mutation probability of bit b_p^i is defined by

$$P_m(b_p^i) = \rho \times (1 - |b_p^i - \text{rate}(w_q^i)|), \quad b_p^i \in w_q^i, \tag{15}$$

where ρ is a scale factor.

For a child marker, the average bit rate $(\text{mean}(m_i) = \{\frac{1}{n^2}\sum_{p=0}^{n^2-1} b_p^i \mid b_p^i \in m_i\})$ should always be lower than α, referring to the discussion in Section 3.2. Then the final probability of mutation is defined by

$$P_m^C(b_p^i) = \frac{\alpha - \text{mean}(m_i)}{\alpha} \times P_m(b_p^i), \quad b_p^i \in m_i. \tag{16}$$

That means, if mutating b_p makes the $\text{mean}(m_i)$ closer to α, then it should have a lower probability of mutating.

3.3.4 | Position information of a child marker

Another important attribute that should be recorded when generating a HArCo system is the position information of each marker. That is because child markers should be able to provide the same poses as the parent marker for a smooth and unified estimation. When all the child markers are generated, their positions are given as described below.

To define the coordinates of positions in a parent marker, a grid whose cell size is the same as the last layer's child markers' size is placed on the marker, and the origin of the grid is at the top left point with the x-axis pointing to the right and the y-axis pointing downwards. A child marker's position is its top left corner's coordinates (x, y) on that grid.

3.4 | HArCo for pose estimation

The main detection procedure of HArCo is the same as the ArUco system. However, as multilayer markers exist in one frame, data fusion is crucial to correctly estimate the target's pose. A score on each marker indicating detection quality is calculated, and those with scores lower than a threshold are filtered out. As the four corners of a marker define a homography, the marker's pose relative to the top left corner of itself can be estimated through the two-dimensional (2D)–3D point correspondences in the 2D image to their 3D positions. Setting the 3D coordinate's origin to the top left corner of the parent marker with the z-axis pointing out of the marker plane, the x-axis pointing right, and the y-axis pointing down, the real 3D position of a marker can be determined from the position information recorded. The final estimation result is given by averaging the detection results of all the remaining markers. The detailed procedure is described below (as shown in Figure 4), and the algorithm for detection, recognition, and data fusion is shown in Algorithm 2.

(a) *Extraction of 4-vertex polygons*. Candidate markers are extracted through edge detection (adaptive thresholding) (Gonzalez & Woods, 1992), contour extraction (Suzuki & Be, 1985), and polygon approximation (Douglas & Peucker, 1973). This is because HArCo markers are all quadrilaterals. (1–3)

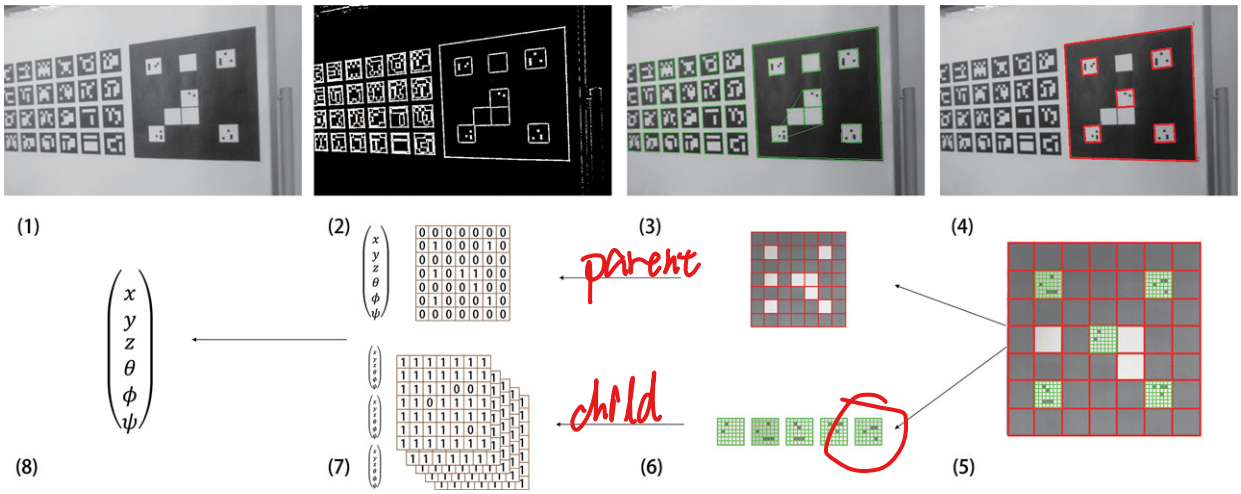

FIGURE 4 Image process for the detection of a two-layer HArCo marker. The original image also has several ArUco markers to show the marker identification feature. (1) Original image. (2) Edge detection result. (3) Marker candidates. (4) Markers in the dictionary. (5) Warped marker and the grid. (6) Markers are separated into parent markers and child markers. (7) Bit assignment for each cell. (8) Pose fusion result

(b) *Warping candidates to squares.* To improve detection accuracy, markers are converted into binary images by thresholding with their means, then warped to a square for further operation. An $(n+2)\times(n+2)$ grid is placed on each candidate square. A first check of the external cells is used for filtering those with errors and separating the remaining ones into parent and child markers. The candidates with all-black external cells are parent markers, whereas ones with all-white external cells are child markers, and others are regarded as wrong. (5)

(c) *Marker identification and error correction.* The average value of pixels in each cell defines a bit, converting the inner $n\times n$ cells to an $n\times n$ matrix. The matrix is reshaped to a binary matrix in row-first form, which provides the marker's binary code. This code is searched in the dictionary to identify the marker. Then the error correlation is applied. If the code is identified as a correct one, the quality score is calculated by counting wrong pixels in the square–that is, warping the square to its template which has the right pixel values and comparing them. Markers with a quality score lower than a threshold κ are filtered out. If the code has no match in the dictionary, we consider the possibility that the detection process fails due to noise in the original image. The nearest marker in the dictionary is picked out, and if the distance between the codes is lower than a and this marker appeared b times in the last c frames, where a, b, and c are manually set parameters, the extracted candidate is identified as this one. (6)

(d) *Pose estimation and data fusion.* As the real positions of the corners all lie on the marker plane, the pose of the marker can be estimated through homography, as in Mondragon, Campoy, Martinez, and Olivares-Méndez (2010). The final pose is given by a weighted average of poses estimated from all the remaining markers and is filtered by a low-pass first-order filter. The weight chosen in this paper is the area of the extracted polygon. (7 and 8)

ALGORITHM 2 Pose estimation process

while camera is ready **do**
  # (1) *Acquire an image from the camera*
  $I \leftarrow I_t$
  $poseSet_{(t)} \leftarrow \emptyset$ # *Reset the group of poses calculated from each identified marker in frame t*
  # (2) *Use edge detection method to produce an edge image*
  $I_e = \text{edgeDetection}(I)$
  $contours = \text{contourExtraction}(I_e)$ # *extract contours*
  # (3) *Fit polygons enclosed in rectangular contours*
  $polygons = \text{fit}(contours)$
  # (4) *Identify markers and correct errors*
  for *polygon* in *polygons* **do**
    # (5) *Warp image in each polygon to square*
    $I_{warp} = \text{warp}(I, polygon)$
    # (6) *Identify code of the square and correct errors*
    $code = \text{extract}(I_{warp})$
    if *code* in $\mathcal{D}$ **then**
      $score = \text{quality}(I_{warp}, code)$ # *Calculate a quality score for the square image*
      if $score < \kappa$ **then**
        continue
      end if
    else
      $nearestCode, distance, appearTimes = \text{nearest}(code, \mathcal{D}, poseSet, c)$
      if $distance < a$ and $appearTimes > b$ **then**
        $code = nearestCode$
      else
        continue
      end if
    end if
    # (7) *Calculate the pose through homography*
    $pose = \text{homography}(I, polygon)$
    $weight = \text{area}(polygon)$ # *Calculate the area of the extracted polygon as the weight for averaging*
    $poseSet_{(t)} \leftarrow (weight, pose)$
  end for
  # (8) *Data fusion*
  $result = \text{weightedAveraging}(poseSet)$ # *Obtain the result by a weighted averaging*
  $result = \text{filter}(result)$ # *Apply the low-pass first-order filter*
end while

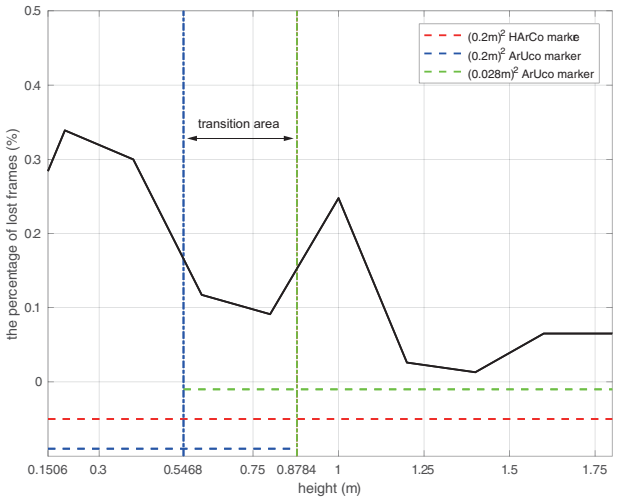

FIGURE 5 The lost count and detectable distance comparison of HArCo and ArUco marker. The vertical lines show the transition area, namely the overlapping area of the effective ranges of the parent and child markers. The figure shows that the lost rate drops within the transition area

As the poses produced by parent and child markers are from the same algorithm and the relative transformations between each other can be precisely measured, the estimations from parent and child markers have the same accuracy. But since the resolution of the camera is limited, the child markers may lead to more noisy pose estimation results compared to the parent marker because they are smaller. In this paper, the proposed fusion method guarantees that smaller markers have smaller influences on the final output, and when the parent marker is out of view, the child markers are large enough for reliable pose estimations so that the transition from parent to child markers is smooth, which can be observed from the experiments described below.

4 | EXPERIMENTAL RESULTS

In this section, the algorithms shown above are tested through several experiments. We demonstrate that the HArCo system can extend the detectable distance of the ArUco system while reducing the lost rate during transition between the layers, and the processing time added for detecting submarkers is short compared to the whole procedure. Then the HArCo system is used in two quadrotor landing tasks, showing its strength for utilization in real scenarios.

4.1 | Effectiveness of HArCo-based pose estimation system

In this experiment, a two-layer HArCo marker whose length is 0.2 m was pasted on the ground, and we manually moved the camera up and down around the transition area 57 times at different constant speeds and different poses, measuring the heights of the camera when the frame was lost into a histogram. The number of lost frames was divided by the total frame count to obtain a percentage of the lost frames. As shown in Figure 5, the percentage of lost frames drops in the transition area (in between the red and the green vertical dashed lines in Figure 5). The maximum ranges of height in which the markers can be fully captured is drawn in dashed lines at the bottom, showing that the HArCo marker as a whole extends any individual marker in it to obtain a greater detectable range of height. In other words, a HArCo marker covers or extends all the detectable ranges of its submarkers.

TABLE 1 Average processing times for each step of the pose estimation method

Preprocessing for each frame	2.8192 ms
Identification for each marker	0.0537 ms
Error correlation and pose fusion for each frame	0.5559 ms
Total processing time for each frame	4.0618 ms

4.2 | Processing time

Processing time is crucial in real-time fiducial applications, so one question is whether the hierarchical structure in the HArCo system will spend too much time on calculation. In this section, we collect several video sequences of a two-layer HArCo marker with other confusing markers to test the processing time. The video sequences, containing about 5,000 frames, were captured using an IDS uEye industrial camera, model number UI-1221LE-C-HQ, with resolution of 750×480 pixels at 30 Hz.

The test was performed on a laptop computer with a 2.9-GHz Intel Core i5 CPU, 8 GB 1867 MHz RAM, and Ubuntu 14.04 as the operating system. The procedure of pose estimation was divided into three phases: preprocessing (including edge detection, polygon fitting, and extraction of candidates); identification of markers in dictionary; error correlation and calculation of the poses. Table 1 summarizes the average processing time for each step. It can be seen that the time added for identifying one marker is much smaller than the whole processing time, indicating that the hierarchical structure in the HArCo system will not have much influence on the entire procedure. In addition, the processing time scales with the number of visually detectable markers, including parent and child ones. Normally, markers in up to three sequential layers, which include less than a hundred markers, are detectable in one frame, whose processing time is acceptable in most applications.

4.3 | Accuracy of the HArCo system

In this section, the HArCo system is compared with ArUco, which is regarded as the baseline, to show its ability to produce accurate pose estimations.

In this experiment, the HArCo marker and the ArUco board were placed as Figure 4 shows, and video sequences with a total of 7,000 frames were collected with a handheld camera making aggressive moves. The settings of the experiment are the same as in the last section. The poses estimated by HArCo is noisier compared with the ArUco board because the poses produced by the child markers are unavoidably noisier as they are smaller and the image resolution is limited. Although the weighted average method is proposed to solve this problem, the pose estimated by the HArCo marker still cannot match that estimated by the averaging of 24 ArUco markers.

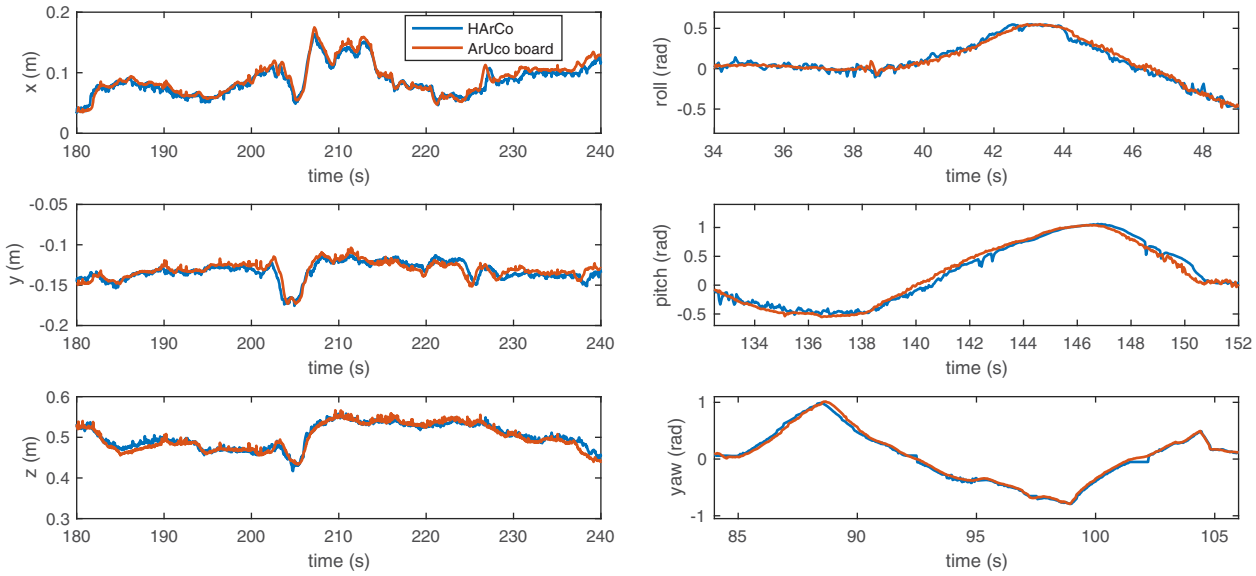

FIGURE 6 The comparison of HArCo and ArUco. The red lines stand for the pose estimation results of ArUco, and the blue lines stand for HArCo. The estimation results of *x*, *y*, and *z* in the same period are drawn on the left, whereas three aggressive moves in roll, pitch, and yaw are drawn on the right

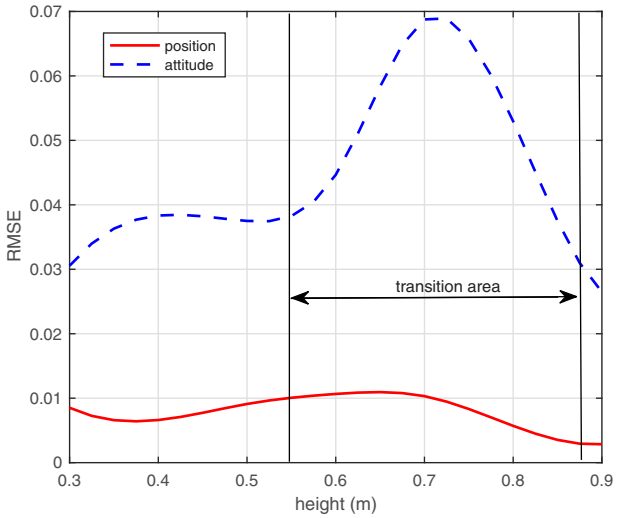

FIGURE 7 The average RMSEs of position and attitude at different heights covering the transition area. The solid line stands for the average RMSE of the three positions, whereas the dashed line stands for the average RMSE of the three angles. Although errors will increase in the transition area, the maximums are still small enough and are acceptable in several visual tasks

The example segments of the comparison are shown in Figure 6. The root mean square errors (RMSEs) of the poses in the example segments are 0.65 cm, 0.54 cm, 1.03 cm, 0.0352 rad, 0.0528 rad, and 0.0147 rad for *x*, *y*, *z*, roll, pitch, and yaw, respectively. The RMSEs at different heights covering the transition area (between 0.5468 m and 0.8784 m) are shown in Figure 7. As can be seen, the errors become larger in the transition area, in which the child markers are small and produce relatively noisier poses, but can be acceptable in several applications. Moreover, as can be seen in the figure, attitudes are more affected by the multilayer property of the estimation.

When the height is at the left of the transition area, the child markers are large enough to produce accurate poses. As the height increases, the child markers become smaller, and the poses from them become relatively unstable, but sufficient for providing reliable pose estimations.

4.4 | Autonomous ground landing based on HArCo

In this experiment, the HArCo pose estimation system was validated in an autonomous landing task of a quadrotor. The quadrotor was based on the mechanical frame of the Flycker MH650, whose diagonal wheelbase of rotors was 650 mm. Four Sunnysky V2814-11 rotors and four APC 1238 propellers combined to give the quadrotor a maximum take-off weight of 5.5 kg. A uEye camera and an Intel Core i5-based microcomputer board were mounted under the quadrotor, both of which were used for visual pose estimation. A digital signal processing (DSP) board was on top of the drone for flight control. Communication between the microcomputer and the DSP board was through a serial port. During the flight, the 3-DoF position information provided by HArCo was transmitted into the DSP board for position control, with no reliance on any other sensors. And the attitude control was by using the onboard inertial measurement unit (IMU). Once the landing procedure started, a series of target heights was generated as the command signals, which started from the current height of the drone and ended 0.2 m below the ground. The extra signals were to make sure that the propellers stopped when the drone landed. The height of the drone was controlled by a proportional-differential (PD) controller. The parameters of this controller were $P = 1$ and $D = 0.8$.

The landing procedure is shown in Figure 8 with the heights shown in Figure 9. In the experiment, the quadrotor was initially at 4 m above the ground, and the large outer marker is in use for pose estimation. Then the drone approaches the ground. At about 0.5 m, the outer marker was occluded, and the system switched to the small markers for

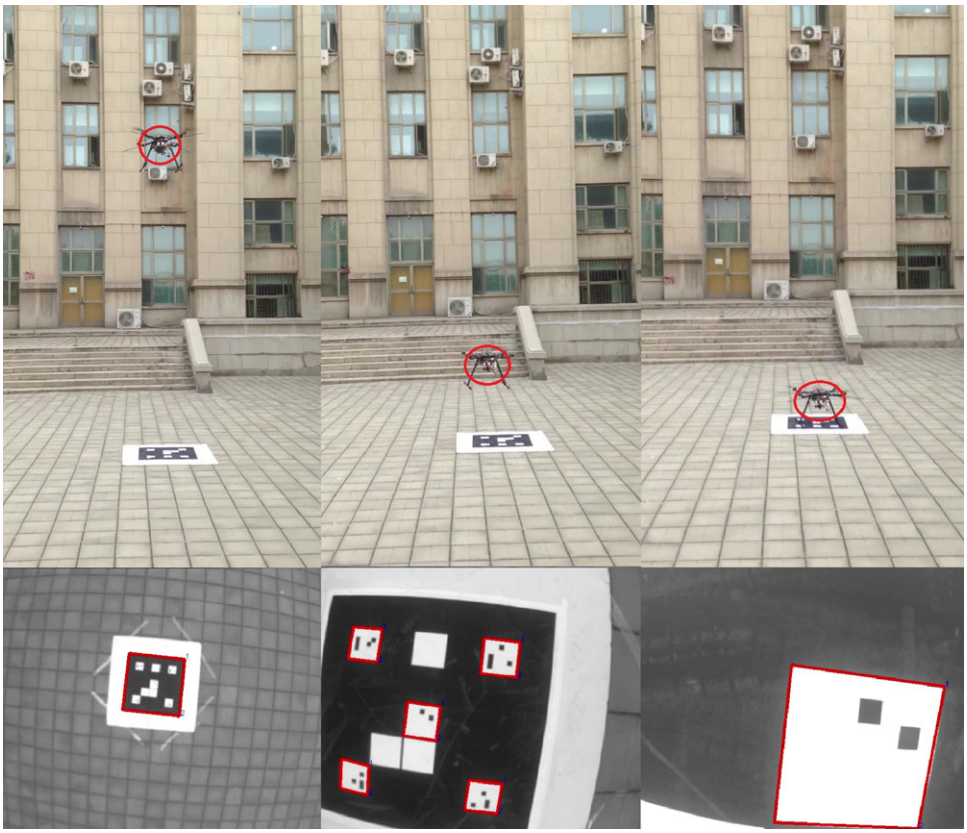

FIGURE 8 The autonomous ground landing procedure. This figure shows the core of the methods proposed, which is the use of the hierarchy of markers to keep track of the landing site at different heights

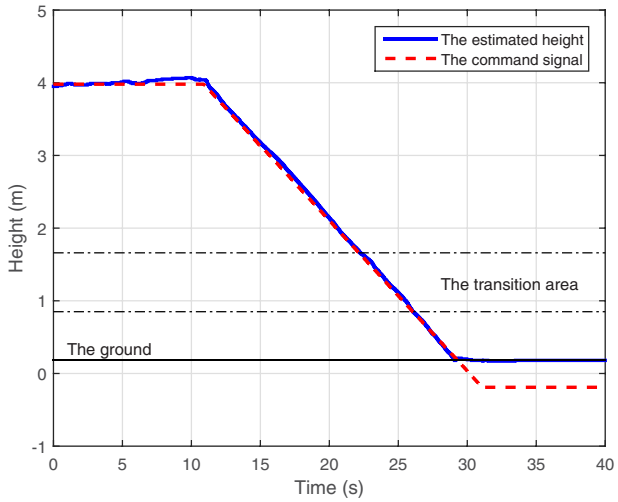

FIGURE 9 The height estimated by HArCo compared with the command signals

pose estimation. The height of the ground in Figure 9 was the height of the camera when the drone had landed, which was about 0.2 m above the actual ground. The marker on the ground was 0.7 m × 0.7 m.

This experiment shows the capability of continuous pose estimation of the HArCo system from high above to the ground. As can be seen from Figure 8, the quadrotor may lose its pose information at the height shown in the middle figure as the parent marker is occluded, but the child markers continue to provide poses. It is the hierarchy of the markers that guarantees the continuous pose information at different heights.

For safety purposes, the drone only flew 4 m above the ground. In fact, the landing height of the system can be extended much farther away, as long as the HArCo marker used is large enough.

4.5 | Pose estimation in a large-scale scenario

This experiment is to show the extensibility of the HArCo system and its accuracy level in terms of the distance from the marker to the camera. The hardware used in this section is the same as that in Section 4.2.

The HArCo system was tested in a large-scale experiment, in which the distance between the camera and the marker varied from 0.15 to 32 m. In the experiment, a three-layer HArCo marker of 1.6 m wide was posted on a wall, and the camera moved horizontally in a triangle, as shown in Figure 10. The three layers function in the ranges of 0.15–0.4, 0.4–4.5, and 4.5–32 m, respectively, and the detectable distance of the entire marker was extended from 4.5–32 to 0.15–32 m as a result.

The poses estimated by the HArCo system were compared with those produced by a differential GPS, and the average RMSE curve of 30 experiments is presented in Figure 11. Although the accuracy level of a differential GPS is not sufficient to grade the HArCo system, one can see from Figure 11 that the accuracy of the HArCo system declines along with the increase in the distance between the camera and the marker. This can also be concluded theoretically from the pinhole

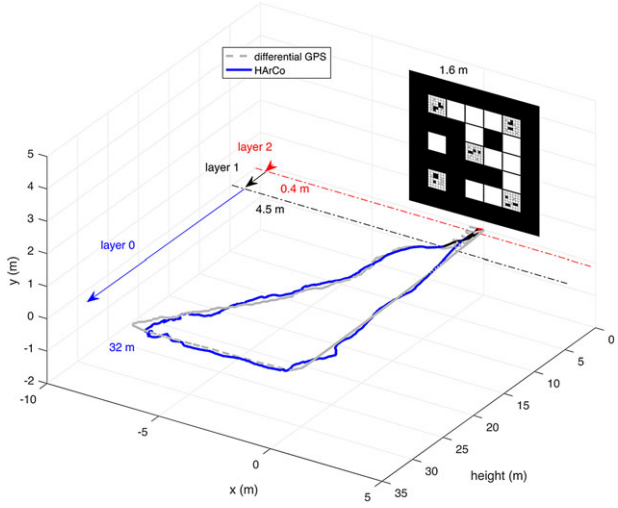

FIGURE 10 A large-scale pose estimation example of HArCo

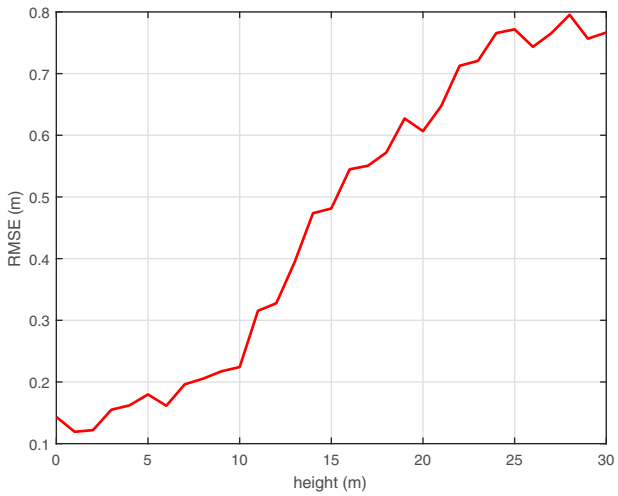

FIGURE 11 The average RMSE curve of the experiment shown in Figure 10 conducted 30 times

projection model (2): Estimating the same length D from different heights h_i and h_j, one has $\frac{s_i}{s_j} = \frac{h_j}{h_i}$. Define the error level at height h_i as

$$e_{h_i} = \frac{\xi_{h_i}}{s_i}, \tag{17}$$

where ξ_{h_i} is the pixel error of s_i at height h_i. Assume that pixel errors are all the same at different heights, namely $\xi_{h_i} = \xi_{h_j}$, then

$$\frac{e_{h_i}}{e_{h_j}} = \frac{h_i}{h_j}. \tag{18}$$

The RMSE error of estimating D at height h_i is proportional to the error level e_{h_i}. Moreover, from Equation 17, the accuracy of a HArCo system can be improved by increasing the side length of the marker or the resolution of the camera (to lower the pixel error).

4.6 | Oscillating platform landing

This experiment provides a practical application of HArCo: that the system can be used to produce continuous pose estimations to support the landing process of a quadrotor on a dynamic platform, for example, a ship's deck.

In this experiment, an oscillating platform is used to simulate a ship's deck. It can oscillate in pitch and roll, and move vertically, thereby simulating the motions of a real ship's deck. The quadrotor hovers according to the relative position and yaw angle against the marker and uses the roll and pitch angles to predict when the platform will be level.

The oscillations of a ship's deck can be modeled by sinusoidal functions in the form (Hess, 2006):

$$y^k = A_0^k + \sum_{i=1}^{M^k} A_i^k \sin\left(\alpha_i^k t + \beta_i^k\right), \tag{19}$$

where t is the time, k can be roll or pitch, M^k is the order of the model, y^k is the angle value, and $A_i^k, \alpha_i^k, \beta_i^k$ are parameters to be estimated. The HArCo-based pose estimation system can provide the relative pose of the oscillating platform with respect to the quadrotor at a sampling frequency of 30 Hz, and the model (19) can be fitted by using nonlinear least square methods. After the model is fitted, it can be used to predict the future level time of the platform.

An example of the modeling results of the oscillating platform during landing is presented in Figure 12. In the top view, the green and blue lines are the raw data of roll and pitch, respectively, and the black and red lines are modeled results. The height of the quadrotor is shown in the bottom view. The first 30 s (before the vertical red line) are used for model estimation, and the samples between the red and blue lines are used for validation. After a successful validation of the estimated model, the system predicts level time periods of the platform, as shown by the black line segments ([65.9 s, 67.5 s], [68.1 s, 70.5 s], [71.1 s, 72.9 s]). The chosen touchdown time was the middle of the third line segment. After that, a linear landing trajectory, which is the same as that in Figure 9, was conducted. The system depended only on IMU for attitude control and the visual pose estimation for position control. The drone successfully landed in the third predicted period when the platform is level, as shown in Figure 12. An example of the experiment is shown in Figure 13.

The method used to predict the level time was tested more than 60 times, and the whole oscillating platform landing task was reproduced 12 times, every time ending in success.

5 | CONCLUSION

A HArCo system was introduced with its application to pose estimation. The hierarchical structure of the markers provides an extension of traditional AR markers, which allows the markers to be detected from a much greater range of height. The choice of layers and the generation of a HArCo system were also discussed. Unlike other dictionary generation methods that eventually produce and add markers to the dictionary, the algorithm in this paper adjusts a randomly generated dictionary to an applicable one that meet the demands. In this way, the cells that contain a child marker can be kept white to help provide the sublayer marker with a border to be detected. The proposed methods were validated by several experiments, including two real landing

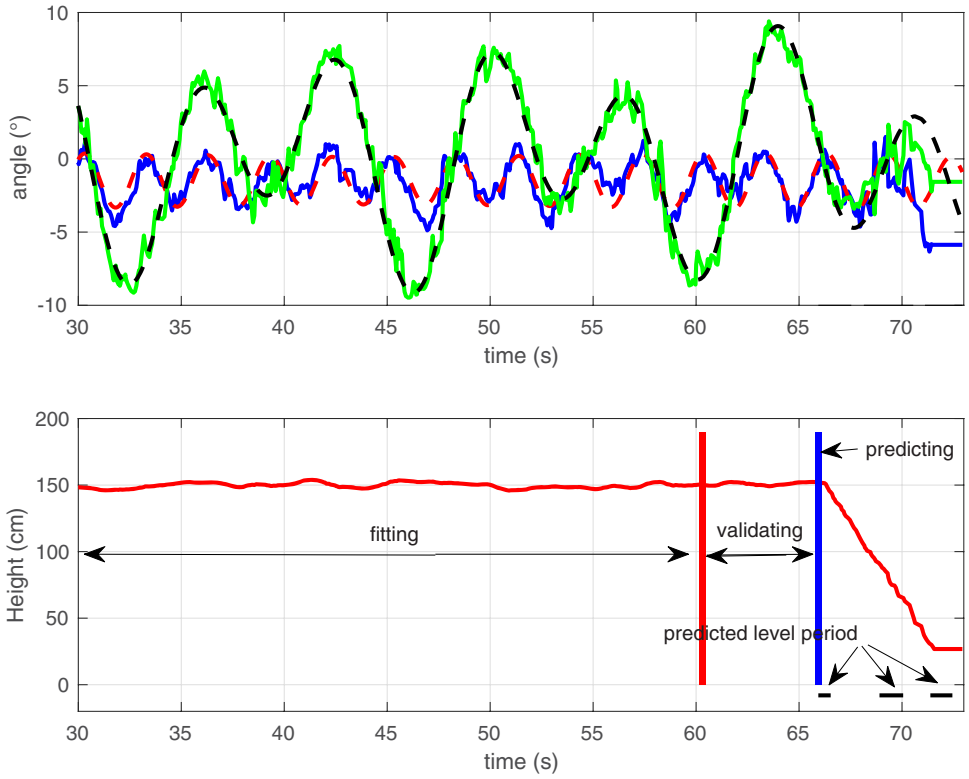

FIGURE 12 The predicted oscillating model and the corresponding touchdown time prediction result. In the top view, the lines with larger amplitudes are roll data, whereas those with lower amplitudes are pitch data. The solid lines are raw data, whereas the dashed lines are fitted curves

FIGURE 13 A quadrotor landing on an oscillating platform

applications to show that the hierarchy of markers can be used to keep track of the landing site.

In general, HArCo has the following three advantages that are quite useful for visual pose estimation. First, HArCo is designed to have good extensibility that can easily increase the detectable range of a single fiducial marker. It is especially useful in large-scale scenarios in which the camera needs to move close to the marker, for example, in quadrotor landing applications. Second, a marker generation method is proposed in this paper, so that HArCo markers can be quickly obtained in large numbers, rather than special designs for each marker. Third, the detectable areas of the parent and child markers overlap well, so that HArCo provides better transition, as discussed in Section 3.1. However, HArCo systems have some drawbacks. First, like most marker systems, the accuracy level of HArCo declines proportionally to the distance between the camera and the marker. Second, HArCo is a frame-wise tracking-by-detection method, which on the one hand has no error accumulation, but on the other hand is sensitive to detection failures.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China under Grants 61374034 and 61210012.

ORCID

Hao Wang http://orcid.org/0000-0003-3123-6043
Zongying Shi http://orcid.org/0000-0002-6805-6858

REFERENCES

Andziulis, A., Drungilas, D., Glazko, V., & Kiseliovas, E. (2015). Resource saving Approach of visual tracking fiducial marker recognition for unmanned aerial vehicle. *Advances in Electrical and Electronic Engineering*, 13(4), 359.

Arora, S., Jain, S., Scherer, S., Nuske, S., Chamberlain, L. & Singh, S. (2013). Infrastructure-free shipdeck tracking for autonomous landing. In *2013 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 323–330). Piscataway, NJ: IEEE.

Baratoff, G., Neubeck, A., & Regenbrecht, H. (2002). Interactive multi-marker calibration for augmented reality applications. In *Proceedings of International Symposium on Mixed and Augmented Reality, 2002 (ISMAR 2002)*. (pp. 107–116). Washington, DC: IEEE Computer Society.

Chaves, S. M., Wolcott, R. W., & Eustice, R. M. (2015). NEEC research: Toward GPS-denied landing of unmanned aerial vehicles on ships at sea. *Naval Engineers Journal*, 127(1), 23–35.

Daniel, W., & Dieter, S. (2007). ARToolKitPlus for pose tracking on mobile devices. Paper presented at *Proceedings of the 12th Computer Vision Winter Workshop*.

Dorfmüller, K., & Wirth, H. (1998). Real-time hand and head tracking for virtual environments using infrared beacons. *Lecture Notes in Computer Science*, 1537, 113–127.

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2), 112–122.

Fiala, M. (2004). *ARTag, An improved marker system based on ARToolkit*. *NRC/erb1111*. Ottawa, Canada: National Research Council Canada.

Fiala, M. (2010). Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7), 1317–1324.

Fröhlich, B., Blach, R., & van Liere, R. (2007). A lightweight ID-based extension for marker tracking systems. Paper presented at *IPT-EGVE Symposium*.

Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6), 2280–2292.

Garrido-Jurado, S., Munoz-Salinas, R., Madrid-Cuevas, F. J., & Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51, 481–491.

Gonzalez, R. C., & Woods, R. E. (1992). Digital image processing. *Prentice Hall International*, 28(4), 484–486.

Hess, R. A. (2006). Simplified technique for modeling piloted rotorcraft operations near ships. *Journal of Guidance, Control, and Dynamics*, 29(6), 1339–1349.

Kaltenbrunner, M., & Bencina, R. (2007). reacTIVision: A computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st International Conference on Tangible and Embedded Interaction* (pp. 69–74). New York, NY: ACM.

Kato, H., & Billinghurst, M. (2002). Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *IEEE and ACM International Workshop on Augmented Reality* (p. 85). New York, NY: ACM.

Kim, J., & Kim, A. (2017). Light condition invariant visual SLAM via entropy based image fusion. In *International Conference on Ubiquitous Robots and Ambient Intelligence* (pp. 529–533). Piscataway, NJ: IEEE.

Konomura, R., & Hori, K. (2016). FPGA-based 6-DoF pose estimation with a monocular camera using non co-planer marker and application on micro quadcopter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4250–4257). Piscataway, NJ: IEEE.

Lee, B. J., Park, J. S., & Sung, M. Y. (2008). Vision-based real-time camera matchmoving with a known marker. *Optical Engineering*, 47(2), 231–242.

Lee, S., Lee, S., & Kim, D. (2006). Recursive unscented Kalman filtering based SLAM using a large number of noisy observations. *International Journal of Control Automation & Systems*, 4(6), 736–747.

Li, F., Wu, X., Shi, Z., & Zhong, Y. (2013). Position and orientation estimation for unmanned helicopter landing based on ellipse feature. In *2013 32nd Chinese Control Conference (CCC)* (pp. 3805–3810). Piscataway, NJ: IEEE.

Martínez, C., Mondragón, I. F., Campoy, P., Sánchez-López, J. L., & Olivares-Méndez, M. A. (2013). A hierarchical tracking strategy for vision-based applications on-board uavs. *Journal of Intelligent & Robotic Systems*, 72(3–4), 517–539.

Masselli, A., & Zell, A. (2012). A novel marker based tracking method for position and attitude control of MAVs. Paper presented at *Proceedings of International Micro Air Vehicle Conference and Flight Competition (IMAV)*.

Medina-Carnicer, R. (2017). Classification of fiducial markers in challenging conditions with SVM. In L. Alexandre, J. Salvador Sánchez, & J. Rodrigues (Eds.), *Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science, Vol. 10255 (pp. 344–352). Cham, Switzerland: Springer International Publishing.

Mondragon, I. F., Campoy, P., Martinez, C., & Olivares-Méndez, M. A. (2010). 3D pose estimation based on planar object tracking for UAVs control. In *2010 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 35–41). Piscataway, NJ: IEEE.

Naimark, L., & Foxlin, E. (2002). Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings of International Symposium on Mixed and Augmented Reality, 2002 (ISMAR 2002)* (pp. 27–36). Piscataway, NJ: IEEE.

Nguyen, H. T., Worring, M., & Van Den Boomgaard, R. (2001). Occlusion robust adaptive template tracking. In *Proceedings of Eighth IEEE International Conference on Computer Vision, 2001 (ICCV 2001)* (Vol. 1, 678–683). Piscataway, NJ: IEEE.

Rekimoto, J. (1998). Matrix: A realtime object identification and registration method for augmented reality. In *Proceedings of Third Asia Pacific Conference on Computer Human Interaction, 1998*. (pp. 63–68). Washington, DC: IEEE Computer Society.

Rekimoto, J., & Ayatsuka, Y. (2000). CyberCode: Designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing Augmented Reality Environments*. (pp. 1–10). New York, NY: ACM.

Ribo, M., Pinz, A., & Fuhrmann, A. L. (2001). A new optical tracking system for virtual and augmented reality applications. In *Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference, 2001 (IMTC 2001)* (Vol. 3, pp. 1932–1936). Piscataway, NJ: IEEE.

Rohs, M., & Gfeller, B. (2004). Using camera-equipped mobile phones for interacting with real-world objects. *Advances in Pervasive Computing*. (pp.265–271).

Sanchez-Lopez, J. L., Saripalli, S., Campoy, P., Pestana, J., & Fu, C. (2013). Toward visual autonomous ship board landing of a VTOL UAV. In *2013 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 779–788). Piscataway, NJ: IEEE.

Suzuki, S., & Be, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision Graphics & Image Processing*, 30(1), 32–46.

Wagner, D., Reitmayr, G., Mulloni, A., Drummond, T., & Schmalstieg, D. (2010). Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization & Computer Graphics*, 16(3), 355.

Wenzel, K. E., Masselli, A., & Zell, A. (2012). Visual tracking and following of a quadrocopter by another quadrocopter. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4993–4998). Piscataway, NJ: IEEE.

Yang, S., Scherer, S. A., & Zell, A. (2013). An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent & Robotic Systems*, 69(1–4), 499–515.

Yang, S., Song, Y., Kaess, M., & Scherer, S. (2016). Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1222–1229). Piscataway, NJ: IEEE.

Zhang, X., Fronz, S., & Navab, N. (2002). Visual marker detection and decoding in AR systems: A comparative study. In *Proceedings of the First International Symposium on Mixed and Augmented Reality*. (pp. 97–106). Washington, DC: IEEE Computer Society.

How to cite this article: Wang H, Shi Z, Lu G, Zhong Y. Hierarchical fiducial marker design for pose estimation in large-scale scenarios. *J Field Robotics*. 2018;1–15. https://doi.org/10.1002/rob.21780