# فرود خودکار پرنده بدون سرنشین با استفاده از بینایی ماشین

**مریم شعاران[1]، محمد فتاحی ثانی[2]**

۱ استادیار، دانشکده مهندسی فناوریهای نوین، دانشگاه تبریز، m.shoaran@tabrizu.ac.ir

۲ دانشجوی کارشناسی ارشد، دانشکده مهندسی برق، دانشگاه تبریز

تاریخ دریافت: ۱۳۹۷/۰۸/۲۹

تاریخ پذیرش: ۱۳۹۷/۱۱/۲۴

## چکیده

یکی از مشکلات پرنده‌های بدون سرنشین خطر فرود ناموفق یا برخورد با زمین است. هدف این مقاله، تخمین دقیق و پیوسته موقعیت پرنده نسبت به نشانگر فرود با استفاده از تصاویر دوربین پرنده و در نهایت فرود خودکار بر روی محل از پیش تعیین شده است. پردازش‌ها به صورت همزمان و با کمترین تاخیر انجام می‌شوند. برای فرود دقیق و کاهش اثرات تاخیرهای موجود در حرکت پرنده الگوریتمی به نام "روش برش حرکتی" ارائه می‌شود که حرکت در نزدیکی نشانگر را به بازه‌های کوچک "حرکت" و "انتظار" تقسیم می‌کند. مدت زمان و سرعت حرکت متناسب با فاصله پرنده از هدف تنظیم می‌شود. نتایج آزمایش‌های تجربی موفقیت عملکرد روش ارائه شده را تایید می‌کند و پرنده می‌تواند با دقت زیر ۳ سانتیمتر و زمان کمتر از ۱۵ ثانیه با موفقیت بر روی هدف فرود آید.

## واژگان کلیدی

پرنده بدون سرنشین، کوادروتور، فرود خودکار، تخمین موقعیت، بینایی ماشین.

### ۱. مقدمه

در سال‌های اخیر تمایل زیادی به تحقیق بر روی پرنده‌های بدون سرنشین در میان محققین هوافضا، کنترل و رباتیک به وجود آمده است. فرود پرنده بدون سرنشین به صورت دستی معمولا با دشواری‌هایی از جمله عدم امکان کاهش منظم ارتفاع و عدم موقعیت‌یابی دقیق همراه است. استفاده از حسگرهای دوربین می‌تواند موقعیت دقیق پرنده را معین کند. بنابراین می‌توان فرآیند فرود را به صورت خودکار پیاده‌سازی کرد. خودکار بودن مراحل فرود باعث کاهش مدت زمان فرود، افزایش اطمینان از فرود ایمن، افزایش عمر مفید پرنده در اثر کاهش ضربات وارده هنگام فرود و کاهش مصرف توان می‌گردد. در [۱] فرود کوادروتور با استفاده از روش‌های کنترل غیرخطی شبیه‌سازی شده است.

محققین دیگری [۲] کوشیدند تا کوادروتور به صورت خودکار در محل مناسب فرود آید و با اتصال به پدهای خاص عمل شارژ شدن انجام پذیرد. در [3] به فرود خودکار کوادروتور با استفاده از حسگر مادون قرمز پرداخته شده است. در [4-5] از کنترل‌کننده تناسبی–انتگرالی–مشتقی (PID) برای تعقیب و فرود خودکار کوادروتور بر روی هدف متحرک استفاده شده است. در [6-7] فرود خودکار کوادروتور در محیط شبیه‌سازی Gazebo آورده شده است. در تحقیق صورت گرفته در دانشگاه برکلی [8] به

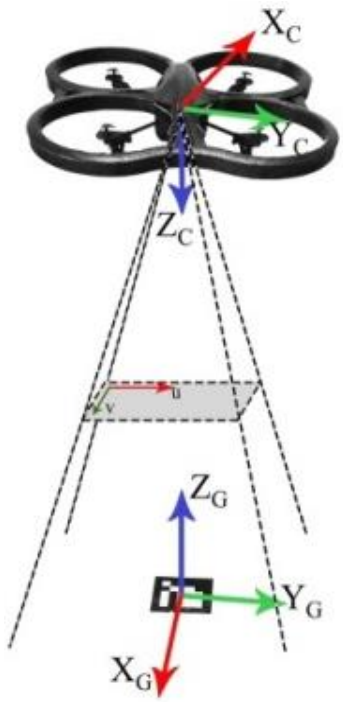
بهینه‌سازی سخت‌افزار و نرم‌افزار پردازش تصویر برای انجام عملیات فرود خودکار پرداخته شده است. در [9] به تعیین ناحیه امن برای فرود خودکار پرداخته شده است.

روشهای ذکر شده در مقالات فوق یا در حد شبیه‌سازی بوده، یا گران‌قیمت هستند و یا بر روی راندمان زمانی و دقت فرود تمرکز کافی ندارند. یکی از مشکلات اصلی که در عملیات ناوبری مبتنی بر تصویر رباتهای پرنده وجود دارد ناپایداری‌های حلقه بسته حاصل از تاخیر ارتباطات رادیویی بین ربات پرنده و ایستگاه مرکزی و تاخیر مربوط به الگوریتم‌های پردازش تصویر نظیر الگوریتم تخمین موقعیت پرنده است. ویژگی متمایز این تحقیق با کارهای مشابه حل مشکل فوق با ارائه الگوریتمی جدید موسوم به "الگوریتم برش حرکتی" است. در این مقاله یک راهکار مناسب برای مساله فرود خودکار کوادروتور بر روی یک نشانگر مشخص ارائه شده و الگوریتم پیشنهادی بر روی کوادروتور AR.Drone 2.0 بطور تجربی پیاده‌سازی می‌گردد. کوادروتور ابتدا با استفاده از حسگر اینرسیایی (IMU) و پردازش داده‌های آن توسط فیلتر کالمن به سمت موقعیت نشانگر فرود حرکت می‌کند. با مشاهده نشانگر پرنده موقعیت خود را نسبت به نشانگر با پردازش تصاویر دریافتی از دوربین تخمین زده و با جهت صحیح بر روی آن فرود می‌آید. در ادامه و در بخش دوم این مقاله به تخمین موقعیت با حسگرهای اینرسیایی و تک‌دوربین می‌پردازیم. در بخش سوم کنترل‌کننده و الگوریتم ارائه شده برای فرود خودکار با ترکیب اطلاعات بدست آمده از حسگرهای اینرسیایی و تک‌دوربین را مورد بحث قرار می‌دهیم. سرانجام در بخش چهارم با بررسی نتایج آزمایش‌های تجربی میزان موفقیت الگوریتم پیشنهادی ارزیابی می‌گردد. نتیجه‌گیری مقاله در بخش پنجم ارائه می‌شود.

## ۲. تخمین موقعیت

برای کنترل موقعیت کوادروتور یک بازخورد بسیار قابل اعتماد مورد نیاز است. اطلاعات بدست آمده از حسگرهای اینرسیایی به علت جمع شدن خطاهای موقعیت قبلی، به مرور زمان از مقدار واقعی دور می‌شود. بنابراین برای دقت بیشتر در نزدیکی نشانگر از اطلاعات دوربین استفاده می‌کنیم.

سه دستگاه مختصات کلی برای توصیف معادلات موجود مورد نیاز است. دستگاه اول دستگاه مختصات مرجع است که با $X_G$، $Y_G$ و $Z_G$ نمایش می‌دهیم و مرکز این دستگاه مختصات را با مرکز نشانگر یکسان در نظر می‌گیریم. دستگاه مختصات دوم دستگاه مختصات پرنده می‌باشد که به علت ثابت بودن دوربین نسبت به بدنه پرنده، مرکز نوری دوربین را مبدا این دستگاه مختصات قرار داده و با $X_C$، $Y_C$ و $Z_C$ نمایش می‌دهیم. در صفحه تصویر نیز دستگاه مختصات پیکسلی $u$ و $v$ را داریم که مبدا آن در گوشه بالای سمت چپ صفحه تصویر است. هر سه دستگاه مختصات در شکل ۱ نمایش داده شده‌اند.




شکل ۱. دستگاه‌های مختصات مرجع، دوربین پرنده و صفحه تصویر

### ۲-۱. تخمین موقعیت با حسگر اینرسیایی

موقعیت و سرعت کوادروتور را می‌توان با استفاده از داده‌های حسگر اینرسیایی به دست آورد [۱۰-۱۱]. همچنین دقت اطلاعات آغشته به نویز حسگر با استفاده از فیلتر کالمن بهبود داده می‌شود [۱۲]. برای این کار $x_k$، بردار حالت در مرحله $k$، به صورت زیر تعریف می‌شود.

$$x_k = [x_{G_{IMU}}, y_{G_{IMU}}, z_{G_{IMU}}, \dot{x}_{G_{IMU}}, \dot{y}_{G_{IMU}}, \dot{z}_{G_{IMU}}]_k^T \quad (۱)$$

که در این رابطه $x_{G_{IMU}}$، $y_{G_{IMU}}$ و $z_{G_{IMU}}$ موقعیت پرنده و $\dot{x}_{G_{IMU}}$، $\dot{y}_{G_{IMU}}$ و $\dot{z}_{G_{IMU}}$ سرعت پرنده نسبت به دستگاه مختصات مرجع هستند. سرعت پرنده با انتگرال‌گیری از شتاب خروجی حسگر اینرسیایی محاسبه می‌شود و با انتگرال‌گیری از سرعت پرنده، مکان پرنده بدست می‌آید [۱۱].

با توجه به مدل فیلتر کالمن، حالت سیستم در مرحله $k$ با استفاده از رابطه زیر بدست می‌آید،

$$x_k = F_k x_{k-1} + B_k u_k + w_k \quad (۲)$$

که در آن $F_k$ ماتریس گذر حالت است، $B_k$ ماتریس مدل ورودی‌های کنترلی است که در اینجا صفر در نظر می‌گیریم و $w_k$ نیز نویز فرآیند است که دارای توزیع گوسین با کواریانس $Q_k$ است.

$$w_k = N(0, Q_k) \quad (۳)$$

ماتریس گذر حالت $F_k$ را به صورت معادله (۴) تعریف می‌کنیم.

$$F_k = \begin{bmatrix} I_3 & dtI_3 \\ O_3 & I_3 \end{bmatrix} \quad (۴)$$

که در آن $dt$ زمان نمونه برداری، $I_m$ ماتریس همانی با ابعاد $m \times m$ و $O_m$ ماتریس صفر با ابعاد $m \times m$ هستند. در مرحله $k$، $z_k$ بردار اندازه‌گیری حالت فیلتر به صورت زیر است.

$$z_k = H_k x_k + v_k \quad (۵)$$

که در آن $H_k$ ماتریس اندازه‌گیری و $v_k$ نویز اندازه‌گیری است که توزیع آن گوسین با کواریانس $R_k$ در نظر گرفته می‌شود.

$$v_k = N(0, R_k) \quad (۶)$$

ماتریس اندازه‌گیری $H_k$ را به شکل زیر در نظر می‌گیریم.

$$H_k = [O_{4\times 2} \quad I_4] \quad (۷)$$

کواریانس نویز فرآیند $Q_k$ و کوواریانس نویز اندازه‌گیری $R_k$ فیلتر کالمن را به صورت زیر در نظر می‌گیریم.

$$Q_k = \begin{bmatrix} 0.1I_3 & O_3 \\ O_3 & 0.3I_3 \end{bmatrix}, \quad R_k = \begin{bmatrix} 0.1I_2 & O_2 \\ O_2 & 0.05I_2 \end{bmatrix} \quad (۸)$$

بردار سرعت دریافتی از پرنده $V_C$، نسبت به دستگاه مختصات پرنده به صورت زیر نمایش داده می‌شود.

$$V_C = [\dot{x}_{C_{IMU}}, \dot{y}_{C_{IMU}}, \dot{z}_{C_{IMU}}]^T \quad (۹)$$

برای انتقال این بردار سرعت به دستگاه مختصات مرجع، باید آن را در یک ماتریس چرخش ضرب کنیم. این ماتریس حاصل‌ضرب سه ماتریس چرخش دیگر به نام‌های $R(\varphi)$، $R(\theta)$ و $R(\psi)$ است که به ترتیب نشان‌دهنده چرخش حول محور طولی، چرخش حول محور عرضی و چرخش حول محور قائم بر دو محور قبل، هستند [۱۳]. بنابراین بردار سرعت $V_G$ نسبت به دستگاه مختصات مرجع به صورت زیر بدست می‌آید.

$$V_G = R(\psi)\, R(\theta) R(\varphi) V_C \quad (۱۰)$$

بردار اندازه‌گیری فیلتر کالمن به صورت زیر تعریف می‌شود.

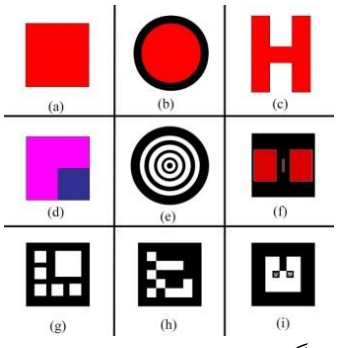
$$m_k = [z_{G_{IMU}}\, V_G]^T = [z_{G_{IMU}}, \dot{x}_{G_{IMU}}, \dot{y}_{G_{IMU}}, \dot{z}_{G_{IMU}}]^T \quad (۱۱)$$

## ۲-۲. تخمین موقعیت مبتنی بر تصویر

در این بخش ابتدا روش تشخیص نشانگر توضیح داده می‌شود. سپس به تخمین موقعیت مبتنی بر تصویر می‌پردازیم.

### ۲-۲-۱. مراحل تشخیص و شناسایی نشانگر

در تحقیقات مختلف از انواع نشانگرها برای فرود خودکار استفاده شده است. در شکل ۲ انواع نشانگرهای استفاده شده برای فرود خودکار آورده شده است. نشانگرهای a و b که به ترتیب در [۱۴] و [۱۵] استفاده شده‌اند ساده هستند، ولی به علت متقارن بودن نمی‌توان مختصات سه بعدی را تخمین زد. البته در نشانگر e [۱۶] با تشخیص دوایر مختلف می‌توانیم ارتفاع را نیز تخمین بزنیم. نشانگر c که در [۵]، [۱۷] و [۱۸] استفاده شده است و همچنین نشانگر f استفاده شده در [۴]، به علت رنگ و شکل خاص نماد خوبی برای پد فرود می‌باشند ولی آنها هم متقارن هستند. نشانگر d استفاده شده در [۱۹]، که در آن از دو رنگ برای تشخیص بهتر استفاده شده است، و یا نشانگر کمی پیچیده تر g [۸]، هردو می‌توانند مختصات سه بعدی را به خوبی تخمین بزنند. اما این طرح‌ها یکتا هستند، یعنی نمی‌توان نشانگرهای مختلفی با این ترکیب داشت. برعکس نشانگر h (نشانگر ArUco [۲۰]) علاوه بر توانایی تخمین کامل مختصات سه بعدی، با کد داخلی آن می‌توان ۱۰۲۴ نشانگر مختلف را تمییز داد. نشانگر i (نشانگر AprilTag) که در [۶] بکار رفته نیز همان مزایا را دارد و علاوه بر آن به علت چند سطحی بودن نشانگر می‌توان از فواصل مختلف آن را تشخیص داد.



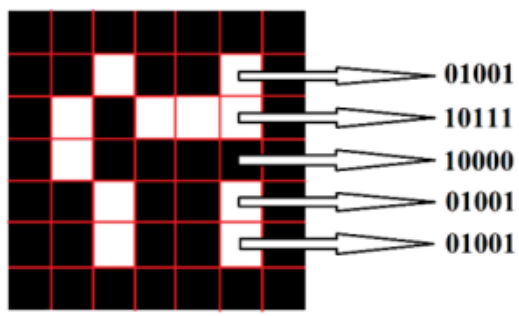

شکل ۲. انواع نشانگرهای استفاده شده به عنوان هدف در مراجع مختلف

ما از نشانگرهای معروف ArUco در این تحقیق استفاده می‌کنیم که اغلب در تحقیقات واقعیت افزوده (Augmented Reality) برای تخمین موقعیت دوربین از آن استفاده می‌شود. در شکل ۳ یک نمونه از نشانگر ۵*۵ استفاده شده نشان داده شده است.

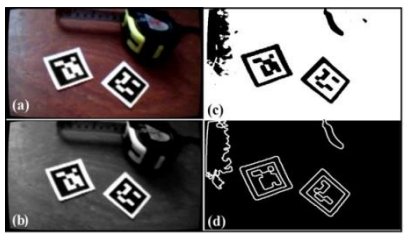
ابتدا تصویر اصلی رنگی با فرمت RGB (شکل ۴-a) را به تصویر با سطوح خاکستری تبدیل می‌کنیم (شکل ۴-b). در مرحله بعد نیاز داریم تا تصویر را باینری کنیم. برای این کار از روش آستانه تطبیقی استفاده می‌کنیم که نتیجه آن تصویری مانند شکل ۴-c می‌باشد. مرحله بعد تشخیص کانتورهای تصویر است. این کار را با استفاده از الگوریتم تعقیب کانتورهای همسایگی ارائه شده توسط Suzuki [۲۱] انجام می‌دهیم. مقدار بازگشتی این الگوریتم لیستی از کانتورهای به هم پیوسته است. در شکل ۴-d کانتورهای تشخیص داده شده نمایش داده شده‌اند. بعد از یافتن کانتورها، اگر چندضلعی تخمین زده شده بیشتر یا کمتر از چهار گوشه داشته باشد قطعا چیزی نیست که ما به دنبال آن می‌گردیم. قبل از اینکه بتوانیم کد نشانگر را تشخیص دهیم باید زاویه‌دار بودن عکس را تصحیح کنیم. ابتدا تبدیل پرسپکتیو مربوطه را با استفاده از چهار جفت نقطه متناظر بدست آورده و این تبدیل را به نشانگر خود اعمال می‌کنیم تا تصحیح شود (شکل ۵). در این مرحله برای عملکرد مطلوبتر بار دیگر تنها ناحیه تشخیص داده شده به عنوان نشانگر را با الگوریتمی دقیق‌تر آستانه‌گیری می‌کنیم. در اینجا از الگوریتم Otsu [۲۲] استفاده می‌کنیم.

یک نشانگر (شکل ۳) به شکل یک ماتریس ۷*۷ باینری دیده می‌شود که بلوک‌های بیرونی با رنگ سیاه پر شده‌اند و یک مربع سیاه را تشکیل می‌دهند که تشخیص آن توسط پردازش تصویر آسان است. مابقی یک ماتریس ۵*۵ است که هر خط آن حاوی ۲ بیت اطلاعات و ۳ بیت برای تشخیص خطا به روشی مشابه روش همینگ است. کل ماتریس یک کد ۱۰ بیتی را تشکیل می‌دهد. در انتها برای اینکه تشخیص دهیم کدام حالت چرخش از چهار حالت ممکن صحیح است، باید خطای کدهای آنها را محاسبه کنیم زیرا جهت صحیح باید دارای خطای صفر باشد.

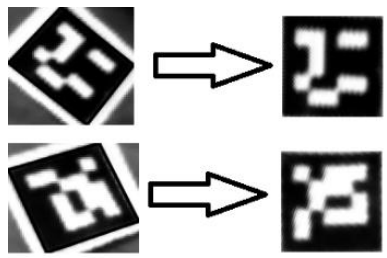
پس از یافتن چرخش صحیح نشانگر، گوشه‌های نشانگر را دوباره با دقت زیر پیکسل پیدا می‌کنیم. در شکل ۶ نشانگرهای یافته شده را مشاهده می‌کنیم.




شکل ۳. تصویر نشانگر ۵*۵ استفاده شده




شکل ۴. (a) تصویر رنگی گرفته شده از دوربین AR.Drone 2.0، (b) تصویر با سطوح خاکستری، (c) تصویر باینری با تبدیل آستانه تطبیقی و (d) تصویر کانتورهای تشخیص داده شده



شکل ۵. حذف چرخش و پرسپکتیو در نشانگرهای تشخیص داده شده

شکل ۶. تصویر نشانگرهای پیدا شده

### ۲-۲-۲. تخمین موقعیت سه بعدی با استفاده از تصویر

در این بخش نحوه محاسبه ماتریس دوران و بردار انتقال بین دستگاه مختصات مرجع و دستگاه مختصات دوربین کوادروتور توضیح داده می‌شود. رابطه زیر نحوه نگاشت نقاط از دستگاه مختصات سه بعدی مرجع به صفحه تصویر را توصیف می‌کند،

$$sp_i = A[R_G|T_G]q_i \qquad (۱۲)$$

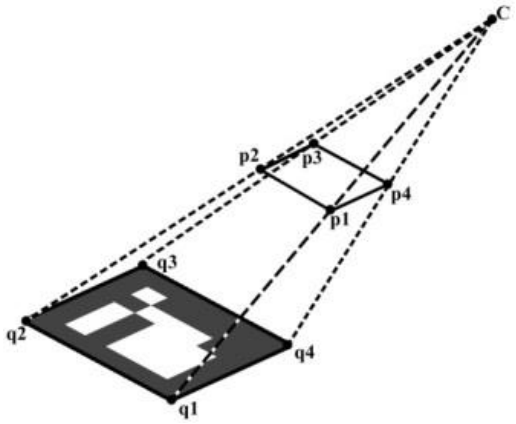
که در آن $s$ پارامتر مقیاس، $q_i$ بیانگر نقطه در دستگاه مختصات سه بعدی مرجع و $p_i$ افکنش $q_i$ $(i = 0 \dots n)$ در صفحه تصویر است. $R_G$ و $T_G$ به ترتیب ماتریس دوران و بردار انتقال از دستگاه مختصات مرجع به دستگاه مختصات دوربین هستند. $A$ ماتریس پارامترهای داخلی دوربین است که به صورت رابطه (۱۳) نشان داده می‌شود.

$$\mathrm{A} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad (۱۳)$$

در این رابطه $[c_x, c_y]^T$، مختصات مرکز تصویر بر حسب پیکسل و $f_x$ و $f_y$ فاصله کانونی در مقیاس پیکسل هستند.

با فرض اینکه نشانگر فرود را در صفحه $X_GY_G$ دستگاه مختصات مرجع قرار داده‌ایم (مولفه $Z_G$ صفر است) و مرکز نشانگر نقطه (0,0,0) دستگاه مختصات مرجع است، برای یافتن ماتریس دوران و بردار انتقال از روش EPnP [۲۳] استفاده می‌کنیم. برای این منظور لازم است مختصات چهار نقطه در دستگاه مختصات مرجع و نقاط متناظر آنها را در صفحه تصویر بدانیم. در این روش مختصات $n$ نقطه سه بعدی به صورت جمع وزن‌دار چهار نقطه کنترلی مجازی نوشته می‌شود. با کاهش این مساله به تخمین مختصات نقاط کنترلی در فریم دوربین، می‌توان این مختصات را با جمع وزن‌دار بردارهای ویژه یک ماتریس ۱۲ در ۱۲ و حل یک معادله کوچک چهارگانه برای انتخاب وزن‌های مناسب بیان کرد. این روش با پیچیدگی $O(n)$ برای محاسبات سریع بسیار مناسب است.

با فرض اینکه مطابق شکل ۷، چهار گوشه نشانگر در دستگاه مختصات مرجع $q_1 - q_4$ و چهار نقطه متناظر آنها در صفحه تصویر $p_1 - p_4$ با استفاده از پردازش تصویر از قسمت قبل بدست آمده باشند و ماتریس $A$ با استفاده از کالیبراسیون دوربین بدست آمده باشد [۲۴]، با استفاده از معادله (۱۲) و روش EPnP می‌توانیم ماتریس $R_G$ و بردار $T_G$ را بدست آوریم.




شکل ۷. افکنش $n$ نقطه از دنیای واقعی به صفحه تصویر (C مرکز دوربین است)

با فرض اینکه $D$ یک بردار دلخواه در دستگاه مختصات مرجع باشد، تبدیل یافته آن در دستگاه مختصات دوربین $D'$ توسط رابطه زیر قابل محاسبه است.

$$D' = R_G D + T_G \qquad (۱۴)$$

با جابه‌جایی معادله (۱۴) خواهیم داشت:

$$D = R_G^T D' - R_G^T T_G \qquad (۱۵)$$

که در آن ماتریس دوران جدید از دستگاه مختصات دوربین به دستگاه مختصات مرجع برابر $R_C = R_G^T$ و ماتریس انتقال جدید برابر $T_C = -R_G^T T_G$ است.

$$[R_C|T_C] = \begin{bmatrix} r'_{11} & r'_{12} & r'_{13} & t'_1 \\ r'_{21} & r'_{22} & r'_{23} & t'_2 \\ r'_{31} & r'_{32} & r'_{33} & t'_3 \end{bmatrix} \qquad (۱۶)$$

بنابراین موقعیت مرکز دوربین نسبت به دستگاه مختصات مرجع از روابط زیر به دست می‌آید [۲۵-۲۶].

$$x_{G_{vision}} = t'_1,\ y_{G_{vision}} = t'_2,\ z_{G_{vision}} = t'_3 \qquad (۱۷)$$

$$\begin{aligned} \varphi_{G_{vision}} &= tan^{-1}(r'_{32}/r'_{33}) \\ \theta_{G_{vision}} &= tan^{-1}(-r'_{31}/\sqrt{(r'_{32})^2 + (r'_{33})^2}) \\ \psi_{G_{vision}} &= tan^{-1}(r'_{21}/r'_{11}) \end{aligned} \qquad (۱۸)$$

که در آنها $x_{G_{vision}}$، $y_{G_{vision}}$ و $z_{G_{vision}}$ مکان پرنده نسبت به دستگاه مختصات مرجع است که با پردازش تصاویر بدست آمده است و $\theta_{G_{vision}}$، $\varphi_{G_{vision}}$ و $\psi_{G_{vision}}$ زوایای پرنده نسبت به دستگاه مختصات مرجع هستند. برای به دست آوردن مختصات دقیق باید دوربین پرنده کالیبره شود. با قرار دادن یک صفحه شطرنجی و یافتن نقاط مشخص و قرار دادن این نقاط در معادله ۱۲ پارامترهای داخلی دوربین به دست می‌آید [27]. پارامترهای کالیبراسیون مربوط به دوربین پرنده به صورت زیر هستند.

$$f_x = 687.6,\ f_y = 687.4$$
$$c_x = 315.5,\ c_y = 185.5 \quad (۱۹)$$

## ۳. کنترل‌کننده فرآیند فرود خودکار

الگوریتم پیشنهادی در این مقاله به دو بخش کنترل با بازخورد حسگر اینرسیایی و بازخورد حسگر دوربین تقسیم می‌شود که در ادامه به بررسی آنها می‌پردازیم.

### ۳-۱. تخمین موقعیت با حسگر اینرسیایی

فرض بر این است که در ابتدای کار پرنده از محل فرود فاصله دارد و نشانگر فرود در محدوده دید دوربین پرنده نیست. بنابراین مختصات تقریبی نشانگر فرود را به پرنده می‌دهیم تا با بازخورد گرفتن از حسگر اینرسیایی به سمت نشانگر پرواز کند. برای کنترل حرکت از کنترل‌کننده PID استفاده شده است. نقطه مقصد به صورت $X_d = [x_{d_G}, y_{d_G}, z_{d_G}]^T$ تعریف می شود که بالای هدف قرار دارد. با فرض $X = [x_{G_{IMU}}, y_{G_{IMU}}, z_{G_{IMU}}]^T$ به عنوان موقعیت فعلی کوادروتور، کنترل‌کننده PID بر روی خطای موقعیت $E = X_d - X$ اعمال خواهد شد، که $E = [e_x, e_y, e_z]^T$ بوده و $e_x$، $e_y$ و $e_z$ به ترتیب خطا در جهت‌های $X_G$، $Y_G$ و $Z_G$ هستند. با توجه به این امر مهم که خطا ممکن است مشتق‌پذیر نباشد، بهره مشتقی به طور مستقیم به مقدار فیدبک اعمال شده است نه به مقدار خطا [۲۵]. معادلات این مرحله به صورت زیر هستند.

$$V_x = K_{px} e_x - K_{dx} \frac{dx_{G_{IMU}}}{dt} + K_{ix} \int_0^t e_x dt \quad (۲۰)$$

$$V_y = K_{py} e_y - K_{dy} \frac{dy_{G_{IMU}}}{dt} + K_{iy} \int_0^t e_y dt \quad (۲۱)$$

$$V_z = K_{pz} e_z - K_{dz} \frac{dz_{G_{IMU}}}{dt} + K_{iz} \int_0^t e_z dt \quad (۲۲)$$

در این روابط $V_x$، $V_y$ و $V_z$ به ترتیب فرمان‌های سرعت اعمالی به پرنده در جهت‌های $X_G$، $Y_G$ و $Z_G$ هستند. $K_p$، $K_i$ و $K_d$ به ترتیب ضرایب تناسبی، انتگرالی و مشتقی کنترل‌کننده PID هستند که در آزمایش‌های تجربی به صورت زیر استخراج شده‌اند.

### ۳-۲. الگوریتم کنترل موقعیت با استفاده از تصویر

زمانی که پرنده به موقعیتی رسید که نشانگر فرود توسط دوربین پرنده قابل رویت باشد الگوریتم تخمین موقعیت مبتنی بر تصویر وارد عمل می‌شود. متاسفانه، پرنده به صورت خطی با فرمان سرعت حرکت نمی‌کند و لغزش‌های فراوانی دارد. برای مثال در نزدیکی هدف که خطای موقعیت کم است، سرعت کوچکی به پرنده اعمال خواهد شد، اما پرنده به دلایل مختلف به سمت خواسته شده و با سرعت خواسته شده حرکت نمی‌کند. علاوه بر این، چون پرنده را از حالت تعادل خارج کرده‌ایم، در جهتی تصادفی شروع به لغزش می‌کند و این باعث می‌شود نتوانیم در حوالی نشانگر که حرکت‌های کوچک و دقیقی مورد نیاز است به خوبی عمل کنیم. علت لغزش پرنده این است که وقتی جابه‌جایی اندک مورد نیاز است فرامین سرعت اعمالی به پرنده نیز مقدار کوچکی خواهند داشت. این فرامین سرعت کوچک قادر نیستند پرنده را به مقدار دلخواه حرکت دهند. از طرفی چون پرنده را از حالت ایستایی خارج کرده‌ایم به دلیل وزن کم خود شروع به لغزش می‌نماید.

مشکل دوم این است که فرمان‌هایی که باید در موقعیت خاصی اجرا می‌شدند به دلیل لختی پرنده و تاخیر موجود اندکی دیرتر اجرا می‌شوند که باعث می‌شود پرنده نتواند دقیقا بالای موقعیت مطلوب توقف کند و حول آن نوسان خواهد کرد. همچنین به علت وجود نداشتن پایدارکننده بر روی دوربین AR.Drone 2.0 ممکن است هنگام زاویه گرفتن پرنده در حین حرکت نشانگر از دید پرنده خارج شود. بنابراین طول بازه حرکت در نزدیکی نشانگر را به بازه‌های کوچکی تقسیم می‌کنیم و بین هر بازه حرکت، یک بازه توقف قرار می‌دهیم تا بتوانیم کنترل دقیق‌تری روی پرنده داشته باشیم. به این الگوریتم، الگوریتم "برش حرکتی" می‌نامیم.

برای این منظور با استفاده از معادلات زیر سرعت حرکت پرنده را ضریبی از اختلاف موقعیت پرنده با نشانگر قرار می‌دهیم.

$$V_x = K_{px_{Vision}} (x_{d_G} - x_{G_{vision}}) \quad (۲۴)$$

$$V_y = K_{py_{Vision}} (y_{d_G} - y_{G_{vision}}) \quad (۲۵)$$

$$V_z = K_{pz_{Vision}} (z_{d_G} - z_G) \quad (۲۶)$$

که در این رابطه $K_{px_{Vision}}$، $K_{py_{Vision}}$ و $K_{pz_{Vision}}$ به ترتیب ضرایب تناسبی کنترل‌کننده در جهت‌های $X_G$، $Y_G$ و $Z_G$ هستند، $x_{G_{vision}}$ و $y_{G_{vision}}$ به ترتیب موقعیت تخمین زده شده پرنده نسبت به دستگاه مختصات مرجع در جهت‌های $X_G$، $Y_G$ مبتنی بر تصویر هستند و $z_G$ جمع وزن‌دار اطلاعات به دست آمده از پردازش تصاویر $(z_{G_{Vision}})$ و اطلاعات ارتفاع گزارش شده

توسط پرنده (حاصل ترکیب موثر حسگرهای آلتراسونیک و فشارسنج) ($z_{G_s}$) به صورت زیر است.

$$z_G = w_1 z_{G_s} + w_2 z_{G_{Vision}} \quad (۲۷)$$

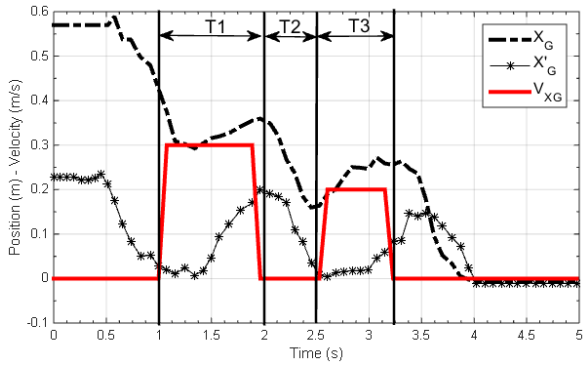
در آزمایش‌های انجام شده بهترین مقدار برای وزن‌های $w_1$ و $w_2$ به ترتیب ۰/۷ و ۰/۳ بدست آمد.

در روش برش حرکتی مدت زمان حرکت را نیز متناسب با اختلاف موقعیت پرنده با نشانگر قرار می‌دهیم.

$$T_x = K_{ptx_{Vision}}(x_{d_G} - x_{G_{vision}}) \quad (۲۸)$$

$$T_y = K_{pty_{Vision}}(y_{d_G} - y_{G_{vision}}) \quad (۲۹)$$

که در این رابطه $K_{ptx_{Vision}}$ و $K_{pty_{Vision}}$ به ترتیب ضرایب تناسبی کنترل‌کننده زمان حرکت در جهت‌های $X_G$، $Y_G$ هستند. به عنوان مثال در شکل ۸ حرکت پرنده به روش برش حرکتی در راستای $X_G$ نشان داده شده است. در این روش دو بازه عملکرد وجود دارد: بازه حرکت و بازه انتظار. برای مثال در بازه T1 پرنده در جهت $X_G$ با سرعت $V_x$ به مدت محاسبه شده $T_x$ حرکت می‌کند و سپس در بازه T2 منتظر می‌ماند تا پرنده به تعادل برسد و تاخیر ناشی از دینامیک یا ارتباط موجود از بین برود و دوباره به مدت T3 حرکت می‌کند. این روند به همین ترتیب ادامه می‌یابد تا به محدوده مناسبی از مرکز هدف برسیم.




شکل ۸ نمودار زمانی بازه‌های حرکت و توقف در روش برش حرکتی

### ۳-۳. الگوریتم تطبیق جهت پرنده

هدف این تحقیق فرود پرنده با جهت معین نسبت به نشانگر است. برای اینکه جهت پرنده را اصلاح کنیم باید پرنده در مرکز نشانگر واقع گردد تا هنگام چرخش، نشانگر از دید دوربین خارج نشود. برای اینکه بفهمیم آیا نشانگر در وسط تصویر است یا نه، یک نقطه مرجع سه بعدی را بر روی نشانگر قرار داده و بررسی می‌کنیم که آیا این نقطه در ناحیه مجاز تصویر است یا خیر. سپس با هدف تطبیق جهت کوادروتور با دستگاه مختصات مرجع، یک فرمان چرخش برای رسیدن به جهت مطلوب به کوادروتور اعمال می‌شود. بنابراین یک کنترل‌کننده PID دیگر نیز به خطای زاویه $\psi$ اختصاص می‌دهیم.

$$e_\psi = \psi_{d_G} - \psi_{G_{Vision}} \quad (۳۰)$$

در این رابطه $\psi_{d_G}$ زاویه مطلوب پرنده نسبت به دستگاه مختصات مرجع و $e_\psi$ خطای زاویه می‌باشد. برای رسیدن به جهت مطلوب پرنده را با سرعت دورانی $\omega_\psi$ در جهت نشانگر می‌چرخانیم، در حالیکه $\omega_\psi$ متناسب با اختلاف زاویه پرنده با نشانگر فرود است. بعلاوه یک عبارت مشتق‌گیر و یک عبارت انتگرال‌گیر به صورت کنترل‌کننده PID مانند رابطه زیر وجود دارد.

$$\omega_\psi = K_{p\psi} e_\psi - K_{d\psi} \frac{d\psi_{G_{vision}}}{dt} + K_{i\psi} \int_0^t e_\psi dt \quad (۳۱)$$

که در این رابطه $K_{p\psi}$، $K_{i\psi}$ و $K_{d\psi}$ به ترتیب ضرایب تناسبی، انتگرالی و مشتقی کنترل‌کننده PID هستند که در آزمایش‌های تجربی به صورت زیر استخراج گردیده‌اند.

$$[K_{p\psi}, K_{i_I\psi}, K_{d\psi}]^T = [0.5, 0.001, 0.001]^T \quad (۳۲)$$

### ۳-۴. الگوریتم فرود

قبل از اینکه کوادروتور ارتفاع خود را کم کند باید سه شرط برقرار گردد: الف) موقعیت کوادروتور در مرکز نشانگر باشد، ب) سرعت افقی پرنده صفر باشد و ج) جهت پرنده با جهت نشانگر همسو باشد.

زمانی که شرایط بالا برقرار باشد به تدریج ارتفاع پرنده را کم می‌کنیم. برای ارتفاع کوادروتور نیز از یک کنترل‌کننده PID به صورت زیر استفاده می‌کنیم.

$$V_z = K_{pz} e_z - K_{dz} \frac{dz_G}{dt} + K_{iz} \int_0^t e_z dt \quad (۳۳)$$

در این رابطه $z_G$ ارتفاع پرنده، $e_z$ خطای ارتفاع و $K_{pz}$، $K_{iz}$ و $K_{dz}$ به ترتیب ضرایب تناسبی، انتگرالی و مشتقی کنترل‌کننده PID هستند که در آزمایش‌های تجربی به صورت زیر بدست آمده‌اند.

$$[K_{pz}, K_{iz}, K_{dz}]^T = [0.5, 0.01, 0.001]^T \quad (۳۴)$$

شبه‌کد الگوریتم فرود خودکار در شکل ۹ و حلقه کنترلی فرود خودکار پرنده در شکل ۱۰ و ارائه شده‌اند. نتایج آزمایش‌های تجربی نشان می‌دهد که در ارتفاع زیر ۲۰ سانتیمتر به علت اثر باد ملخ‌ها بر روی زمین پرنده غیر قابل کنترل می‌شود. راه حل فرود

خودکار ایمن این است که موتورها را در نزدیکی زمین خاموش کنیم.

## ۴. نتایج و بحث

آزمایش‌های تجربی بر روی پرنده AR.Drone 2.0 انجام شده و از دوربین رو به پایین پرنده با کیفیت QVGA استفاده شده است. پرنده توسط ایستگاه زمینی کنترل گردیده است.

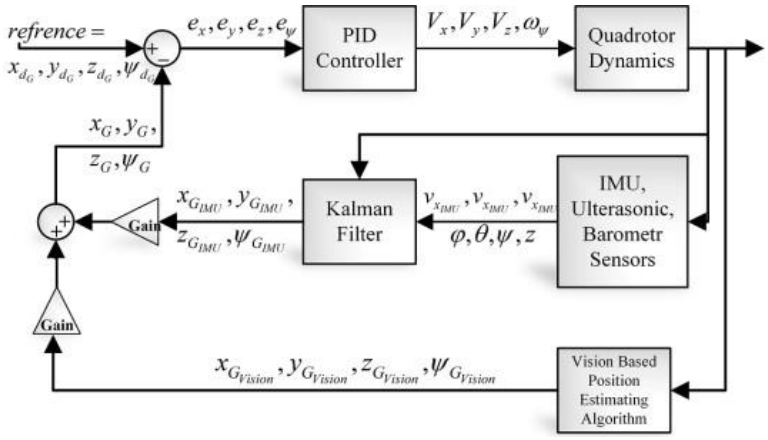
برنامه فرود خودکار پرنده از زیربرنامه‌های مختلفی تشکیل شده است و هر کدام از زیربرنامه‌ها در Threadهای جداگانه نوشته شده‌اند تا سرعت اجرای برنامه افزایش یابد.

به محض برقراری ارتباط ایستگاه زمینی با پرنده، زیربرنامه "دریافت ویدیو" شروع به دریافت تصاویر و کدگشایی آنها می‌کند. جریان ویدیو که روی پورت TCP5555 در حال ارسال است با پروتکل H264 فشرده‌سازی شده است که باید آن را از حالت فشرده خارج کنیم. اطلاعات ناوبری از جمله سرعت‌ها، ارتفاع و زاویه‌های پرنده توسط زیربرنامه "دریافت اطلاعات ناوبری" و در پورت UDP5554 به ایستگاه زمینی ارسال می‌شوند. تمام دستورهای ارسالی به پرنده توسط زیربرنامه "ارسال دستورات کنترلی" و از پورت UDP5556 صورت می‌گیرد. این زیربرنامه وظیفه آماده‌سازی اطلاعات به شکل گفته شده و همچنین ارتباط با کتابخانه‌های socket برای ارسال اطلاعات به پرنده را دارا می‌باشد. زیربرنامه "تخمین موقعیت مبتنی بر تصویر" تصاویر را از زیربرنامه "دریافت و بازگشایی تصاویر" گرفته و مراحل مورد نیاز برای تخمین موقعیت را بر روی آن انجام می‌دهد. اطلاعات سنسورها توسط زیربرنامه دریافت اطلاعات ناوبری دریافت می‌شود. برای به دست آوردن موقعیت برخی پردازش‌ها از جمله فیلتر کالمن مورد نیاز است. برنامه اصلی فرود خودکار در سیستم عامل لینوکس اجرا می‌شود و در یک پنجره تصویر دریافتی از پرنده برای درک بهتر اتفاقات نمایش داده می‌شود. این برنامه توسط محیط برنامه نویسی qt نوشته شده است.

**PSEUDO CODE 1: AUTO LANDING ALGORITHM**

```
1  T, t ←0;
2  While 1 do
3     frame ← get-frame();(V_x, V_y, V_z, ω_ψ) ← 0;
4     (x_GVision, y_GVision, z_GVision, ψ_GVision) ← pose-estimation (frame);
5     (x_GIMU, y_GIMU, z_GIMU) ← Kalman-filter ();
6     if marker is not detected then
7           (x_G, y_G, z_G) ←(x_GIMU, y_GIMU, z_GIMU);
8           (V_x, V_y) ← PID-calculate[ (0, 0), (x_G, y_G)];
9     else
10          (x_G, y_G, z_G) ← (x_GVision, y_GVision, w_1 z_GVision + w_2 z_GS);
11          if −0.05 < x_G, y_G < 0.05 and −0.04 < ẋ_GIMU, ẏ_GIMU < 0.04then
12                if −0.05 < ψ_GVision < 0.05 then
13                      V_z ← PID-calculate[ 0, z_G];
14                      if z_G < 0.2 then
15                            turn off motors;
16                else
17                      ω_ψ← PID-calculate[ 0, ψ_GVision];
18                end if
19          else
20                if t < T then      // moving slot
21                      t ←t + dt;
22                      (V_x, V_y) ← (V_x_temp, V_y_temp);
23                else                //halting slot
24                      if −0.04 < ẋ_GIMU, ẏ_GIMU < 0.04 then
25                            (T_x, T_y) ←K_pt_vision[(x_dG, y_dG) − (x_GVision, y_GVision)];
26                            T ←Max(|T_x|, |T_y|); (V_x, V_y, V_z, ω_ψ)←0; t ←0;
27                            (V_x_temp, V_y_temp) ←K_p_vision[(x_dG, y_dG) − (x_GVision, y_GVision)];
28                      end if
29                end if
30          end if
```

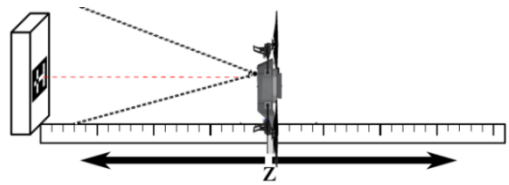
شکل ۹. شبه‌کد الگوریتم فرود خودکار



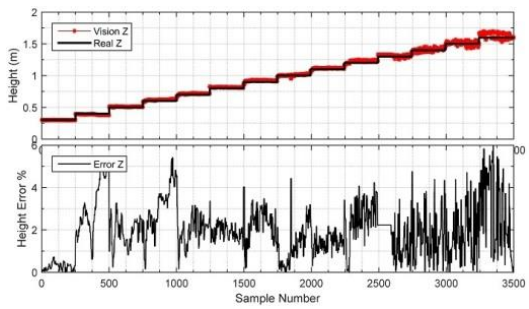

شکل ۱۰. حلقه کنترلی فرود خودکار پرنده بدون سرنشین

### ۴-۱. تخمین ارتفاع با استفاده از تصویر

به منظور بررسی تاثیر فاصله از نشانگر بر دقت خروجی، یک میز آزمایش مطابق شکل ۱۱ تهیه شد. با تغییر فاصله از نشانگر، پارامتر $z_{G_{Vision}}$ را اندازه می‌گیریم در حالی که $x_{G_{vision}}$ و $y_{G_{vision}}$ هر دو صفر هستند. دوربین بین ارتفاع ۳۰ تا ۱۶۰ سانتیمتری حرکت می‌کند. نتیجه این آزمایش و اندازه خطای آن را در شکل ۱۲ مشاهده می‌کنیم.



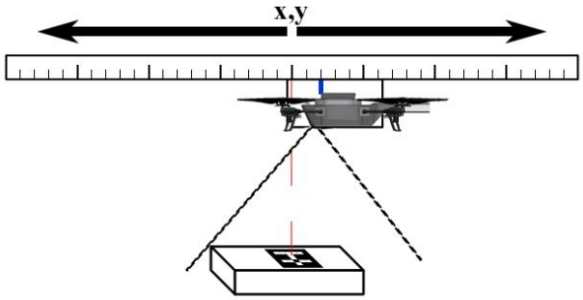

شکل ۱۱. شمای میز آزمایش برای تعیین میزان خطای تخمین ارتفاع



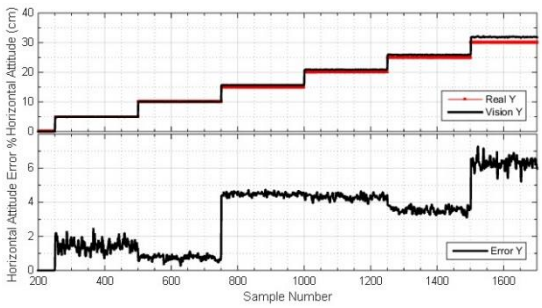

شکل ۱۲. (بالا) نمودار ارتفاع واقعی با ارتفاع تخمینی توسط پردازش تصویر، (پایین) نمودار درصد خطای ارتفاع تخمینی نسبت به مقدار واقعی

### ۴-۲. تخمین موقعیت با استفاده از تصویر

در این آزمایش مطابق شکل ۱۳ دوربین را در ارتفاع ۷۸ سانتی‌متری ثابت می‌کنیم تا بتوانیم $X$ و $Y$ را تغییر دهیم در صورتی که بقیه مختصات را ثابت نگه داریم. در این آزمایش نشانگر از مرکز دوربین تا فاصله ۳۰ سانتی‌متری از مرکز دوربین حرکت داده شده و نتایج آن در شکل ۱۴ قابل مشاهده است.

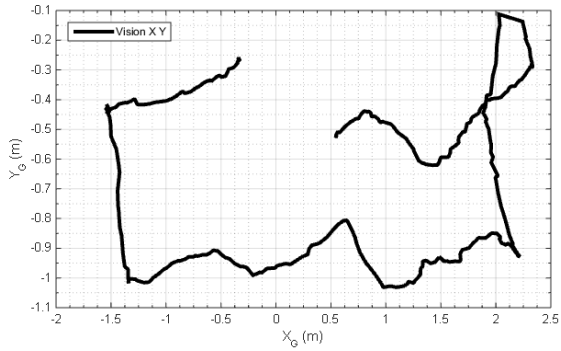



شکل ۱۳. شمای میز آزمایش برای تعیین خطای تخمین موقعیت

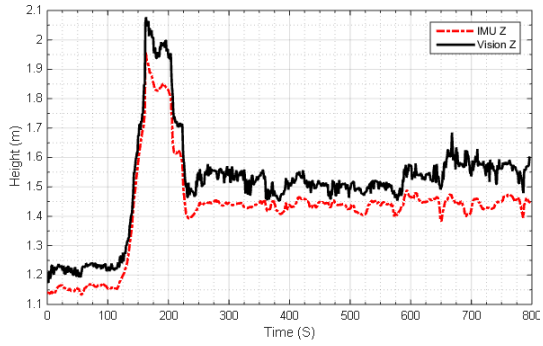



شکل ۱۴. (بالا) نمودار موقعیت واقعی با تخمینی توسط پردازش تصویر، (پایین) نمودار درصد خطای موقعیت تخمینی نسبت به مقدار واقعی

در نمودار شکل ۱۵ پرنده در بالای نشانگر به صورت دستی و به آرامی حرکت داده شده است تا اطلاعات استخراجی به صورت موقعیت مشاهده گردد. همچنین در نمودار شکل ۱۶ پرنده بالای نشانگر به پرواز در آمده و اطلاعات دریافتی از حسگر اینرسیایی و اطلاعات استخراج شده از دوربین با هم مقایسه شده‌اند.



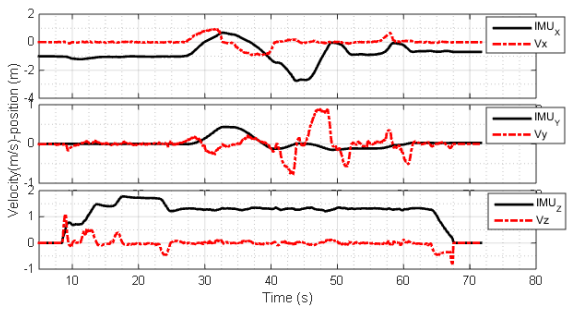

شکل ۱۵. نمودار تخمین موقعیت در هنگام حرکت دستی پرنده




شکل ۱۶. مقایسه اطلاعات ارتفاع دریافتی از حسگر اینرسیایی و دوربین

### ۴-۳. تخمین موقعیت با حسگر اینرسیایی

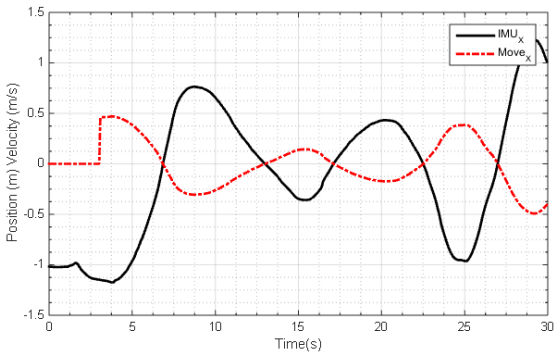
در این آزمایش پرنده را از موقعیت مشخصی پرواز داده و پس از حرکت به سمت‌های مختلف، در همان نقطه اول فرود می‌آوریم. نمودارهای آزمایش در شکل ۱۷ آمده است. با توجه به این نمودار مشاهده می‌کنیم در جهت $X_G$ حدود ۳۰ سانتیمتر و در جهت $Y_G$ حدود ۲ سانتیمتر خطا به وجود آمده است. این خطا ناشی از لغزش و عدم قطعیت حسگرهای اینرسیایی است. در جهت $Z_G$ به علت ترکیب اطلاعات سنسورهای ارتفاع پرنده قطعیت بیشتری وجود دارد و خطا در حد میلیمتر یا کمتر است.



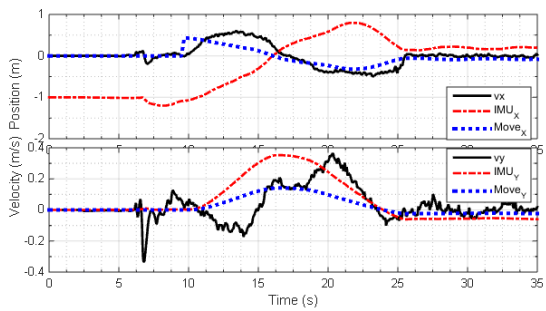

شکل ۱۷. تخمین موقعیت و سرعت با استفاده از حسگر اینرسیایی

### ۴-۴. حرکت به سمت هدف

در نمودار شکل ۱۸ مکان و سرعت پرنده در راستای $X_G$ در یک حرکت دلخواه پرنده از موقعیت $(-1,0,1)$ به $(0,0,1)$ که نشانگر در آنجا قرار دارد، نشان داده شده است. برای این کار ابتدا از یک کنترل‌کننده PID با فیدبکی از حسگر اینرسیایی استفاده می‌کنیم. در شکل ۱۸ مشاهده می‌شود که پرنده در حوالی نقطه هدف نوسان می‌کند که علت آن تاخیر پرنده در اجرای دستورات می‌باشد. برای مثال در لحظه ۶/۵ ثانیه کنترل‌کننده PID متوجه صفر شدن موقعیت می‌شود و در همان لحظه، فرمان سرعت به پرنده را صفر می‌کند. ولی پرنده هنوز در حال حرکت بوده و در ثانیه ۸ این دستور را اجرا می‌کند که دیر شده است و از موقعیت مطلوب اندکی فاصله دارد. دوباره همین روند تکرار می‌گردد و باعث نوسان پرنده می‌شود. برای حل این مشکل، در روش برش حرکتی (شکل ۱۹)، پرنده را به محض رویت هدف متوقف می‌کنیم تا ادامه عملیات فرود توسط بخش پردازش تصویر انجام شود.




شکل ۱۸. نمودار حسگر اینرسیایی در جهت $X_G$ طی حرکت ناموفق



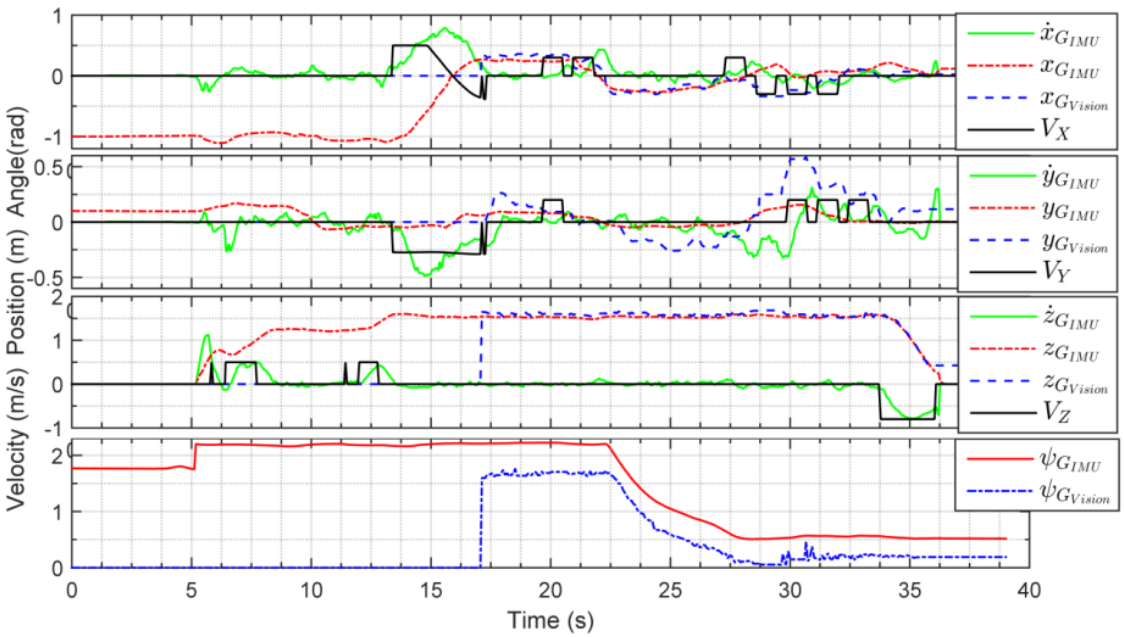

شکل ۱۹. نمودار مربوط به حرکت موفق به سمت موقعیت تعیین شده، سرعت اعمالی، سرعت واقعی و موقعیت پرنده در جهت‌های $X_G$ و $Y_G$

### ۴-۵. قرارگیری دقیق بر روی نشانگر فرود

در این بخش از الگوریتم ارائه شده برش حرکتی استفاده می‌کنیم. با توجه به شکل ۲۰ (در انتهای مقاله) پرنده در ثانیه ۶ ام شروع به پرواز می‌کند و تا ارتفاع حدود ۱/۳ متری بالا می رود و در ثانیه ۱۰ام عملیات فرود خودکار آغاز می‌گردد. پرنده ابتدا نشانگر را مشاهده نمی‌کند، بنابراین با استفاده از حسگر اینرسیایی به سمت موقعیت داده شده حرکت می‌کند تا به حوالی نشانگر برسد. در ثانیه ۱۷ام نشانگر رویت می‌شود و کنترل به حالت پردازش تصویر تغییر می‌کند. پرنده منتظر می‌ماند تا سرعتش صفر شود، سپس اندکی در جهت $X_G$ و $Y_G$ جا به جا می‌گردد تا نشانگر تقریبا در مرکز قرار گیرد. سپس پرنده با کنترل‌کننده PID شروع به تغییر جهت خود و تطبیق آن با نشانگر می‌کند. سپس دوباره پرنده در جهت $X_G$ و $Y_G$ به میزان ۷ ثانیه با سرعت معلوم حرکت می‌کند تا دقیقا مرکز پرنده و مرکز نشانگر برهم منطبق شوند. بعد از آن ارتفاع پرنده با حفظ موقعیت کاهش پیدا می‌کند تا به سطح زمین برسد. کل زمان فرود از ارتفاع ۱/۳ متری از زمین، ۱۵/۷ ثانیه به طول انجامید. همچنین فاصله مرکز دوربین با مرکز نشانگر پس از فرود ۲/۳ سانتیمتر است. این آزمایش به تعداد ۲۰ مرتبه انجام گردید و نتایج در جدول ۱ آورده شده است.

در جدول ۲ مقایسه الگوریتم پیشنهادی با برخی از روش‌های مشابه نشان داده شده است. همچنین از جمله پژوهش‌های انجام شده در سال‌های اخیر در زمینه فرود پرنده بدون سرنشین مبتنی بر تصویر می توان به مرجع [۲۸] اشاره نمود. در این مقاله از نشانگر به شکل H برای فرود استفاده شده است و همانند این تحقیق از فیلتر کالمن و کنترل‌کننده PID برای تخمین و کنترل موقعیت استفاده شده است. همانطور که در [۲۸] اشاره شده است مشکلی که در این تحقیق وجود دارد این است که پرنده در برخی مواقع حرکت‌های ناگهانی انجام داده و ناپایدار می‌شود. در حالی که در الگوریتم پیشنهادی این مقاله به دلیل استفاده از الگوریتم جدید برش حرکتی مشکل فوق مرتفع گردیده است.

همچنین الگوریتم پیشنهادی با نرخ پردازش ۲۴ فریم در ثانیه و خطای جذر میانگین مربعات برابر ۲/۵ سانتیمتر در تخمین موقعیت نسبت به الگوریتم فوق با نرخ پردازش ۲۰ فریم در ثانیه و خطای ۱/۳۷سانتیمتر عملکرد قابل قبولی دارد. همچنین در [۲۹] فرود پرنده بدون سرنشین بر روی عرشه کشتی با استفاده از نشانگر به شکل H شبیه‌سازی شده است. در [۳۰] از یک الگوی فرود خاص استفاده شده است که این الگو باید بصورت مکانیکی طراحی شود و هزینه محاسباتی پیدا نمودن الگو نیز زیاد است. در هر دو مرجع فوق نیز مشکل اشاره شده اولیه کماکان پابرجاست.




شکل ۲۰. نمودار کامل فرود خودکار (سرعت، موقعیت پرنده استخراج شده توسط حسگر اینرسیایی، موقعیت پرنده استخراج شده از دوربین و دستور حرکتی پرنده)

جدول ۱. نتایج تجربی

| موضوع آزمایش | نتیجه |
|---|---|
| تعداد دفعات آزمایش فرود خودکار | ۲۰ بار |
| تعداد دفعات فرود موفق | ۱۷ بار |
| درصد فرودهای موفق | ۸۵ درصد |
| میانگین زمان فرودهای موفق از فاصله ۱ متری | ۱۴/۳ ثانیه |
| میانگین فواصل پرنده از نشانگر پس از فرود | ۲/۹ سانتیمتر |
| میانگین فرکانس پردازش تصاویر | ۲۴ هرتز |
| زمان نمونه برداری $dt = 1/fps$ | $\frac{1}{24}$ =۰/۰۴ |

جدول ۲. مقایسه الگوریتم‌های فرود خودکار

| مرجع | روش کار | ارتفاع فرود (متر) | اندازه نشانگر (سانتیمتر) | حداکثر خطای تخمین موقعیت (سانتیمتر) | حداکثر خطای تخمین ارتفاع (سانتیمتر) | زمان فرود (ثانیه) |
|---|---|---|---|---|---|---|
| [۲۵] | تخمین موقعیت مبتنی بر تصویر بصورت شبیه سازی | ۱/۶ | ۶*۶ | ۱/۴ در ۲۵ (۵/۶٪) | ۵/۵ (۳/۹٪) | ۶۰ |
| [۲۶] | فرود خودکار بر هدف ساکن بصورت شبیه سازی | ۲/۵ | ۳۰*۳۰ | گزارش نشده | گزارش نشده | ۱۷ |
| [۱۶] | تخمین موقعیت مبتنی بر تصویر | گزارش نشده | ۴۵ | ۳/۹٪ | ۲ در ۲۰۰ (۱٪) | گزارش نشده |
| الگوریتم پیشنهادی | تخمین موقعیت مبتنی بر تصویر و اطلاعات حسگرهای اینرسیایی | ۱ | ۱۳*۱۳ | ۱/۸ در ۳۰ (۷٪) | ۱۱/۲ در ۱۶۰ (۶٪) | ۱۴/۳ |

## ۵. نتیجه‌گیری

در این مقاله یک روش جدید برای فرود خودکار کوادروتور پیشنهاد گردید و بر روی یک کوادروتور AR.Drone 2.0 پیاده‌سازی شد. یکی از چالش‌های اساسی در کنترل موقعیت مبتنی بر تصویر یک پرنده نوسانات پرنده حول نقطه مقصد است که ناشی از اینرسی پرنده، تاخیر مربوط به الگوریتم تخمین موقعیت و نیز تاخیر ارسال فرمان‌ها از ایستگاه مرکزی به پرنده می‌باشد. الگوریتم پیشنهادی در این مقاله موسوم به روش برش حرکتی با تقسیم زمان حرکت در نزدیکی نقطه مقصد به بازه‌های زمانی کوچکتر به نام بازه حرکت و بازه انتظار، این مشکل را حل نموده است. زمان بازه حرکت متناسب با خطای موقعیت پرنده است و بازه انتظار تا زمانی که پرنده به وضعیت پایدار برسد و نوسانات زودگذر ناخواسته از بین برود، به طول می‌انجامد. نتایج آزمایش‌های تجربی نشان می‌دهد که الگوریتم پیشنهادی نه تنها باعث پایداری پرنده می‌شود بلکه زمان فرود نیز با خطای قابل قبول در موقعیت پرنده کاهش می‌یابد. برای تعیین تاثیر فاصله از نشانگر بر دقت خروجی الگوریتم یک میز آزمایش تهیه شد و طی آزمایش‌های مختلف خطای تخمین موقعیت با استفاده از حسگرهای اینرسیایی و دوربین مورد مقایسه قرار گرفت. نتایج تجربی بیانگر کارایی مناسب الگوریتم پیشنهادی با خطای در حد سانتیمتر در جابه‌جایی‌هایی به اندازه متر است. در ادامه این تحقیق استفاده از دوربین جلوی پرنده برای یافتن موقعیت تقریبی نشانگر فرود و نیز استفاده از یک پرنده مجهز به پردازنده گرافیکی مانند JETSON TX1 و پردازش بلادرنگ تصاویر بدون نیاز به ارسال به ایستگاه زمینی پیشنهاد می‌شود.

## ۶. مآخذ


[1] J. M. Daly, Y. Ma, S. L. Waslander, Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays, Autonomous Robots, Vol. 38, No. 2, pp. 179-191, 2015.

[2] Y. Mulgaonkar, Automated recharging for persistence missions with multiple micro aerial vehicles, M.Sc. Thesis, University of Pennsylvania, USA, 2012.

[3] K. E. Wenzel, A. Masselli, A. Zell, Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. Journal of intelligent & robotic systems, Vol. 61, No. 1-4, pp. 221-238, 2011.

[4] M. Saska, T. Krajnik, L. Pfeucil. Cooperative μUAV-UGV autonomous indoor surveillance, Proceedings of 9th International Multi-Conference on Systems, Signals and Devices (SSD), Germany, March 20-23, 2012.

[5] Y. Bi, H. Duan, Implementation of autonomous visual tracking and landing for a low-cost quadrotor. Optik-International Journal for Light and Electron Optics, Vol. 124, No. 18, pp. 3296-3300, 2013.

[6] P. Benavidez, J. Lambert, A. Jaimes, M. Jamshidi, Landing of an ardrone 2.0 quadcopter on a mobile base using fuzzy logic, Proceedings of 2014 World Automation Congress (WAC), USA, Aug 3-7, 2014.

[7] C. Yu, J. Cai, Q. Chen, Multi-resolution visual fiducial and assistant navigation system for unmanned aerial vehicle landing, Aerospace Science and Technology, Vol. 67, pp. 249-256, 2017.

[8] S. Shah, Real-time Image Processing on Low Cost Embedded Computers, Techincal report No. UCB/EECS-2014, 2014.

[9] Y. H. Shin, S. Lee, J. Seo, Autonomous safe landing-area determination for rotorcraft UAVs using multiple IR-UWB radars, Aerospace Science and Technology, Vol. 69, pp. 617-624, 2017.

[10] S. Piskorski, N. Brulez, P. Eline, F. Dhaeyer, Ar. drone developer guide, Revision SDK 2.0, Parrot, S.A., 2012.

[11] J. L. Bowditch, The new American practical navigator, E. & G. W. Blunt, New York, 1857.

[12] R. E. Kalman, A new approach to linear filtering and prediction problems, Journal of basic Engineering, Vol. 82, No. 1, pp. 35-45, 1960.

[13] J. Diebel, Representing attitude: Euler angles, unit quaternions, and rotation vectors, Matrix, Vol. 58, No. 15-16, pp. 1-35, 2006.

[14] S. Mitra, Autonomous quadcopter docking system, Project report, Cornell Uiversity, USA, 2013.

[15] M. Podhradsky, Visual Servoing for a Quadcopter Flight Control, M.Sc. Thesis, Czech Technical University, Czech Republic, 2012.

[16] S. Lange, N. Sünderhauf, P. Protzel, Autonomous landing for a multirotor UAV using vision, Proceedings of International Conference on Simulation, Modeling, and Programming for Autonomous Robots, Italy, Nov. 3-4, 2008.

[17] S. Lin, M. A. Garratt, A. J. Lambert, Monocular vision-based real-time target recognition and tracking for autonomously landing an UAV in a cluttered shipboard environment, Autonomous Robots, Vol. 41, No. 4, pp. 881-901, 2017.

[18] C. Patruno, M. Nitti, E. Stella, T. D'Orazio, Helipad detection for accurate UAV pose estimation by means of a visual sensor, International Journal of Advanced Robotic Systems, Vol. 14, No. 5, pp. 1-15, 2017.

[19] T. Krajník, V. Vonasek, D. Fiser, J. Faigl, AR-drone as a platform for robotic research and education, Proceedings of International Conference on Research and Education in Robotics, Czech Republic, June 15-17, 2011.

[20] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, M. J. Marin-Jimenez, Automatic generation and detection of highly reliable fiducial markers under occlusion, Pattern Recognition, Vol. 47, No. 6, pp. 2280-2292, 2014.

[21] S. Suzuki, Topological structural analysis of digitized binary images by border following, Computer Vision, Graphics, and Image Processing, Vol. 30, No. 1, pp. 32-46, 1985.

[22] N. Otsu, A threshold selection method from gray-level histograms, Automatica, Vol. 11, No. 285-296, pp. 23-27, 1975.

[23] V. Lepetit, F. Moreno-Noguer, P. Fua, EPnP: An accurate O(n) solution to the PnP problem, International journal of computer vision, Vol. 81, No. 2, pp. 155-166, 2009.

[24] Z. Zhang, A flexible new technique for camera calibration, IEEE Transactions on pattern analysis and machine intelligence, Vol. 22, No. 11, pp. 1330-1334, 2000.

[25] T. G. Carreira, Quadcopter automatic landing on a docking station, M.Sc. Thesis, Instituto Superior Técnico, Portugal, 2013.

[26] K. Ling, Precision Landing of a Quadrotor UAV on a Moving Target Using Low-Cost Sensors, M.Sc. Thesis, University of Waterloo, Canada, 2014.

[27] G. Bradski, A. Kaehler, Learning OpenCV: Computer vision with the OpenCV library, O'Reilly, 2011.

[28] C. Patruno, M. Nitti, A. Petitti, E. Stella, T. D'Orazio, A Vision-Based Approach for Unmanned Aerial Vehicle Landing, Journal of Intelligent & Robotic Systems, pp 1-20, 2018.

[29] L. Wang, X. Bai, Quadrotor Autonomous Approaching and Landing on a Vessel Deck, Journal of Intelligent & Robotic Systems, Vol. 92, No. 1, pp. 125–143, 2018.

[30] F. Cocchioni, E. Frontoni, G. Ippoliti, S. Longhi, A. Mancini, P. Zingaretti, Visual Based Landing for an Unmanned Quadrotor, Journal of Intelligent & Robotic Systems, Vol. 84, No. 1–4, pp. 511–528, 2016.