Navegação autônoma de drones de tamanho reduzido em ambientes internos

Leonardo Toshinobu Kimura

Resumo

Muitas são as pesquisas conduzidas com o objetivo de se realizar a navegação autônoma de um quadricóptero, particularmente em ambientes internos, onde alguns recursos como GPS não estão disponíveis. Nesse artigo, realizamos a implementação de uma dessas possibilidades, uma navegação baseada apenas em imagens obtidas através de uma câmera, em um drone de tamanho reduzido e limitada carga útil. Realizamos o processamento de imagens através da biblioteca OpenCV, e encontramos a posição da câmera em relação à um marcador artificial através da solução de problemas PnP. Construímos um pequeno ambiente de posicionamento baseado em marcadores, e realizamos com sucesso a localização de um mini drone dentro dele.

1 Introdução

Os drones estão ficando cada vez mais populares, e com isso, diversos usos estão sendo aplicados para eles. Seja em fotografia, filmagem, entrega, monitoração e segurança, as possibilidades são inúmeras. Nesse contexto, muitos esforços estão sendo empreendidos para tornar autônoma a performance do drone, desvinculando da necessidade de um piloto humano para cada movimento do quadricóptero. Enquanto muitos dos modelos comerciais já vem equipados com diversos sensores, como GPS, giroscópio e acelerômetro - o que os torna uma ferramenta ideal para aplicações outdoor - a sua utilização em ambientes cobertos ainda é um desafio não resolvido. Logo, para tarefas como entrega de objetos dentro de um edifício, controle de estoque automatizado, e resgate em ambientes indoor no caso de algum desastre (como terremotos e incêndios), a navegação autônoma desses quadricóptero é, ainda, um obstáculo a ser sobrepujado.

Os veículos micro aéreos, ou chamados mini-drones, são uma ferramenta adequada para esse fim. Com o seu tamanho reduzido e massa mínima, tem a capacidade de navegar por ambientes humanos com baixo grau de interferência. Tendo a capacidade de manter uma mesma posição no ar, bem como executar movimentos precisos, possui uma vantagem adicional de não oferecer grande perigo às pessoas no caso de uma eventual queda. Porém, a autonomação desse dispositivo para uma navegação inteligente oferece desafios adicionais. A sua pequena potência do motor reduz substancialmente a sua carga útil, não sendo possível a instalação de muitos sensores adicionais no mini-drone. Floreano e Wood declararam que "o vôo autônomo em espaços confinados apresenta grandes desafios científicos e técnicos" [1]. Eles destacaram também desafios no design e no balanço energético (o vôo se torna energeticamente mais caro quando o tamanho do componente é reduzido).

Além disso, procuramos implementar uma tecnologia em que seja possível, futuramente, a navegação em ambientes desconhecidos. Isso impossibilita a utilização dos algoritmos que exigem a instalação de dispositivos auxiliares no meio, como câmeras ou radiotransmissores, para a determinação da posição do drone, exceto se fossem objetos naturalmente presentes. Mesmo se nos permitíssemos iniciar com um objetivo mais baixo - como a navegação em um ambiente conhecido com alguns auxílios visuais - procuramos sempre desenvolver a pesquisa de modo a progredir rumo ao objetivo principal, que é a navegação autônoma de um drone de tamanho reduzido em ambientes internos desconhecidos.

Desse modo, nesse projeto, estudamos as diversas alternativas para poder realizar um vôo autônomo de um mini-drone. Estudamos as diversas técnicas de localização disponível na comunidade, mais abundantes para drones de tamanho convencional e qual deles seria o mais adequado para se aplicar em um dispositivo de tamanho e recursos limitados. Implementamos uma dessas soluções, que é a utilização de apenas uma pequena monocâmera de qualidade limitada, e estudamos quais são os algoritmos possíveis nesse caso, bem como as limitações dessas soluções. Por fim, propusemos possíveis desenvolvimentos futuros de modo a poder se avançar no projeto.

2 Modos de navegação atuais

Devido ao interesse crescente da comunidade para as aplicações possíveis utilizando um quadricóptero, diversas pesquisas foram desenvolvidas com o propósito de se propiciar o controle e a navegação autônoma dos drones em ambiente interno. [2] cita vários técnicas utilizadas para se obter o posicionamento do dispositivo, incluindo triangulação, trilateração, tempo de conexão e outros - o que pode ser conseguido por diversas tecnologias, seja por rádio ou sonar. Também existem várias aplicações em que um sistema de tracking externo é utilizado de modo a identificar o drone e, consequentemente, a sua posição [3] [4]. [5] utiliza uma abordagem utilizando uma câmera e um sensor de distância acoplados no veículo (como um laser), [6] realizou um sistema de desvio de obstáculos com uma câmera RGB-D, e [7] trabalha apenas com uma estéreo câmera instalados no drone.

Embora essas abordagens sejam eficazes e consigam cumprir o objetivo delas com suficiente precisão, a aplicação delas no nosso projeto é bastante complicada, considerando um drone de tamanho reduzido e a exigência de interferência mínima do ambiente. A falta de potência nos motores impede a carga de muitos sensores acoplados (como uma estéreo câmera ou um laser frontal), e qualquer peso extra acrescentado ao quadricóptero causaria um distúrbio considerável na performance de vôo dele. Além disso, optamos pela não instalação de dispositivos no ambiente, como as câmeras ou rádiotransmissores, com o objetivo de interferir o mínimo possível o ambiente.

Desse modo, nos concentramos nas soluções que consigam realizar a navegação e controle de um mini-drone utilizando apenas uma câmera como sensor principal.[8][9][10] propuseram uma implementação utilizando de técnicas de processamento de imagem e SLAM com uma mono câmera, porém num drone de tamanho convencional. [11] realizou uma implementação num mini-drone, porém a navegação dele era limitada a se locomover para um ambiente de maior luminosidade. [12] também elaborou o seu projeto num mini-drone, porém a sua navegação era limitada a pairar numa posição constante e, no máximo, a pequenos movimentos pré-definidos. Assim, verificamos que não existe ainda um sistema de navegação autônoma muito complexa utilizando um mini-drone, muito menos o mapeamento de locais desconhecidos de tamanho razoável por técnicas SLAM.

3 Processamento de Imagem

3.1 Processamento de imagem por OpenCV

Para se poder extrair as informações relevantes de uma imagem, como a localização dos marcadores na foto ou das features que puderam ser detectadas no local, foi necessário a utilização de alguns algoritmos de processamento de imagem. Para isso, foi utilizada como ferramenta a biblioteca OpenCV[13], que permite desenvolvimento tanto na linguagem C++ ou Python, e realizou de modo satisfatório o tratamento de imagem que era exigido.

A detecção de features pode ser realizada com diversos algoritmos, como por exemplo, SIFT (Scale Invariante Feature Transform)[14], SURF (Speeded Up Robust Features)[15], FAST (Features from Accelerated Segment Test)[16], Shi-Tomasi[17] e ORB (Oriented Fast and Rotated Brief)[18]. Todos eles possuem diferentes vantagens e desvantagens com relação a características como repetibilidade(as mesmas features devem ser detectadas nas imagens anteriores e posteriores), distinção (capacidade de distinguir uma feature de uma outra, de modo a poderem ser adequadamente correspondida), robustez (resistência a variados níveis de ruído) e custo computacional. Os métodos SURF e FAST costumam ser executados de uma maneira bastante veloz, porém a sua capacidade de distinção das features tende a ser menor.

Uma vez identificada as features, é necessária uma maneira compacta de descrever a região em volta dela, de modo a as podermos diferenciar eficazmente de outras parecidas. Esse método pode ser chamada de extração dos descritores, e a comparação entre elas é chamado de correspondência dos descritores. Existem muitas maneiras de se realizar a correspondência de descritores, e muito esforço já foi empregado no objetivo de se tornar essa correspondência mais rápida, sem prejudicar, porém, a qualidade. Algumas das maneiras mais populares de se extrair esses descritores são SIFT (Scale Invariant Feature Transform)[14], SURF (Speeded Up Robust Features)[15], BRIEF (Binary Robust Independent Elementary Features)[20], BRISK (Binary Robust Invariant Scalable Keyponts)[21], e ORB (Oriented Fast and Rotated Brief)[18]. E uma das maneiras mais simples de se conseguir realizar a correspondência das features é pela estratégia da força bruta, que se consiste apenas de uma comparação de todos os descritores da primeira imagem com todos os descritores

Figura 1: Exemplo de extração de features, pelo algoritmo SIFT [19]

da segunda imagem, selecionando os que são mais similares entre si.

Não é objetivo desse artigo realizar uma extensa comparação de todos os possíveis algoritmos de extração e descrição de features, nem de todos os seus prós e contras. Optamos por utilizar a extração e a descrição através do algoritmo SIFT, por possuir um desempenho mais acurado do que, por exemplo, SURF ou FAST. Além disso, o custo computacional, apesar de ser um pouco maior do que o método FAST, não foi demasiadamente excessivo de modo que impedisse a sua utilização em nossa pesquisa. A correspondência de features foi realizada pelo método da força bruta (BFMatcher).

3.2 A biblioteca ArUco

Uma das tarefas mais importantes para se estimar a localização da câmera pela imagem é a correspondência entre os pontos do ambiente real e suas respectivas projeções 2D na imagem, um processo normalmente difícil e trabalhoso. Por esse motivo, é comum a utilização de marcadores artificais, particularmente os marcadores binários com forma quadriculada. Eles fornecem correspondências suficientes para se calcular a posição da câmera (seus quatro cantos), a partir de um único marcador. Além disso, a sua codificação binária os torna especialmente robustos, sendo funcionais até mesmo em níveis consideráveis de distorção e ruído.

Para o nosso projeto, foi decidido utilizar os marcadores ArUco[22], que se compõe de marcadores quadrados sintéticos compostos por uma larga borda preta e uma matriz binária no seu interior que determina a sua identificação (id). A sua borda larga facilita a sua detecção de maneira veloz, e a sua codificação binária permite a sua identificação de modo acurado e robusto. Os marcadores são construídos de modo que cada uma das suas quatro extremidades são únicas, o que permite reconhecer os marcadores mesmo se elas estiverem rotacionadas, e, assim, é capaz de determinar a sua rotação original.

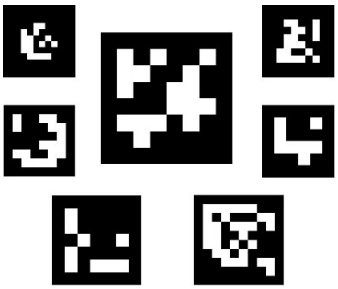

Figura 2: Exemplos de marcadores Aruco utilizados no projeto

Cada marcador pertence a um dicionário, que é um grupo de marcadores considerados para uma aplicação específica. As principais propriedades de um dicionário são o tamanho dos marcadores e o tamanho do dicionário. O tamanho do marcador determina a quantidade de bits dele: por exemplo, um marcador 4x4 possui, em sua matriz interior, 16 bits. Já o tamanho do dicionário indica a quantidade de marcadores que compõe o dicionário. O módulo aruco inclui diversas opções

de dicionários predefinidos, com variados parâmetros de tamanho, tanto do dicionário quanto do marcador.

O processo de detecção de um marcador, dada uma imagem, é realizada em duas etapas: a detecção dos candidatos a marcadores, e a identificação do marcador. A detecção dos candidatos a marcadores começa com uma segmentação de imagem com threshold, seguida pela extração de contornos convexos e que se aproximem da forma de um quadrado. Alguns filtros são aplicados, removendo os contornos muito pequenos ou muito grandes ou contornos muito próximos um do outro, e as regiões restantes são candidatas a corresponderem a um marcador. A identificação do marcador, por sua vez, inicia-se realizando uma transformação em perspectiva para obter a forma original do marcador, uma segmentação de modo a separá-los em bits branco e preto, e a divisão da imagem em diferentes células dependendo do tamanho da matriz. A quantidade de pixels branco e preto de cada célula é analisada, de modo a determinar se o local corresponde a um bit branco e preto. Por fim, todos os bits são analisados de forma a determinar se um marcador pertence ou não a um dicionário específico, e qual seria o número de identificação dele. Quando necessário, são empregadas técnicas de correção de erro adicionais.

4 Localização da Câmera

4.1 Geometria da Imagem

Para ser possível extrair as informações sobre a distância de um objeto a partir da imagem,é necessário um melhor entendimento sobre como a câmera realiza a conversão 3D do mundo para 2D da imagem. Esse conhecimento pode ser adquirido através do modelo de câmera pinhole (significando "buraco de agulha", em inglês). O funcionamento da maioria das câmeras podem ser aproximada para esse modelo, não incluindo os efeitos da distorção causados pelas lentes.

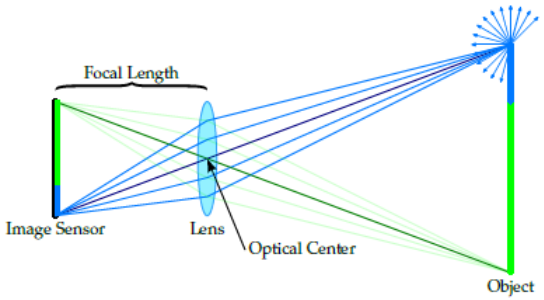

Figura 3: A luz ambiente refletida em um único ponto de um objeto, dos quais alguns raios atingem a lente da câmera, que foca a luz em um único ponto no sensor de imagem [12]

A imagem 3 mostra raios de luz refletindo do objeto para todas as direções, dos quais uma parte deles são capturadas por uma lente e condensadas a um mesmo ponto num plano. Nesse plano há um sensor de imagem, normalmente do tipo CCD (Dispositivo de carga acoplada) ou CMOS (semicondutor de metal-óxido complementar). Diferentes pontos do mundo tem os seus raios de luz condensados em pontos diferentes no sensor. Porém, não é possível determinar a distância que a imagem percorreu antes de atingir o sensor de imagem, podendo ter sido por anos-luz ou por nano-metros. Consequentemente, não é possível determinar se é um pequeno objeto que está perto, ou é um grande objeto que está distante, apenas pela imagem. Essa perda de informação da distância é um grande desafio para a navegação por visão computacional.

A figura 4 representa os sistemas de coordenada utilizada nesse modelo. Uma determinada imagem do mundo ${}^{c}X = (X, Y, Z)^T$ é mapeada no plano de imagem no ponto $x = (u_x, v_x)^T$. O eixo Z da câmera é chamada de eixo principal, e a intercecção entre o eixo principal e o plano de imagem é definida como ponto principal. A foco da câmera é definida como a menor distãncia entre o plano de imagem e o centro da câmera, e pode ser dividida em duas componentes, f_x e f_y. Com isso, é possível realizar a transformação entre a coordenada do mundo e a coordenada na imagem através da seguinte operação:

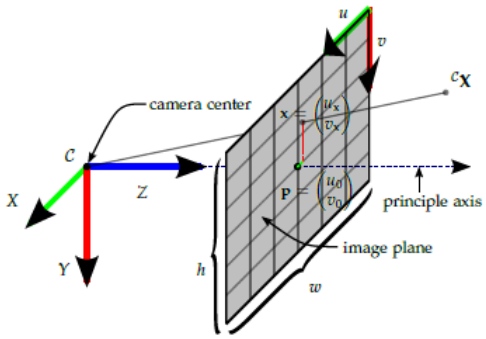

Figura 4: Geometria da câmera Pinhole. [12]

$$\begin{pmatrix} u_x \\ v_x \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & u_x & 0 & 0 \\ 0 & f_z & u_x & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}X \\ 1 \end{pmatrix} \tag{1}$$

Essa matriz K é chamada de matriz dos parâmetros intrínsecos da câmera.

Porém, para determinar a exata posição de um ponto ${}^{w}X$ no referencial do mundo, é preciso também considerar a orientação exterior da câmera. Essa orientação pode ser decomposta por movimentos de translação ${}^{w}_{c}\overrightarrow{t}$, e rotação ${}^{w}_{c}R$, e então a transformação pode ser descrita da seguinte maneira.

$$\begin{pmatrix} u_x \\ v_x \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & u_x & 0 & 0 \\ 0 & f_z & u_x & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} ({}^{w}_{c}R - {}^{w}_{c}R^{T}\,{}^{w}_{c}\overrightarrow{t}) = KP \begin{pmatrix} {}^{w}X \\ 1 \end{pmatrix} \tag{2}$$

Onde a matriz P é chamada de matriz dos parâmetros extrínsecos da câmera.

Porém, apesar de o modelo pinhole conseguir representar suficientemente o funcionamento de uma câmera, ela não contempla as distorções que ocorrem na aquisição da imagem. Dois dos principais tipos de distorções é a distorção radial e a distorção tangencial. As distorções radiais provocam uma curvatura em linhas retas, sendo mais perceptível quando mais longe do centro da imagem. Já as distorções tangenciais ocorrem quando a lente não está alinhada perfeitamente na imagem, tornando algumas áreas aparentemente mais próximas do que o esperado. Esses parâmetros podem ser resumidas em cinco coeficientes, $k1, k2, p1, p2, k3$.

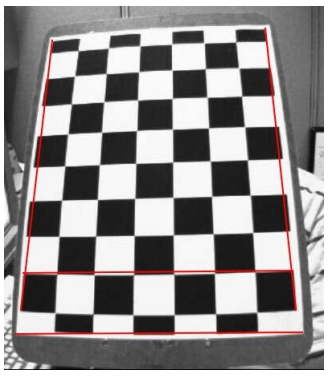

Figura 5: Exemplo de distorção radial da câmera. Note que linhas perto da borda não estão paralelas [19]

4.2 Pontos de fuga

Um ponto de fuga pode ser definido como um ponto no plano da imagem onde as projeções de duas linas paralelas no mundo real parecem convergir. A sua localização fornece muitas informações

úteis sobre a câmera, como o comprimento focal, ou os parâmetros internos da câmera. Do mesmo modo, com dois pontos de fuga associados com retas ortogonais entre si é possível determinar também a rotação da câmera. Sendo os dois pontos de fuga v_x e v_z, podemos utilizar a seguinte relação:

$$r_3 = K^{-1} \cdot v_z \tag{3}$$
$$r_1 = K^{-1} \cdot v_x \tag{4}$$
$$r_2 = r_3 \times r_1 \tag{5}$$

Através da normalização desses valores obtidos, é possível conseguir a matriz de rotação $R = [r_1, r_2, r_3]$. Esse algoritmo foi utilizado nas fases iniciais de nosso projeto.

Figura 6: Exemplo de um ponto de fuga

4.3 Solução de mapas conhecidos por PnP

Quando o ambiente do mundo é conhecido, a estimação da posição da câmera é chamado de problemas PnP (perspective-n-point problem). Isso é, essencialmente, a determinação de 6 graus de liberdade, 3 de translação e 3 de rotação, como podemos ver na equaçao 6, e é um problema trabalhado desde 1841, com Johann Grunert. Quando n=1,2, ou seja, é conhecida a correspondência entre imagens 2D e 3D em apenas 1 ou 2 pontos, o problema não há solução. Quando n = 3,4,5, a resolução do problema são normalmente não lineares, e com $n \geq 6$, os problemas PnP são lineares.

$$s \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{pmatrix} \begin{pmatrix} U \\ V \\ W \\ 1 \end{pmatrix} \tag{6}$$

Existem vários métodos para se resolver esse problema, dependendo do número de pontos e da performance desejada. A biblioteca OpenCV fornece uma função solvePnP, e a possibilidade de se optar por um dos seguintes métodos:

- Método Iterativo. É baseado no algoritmo de otimização de Levenberg-Marquardt, e utiliza o método dos mínimos quadrados para encontrar uma posição que minimize o erro de retroprojeção. É o método que utilizaremos em nosso projeto.
- Método P3P. É baseado no artigo de [23], e necessita de exatamente quatro pontos de entrada. Com os três primeiros pontos, é possível resolver o problema PnP, porém se encontra quatro soluções possíveis para a rotação e para a translação. O quarto ponto, então, é utilizado para se determinar a melhor das soluções encontradas.
- Método AP3P. Desenvolvido no artigo [24], é também uma solução para exatamente quatro pontos de correspondência. É um método otimizado, buscando melhorar tanto a velocidade quanto a acuratividade do método anterior.

- Método EPNP. Sendo um método introduzido no artigo [25], baseia-se na noção de que cada um dos n pontos podem ser expressas como a soma ponderada de quatro pontos virtuais de controle, e então procura resolver as coordenadas desses 4 pontos.

4.4 Soluções em mapas desconhecidos Slam

A Localização e Mapeamento Simultâneos (SLAM, do inglês) é uma técnica computacional que permite ao robô construir ou atualizar um mapa num ambiente desconhecido ao mesmo tempo que consegue determinar a sua própria posição nele. As duas tarefas não podem ser realizadas independentemente: é necessário um mapa acurado para poder se realizar a localização precisa do robô, mas ao mesmo tempo é preciso ter uma exatidão no movimento dele de modo a se poder construir um mapa de qualidade. Além disso, os níveis de ruído e de imprecisão dos instrumentos são um obstáculo, introduzindo erros crescentes no mapa.

Muitos tipos de sensores podem ser utilizados para auxiliar nessa tarefa, como sonares, lasers, ou câmeras. Quando é utilizado apenas uma ou mais câmeras como seus sensores principais, a técnica pode ser definida como Visual SLAM (ou VSLAM). Devido às limitações físicas do mini-drone que estamos utilizando, iremos nos concentrar em técnicas que utilizam apenas uma câmera, resultando num caso que podemos chamar de monoSLAM. Entretanto, devido a impossibilidade de se obter a profundidade diretamente da imagem, os algoritmos em um monoSLAM são muito mais complicados e substanciais. Desenvolvimentos futuros na autonomação do Crazyflie devem seguir essa linha de pesquisa, por isso é importante ter um conhecimento dessas técnicas de navegação robótica.

O primeiro algoritmo de um monoSLAM foi apresentado por Davison, em 2003 [26]. Utilizando um mapa probabilístico baseado em features, ele trabalha em ciclos de um filtro de Kalman Extendido, predição e atualização. Esse método também exige uma inicialização, ou seja, a posição de alguns dos features e da própria câmera já deve ser conhecida com antecedência.

Alguns anos depois, já em 2007, foi apresentado uma versão melhorada desse primeiro algoritmo, chamado de PTAM-SLAM[27] (rastreamento e mapeamento paralelo), que não exige a inicialização com alguns marcadores conhecidos do ambiente. A essência dessa tecnica é dividir as duas tarefas, o rastreamento da câmera e o mapeamento dos pontos, em threads separadas que são executadas concomitantemente num processador multi-core. O mapeamento também não é feito em todo quadro que é adicionado, mas sim em algumas imagens específicas, chamadas de keyFrames. Isso possibilita ao processador utilizar o tempo restante para tornar o mapa tão rico e acurado quanto possível. Porém, esse método exige computadores razoavelmente potentes, e o custo computacional é extremamente alto em ambientes maiores.

Para resolver esse problema, foi desenvolvido um outro tipo de monoSLAM, baseado diretamente nas imagens para o rastreamento e mapeamento, chamado de LSD-SLAM[28] (monoSLAM direto de larga escala). É um método que permite mapear ambientes extensos e não exige nenhum hardware especializado, podendo ser executado até mesmo num smartphone convencional. Não utiliza features, como as outras abordagens de SLAM e, por esse mesmo motivo, consegue gerar um mapa semi-denso, com uma imagem mais significativa para os humanos.

5 Tecnologias utilizadas

5.1 Crazyflie

Para o desenvolvimento do projeto optou-se por utilizar o drone de modelo Crazyflie 2.0, da BitCraze[1]. Esse mini-drone é uma plataforma de desenvolvimento versátil, e pesa apenas 27 gramas, sendo uma ferramenta ideal para vôos em ambientes confinados. Devido à sua pequena capacidade de gerar energia cinética, não causa graves consequências caso ocorra algum acidente, como uma queda ou uma eventual colisão. Medindo apenas 8 centímetros de largura e pesando apenas 27 gramas, possui a capacidade de carregar até 15 gramas e cabe na palma da mão. O modelo possui um microcontrolador 32 bits (STMicroelectronics STM32F405), e é equipado com Bluetooth Low Energy (BLE).

O Crazyflie possui, entre outros sensores, um giroscópio, um acelerômetro e um magnetômetro tridimensionais, e possui uma bateria do tipo LiPo 3,7V com um carregador onboard. Foi construído para ser controlado através de um aplicativo Android ou iOS, porém é possível também

[1] http://www.bitcraze.io/

Figura 7: Foto do mini-drone utilizado no projeto

controlá-lo pelo computador através de um cliente Python e conexão via rádio. Sendo totalmente de código aberto, todos os programas relacionados a ele são facilmente acessíveis e alteráveis, incluindo o firmware, o logdata e os protocolos de comunicação, permitindo um desenvolvimento versátil e flexível.

Foi utilizado também um conjunto de sensores auxiliares com a finalidade única de manter a estabilidade do voô: um sensor de distância até o solo (VL53L0x) e um sensor de movimento óptico (PMW3901). Desse modo é possível manter uma altitude constante, e realizar alguns movimentos simples, como em linha reta ou em diagonal. Porém o alcance desses sensores não permite uma navegação em alturas maiores do que 1,5 metros, e a imprecisão do sensor óptico tente a acumular um erro significativo quando é efetuado um conjunto razoável de movimentos consecutivos.

5.2 Transmissão e recepção da imagem

Para os nossos propósitos, o ideal seria a utilização de uma câmera digital com o máximo de resolução possível e um sistema de captação e transmissão de imagem com a mesma qualidade. Porém, as limitações de alimentação e de processamento existentes num mini drone tornam isso impraticável. Por esse motivo, foi utilizada no projeto uma câmera analógica, de alimentação 2,5-5V, de tamanho reduzido, modelo Eachine TX04, 1000TVL. Possui, embutido nela, um sistema de transmissão de imagem via rádio, com 40 canais de frequência que pode variar de 5,865 a 5,917GHz. O tamanho é de 17,5x17x14mm, e pesa aproximadamente 4g. É do tipo fisheye e, por isso, possui um grande angulo de visão (120°); em contrapartida as imagens resultantes são grandemente distorcidas. É preciso realizar um trabalho de calibragem da câmera, para se encontrar os seus parâmetros internos, e depois para poder compensar a distorção encontrada.

Para se poder fazer a captura da imagem transmitida da câmera, foi utilizado um receptor AV de 5,8GHz, modelo RC832, 12V de alimentação. A sua saída de vídeo foi convertida de formato analógico para digital através de um dispositivo de conversão, Video DVR USB 2.0 Video Adapter. O vídeo resultante é de 640x480, e 30fps, que é enviada por uma porta USB convencional até o computador linux, que realiza todo o processamento de imagem necessário.

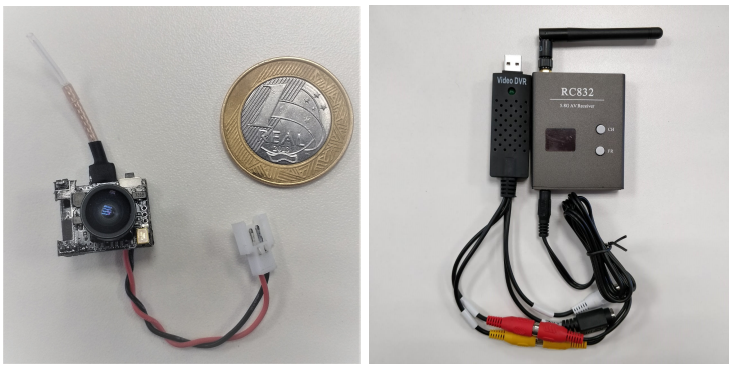

Figura 8: (Esquerda) Fotos da mini câmera; (Direita) Foto do receptor e do conversor DVR

6 Experimentos e resultados

6.1 Construção de trajetórias em vôo cego

Em primeiro lugar, exercitamos o controle do vôo cego do drone. Utilizando-se apenas dos sensores auxiliares de estabilidade e velocidade, foi possível realizar algumas trajetórias simples, como em linha reta, diagonal e movimentos de subida: o sensor de distância mantêm o drone a uma altura constante, e através do sensor óptico é possível controlar a velocidade desejada. Posteriormente, utilizando-se desses movimentos mais simples foi possível construir circuitos mais complexos, desde um movimento de zigue zague ou de arco, até um movimento em loop ou em helicoidal. Porém, a precisão dos sensores era limitada e, por isso, nem sempre o drone encerrava o movimento no local exato.

Um dos problemas constatados foi uma instabilidade significativa do drone no momento da decolagem, causando um pequeno deslocamento de alguns centrímetros de sua posição inicial. Parte desse problema pode ser explicado pelo efeito do ground effect, causada pela turbulência aérea resultante das hélices quando o quadricóptero está muito próximo do solo. Esse efeito foi minimizado pela instalação quatro apoios de x centímetors nos suportes de motores do mini-drone, de modo a mantê-lo mais distante do solo durante a decolagem. Como efeito adicional, o centro de massa do quadricóptero foi deslocado para baixo, tornando-o ligeiramente mais estável durante o vôo.

Foi realizado também um exercício de controle simultâneo de mais de um drone, realizando movimentos coordenados entre si. Foi constatado que a precisão dos movimentos decrescia à medida que a quantidade de drones era incrementada: eventualmente, o quadricóptero avançava mais do que o esperado, ou rotacionava menos do que o desejado. Com o fim de minimizar os erros de comunicação, a frequência de transmissão dos pacotes de comando foi incrementada para o seu máximo (2M), e cada drone foi configurado para operar numa faixa de frequência de 10MHz (de 2,4GHz, 2,41GHz, 2,42GHz...). Os erros continuavam a ocorrer, porém numa taxa substancialmente menor.

6.2 Instalação da câmera no drone

O acoplamento da câmera no mini-drone e o seu correto funcionamento exigiu várias experimentações até atingir um resultado satisfatório. Inicialmente, foi utilizado uma mini-câmera convencional, com uma alimentação de 5V e antena do tipo bulbo. Ela foi instalada na parte superior do mini-drone, e alimentada pela própria bateria do Crazyflie. A câmera apresentou um funcionamento satisfatório, porém somente enquanto os motores não estavam acionados: nesse momento, a imagem adquiria uma quantidade de ruído tamanha que era impraticável detectar qualquer coisa. Experimentamos utilizar vários tipos de filtros com componentes eletrônicos, como filtros RLC, porém sem sucesso. Testamos também a utilização uma outra bateria à parte, dedicada somente alimentação da câmera, porém o excesso de peso não permitia nem mesmo uma decolagem estável.

Uma terceira tentativa foi adicionar um outro componente, um regulador de tensão stepUp, de modo a diminuir essa interferência na câmera. A entrada dessa tensão estava conectada na bateria do mini-drone, e a saída foi configurada de modo a ser exatos 5V. Como resultado, a câmera conseguiu ter a estabilidade necessária, porém o peso adicional proveniente do regulador não também não permitiu uma navegação estável.

Por fim, foi realizada a aquisição de uma nova mini-câmera, com um sistema de antena simples e tensão de alimentação entre 2,5-5V. A energização foi realizada pela bateria do Crazyflie, porém a imagem não foi corrompida de maneira perceptível mesmo quando os motores eram acionados para se efetuar os movimentos. A instalação foi executada na parte superior do mini-drone,e autonomia da bateria foi reduzida consideravelmente, para 7 minutos para 3 a 4 minutos. Maiores informações sobre a câmera podem ser encontrados na sessão 5.2.

6.3 Estimação da posição através de um marcador

Um requisito básico para a estimação da posição da câmera em relação ao algum objeto é a detecção desse próprio objeto na imagem. Com esse fim, optamos inicialmente pela utilização de uma imagem A4 repleta de marcadores como referência, selecionada pela sua grande quantidade de features fáceis de serem rastreados. Com isso realizamos, por algoritmo SIFT, a extração e a detecção das features presentes na imagem modelo (template). Efetuamos a comparação

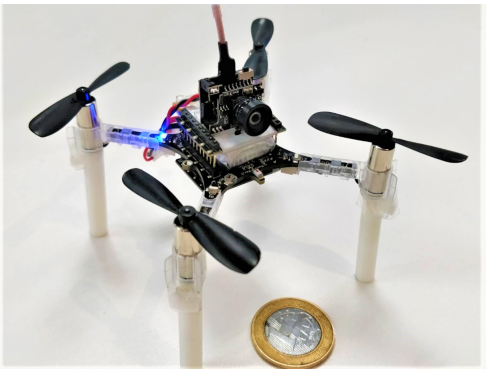

Figura 9: Foto do Crazyflie com uma mini câmera

das features do template com as features extraídas de uma imagem de um vídeo para realizar a detecção, como é possível visualizar na imagem 10. A detecção é bastante satisfatória, sendo possível determinar os contornos do template com bastante precisão. Porém, quando o marcador está a uma distância acima de 0,5 metros, o reconhecimento começa a apresentar bastante sinais de fragilidade, seja pelo fornecimento de coordenadas incorretas ou pela não detecção de um marcador muito distante.

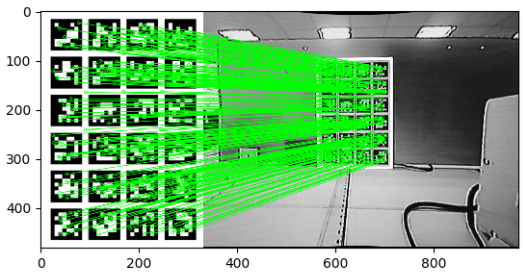

Figura 10: Busca e detecção de um marcador numa imagem, utilizando a extração de features por SIFT e correspondência por força bruta

Obtidos os contornos da imagem, o passo seguinte é a determinação da posição da câmera com relação ao objeto. Para isso vários métodos foram utilizados, de maneira gradual, até a obtenção do resultado desejado. O primeiro método, mais simples, foi apenas uma relação de proporção com uma imagem de referência: ao se fornecer uma imagem cuja distância ao marcador é conhecida, é possível estimar a distância em todas as imagens por comparação. O método é eficaz, porém tem a desvantagem de funcionar apenas com imagens cujo marcador é visto frontalmente. Um pequeno ângulo de incidência já torna essa relação inválida, resultando em valores incorretos de distância.

Foi utilizado, então, um outro método de determinação de distância, baseado em alguns pontos de fuga extraídos da imagem. Com dois pontos de fuga obtidos através de duas pares de retas perpendiculares entre si no mundo real, é possível encontrar os parâmetros extrínsecos da câmera, os vetores de rotação e translação da câmera. Assim, foi possível obter não só a distância da câmera com relação ao objeto, mas também o ângulo de captura entre esses dois elementos.

Porém, este método possui a desvantagem de exigir a imagem de dois pares de retas perpendiculares entre si, o que nem sempre pode estar presente. Além disso, localização dos pontos de fuga nem sempre podem ser detectadas precisamente, o que torna esse método bastante suscetível a erros e ruídos. Por isso, foi utilizado um algoritmo baseado na solução de problemas PnP, um método já implementado na biblioteca OpenCV. Esse método exige uma correspondência entre pontos conhecidos no mundo real em suas projeções na imagem.Para isso, foram fornecidas as coordenadas 3D dos quatro cantos do template e a suas correspondentes 2D obtidas pela detecção do template através do algorítmo SIFT. Porém, como não há a exigência de que esses pontos pertençam a duas pares de retas perpendiculares entre si (como no método por pontos de fuga), seria possível utilizar qualquer ponto conhecido, abrindo o leque de possibilidades para futuras

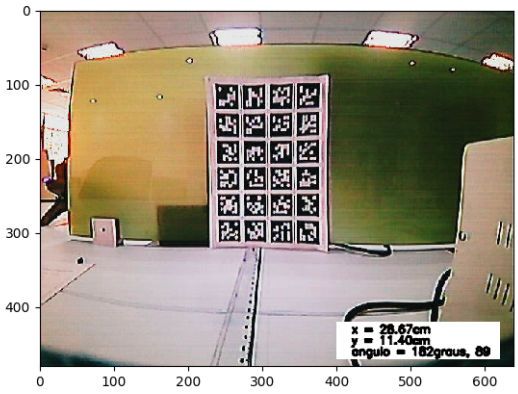

Figura 11: Resultado do cálculo da distância em relação ao marcador. As coordenadas estão em relação ao canto inferior esquerdo do marcador

implementações.

Por fim, foi realizada uma última alteração, com o objetivo de aumentar o alcance da detecção, bem como diminuir seu tempo de processamento: ao invés de se buscar uma dada imagem fornecida anteriormente, foi utilizado o sistema de detecção e reconhecimento de marcadores arUco. Assim, o objeto a ser reconhecido na imagem não seria mais um template, mas marcadores pré-definidos na biblioteca. Como resultado, apesar de restringir a possiblidade de se utilizar imagens genéricas para a orientação, o tempo de processamento tornou-se XXXX% menor, e o alcance da detecção aumentou de 0.5metros para mais de 1 metro (vide tabela 1 e 2). Além disso, foi possível ampliar a busca e detecção para mais de um marcador com pouco esforço, algo que seria bastante trabalhoso com os templates convencionais.

Tabela 1: Desempenho do programa utilizando correspondência de imagem por SIFT

Função	Tempo de processamento (ms)	% de processamento
detectar e computar as features	106	51%
Matching das features	70	34%
calibração das imagens	23	11%
SolvePnP	0.2	<1%
outros	6	3%
total	205	100%

Tabela 2: Desempenho do programa utilizando biblioteca de reconhecimento ARUCO

Função	Tempo de processamento (ms)	% de processamento
calibração das imagens	24	66%
arUco detect	11	30%
SolvePnP	0.2	<1%
outros	1	3%
total	36	100%

6.4 Sistema de posicionamento em um ambiente reduzido

Feito o sistema de cálculo da distância da câmera em relação à um marcador, foi realizado um sistema de localização do mini drone em relação a um ambiente conhecido. Para isso, foi mapeado um pequeno espaço de 1,5x2m, com nove marcadores diferentes, espalhados a uma distância constante. (vide figura 12). Foi definido um sistema de coordenadas cartesiano, com a origem convencionada no canto inferior esquerdo do primeiro marcador. Foi considerado também que o mini

drone se manteria a uma altura constante durante o seu vôo, simplificando os cálculos para apenas dois eixos. As coordenadas dos marcadores, bem como as suas identificações, foram fornecidas anteriormente.

Desse modo, através do vídeo fornecido pela câmera instalada no drone, foi possível determinar a posição do drone em qualquer ponto dentro da área delimitada. Contanto que tivesse algum marcador visível para a câmera, o algoritmo foi capaz de: (1) detectar e identificar o marcador;(2) calcular a distância e o ângulo com relação a marcador; e (3) determinar a coordenada do drone em relação à área delimitada. Para demonstrar o seu funcionamento, foi construído um programa simples que, independente da posição inicial do drone, conseguisse comandar o quadricóptero sempre até uma mesma coordenada definida anteriomente. O programa realizava inicialmente a análise do vídeo, seguido pelo o cálculo da posição do drone no ambiente. Depois, era determinado o movimento que deveria ser feito para se atingir a posição desejada, e, por fim, o comando do drone até essa posição era efetuado. O desempenho e a execução dessa sequencia foi bastante satisfatória, obtendo um erro de menos do que 5 centrímetros em relação a posição final desejada.

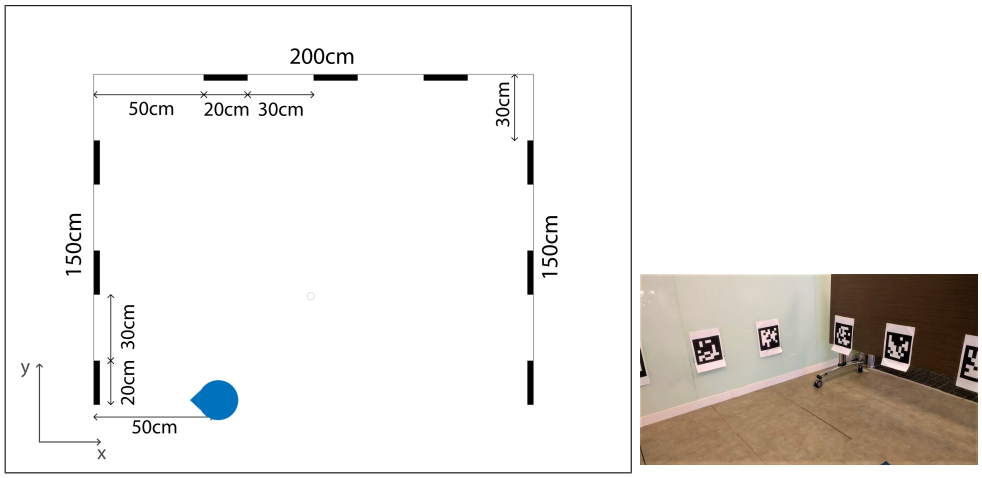

Figura 12: (Esquerda) Esquema do sistema de posicionamento construído. Está indicado também a posição de retorno dele; (Direita) Foto do sistema de posicionamento, com marcadores

7 Conclusão e trabalhos futuros

Nesse artigo, nós trabalhamos a questão da navegação autônoma de quadricópteros, particularmente de drones de tamanho reduzidos em ambientes indoor. Lidamos com as dificuldades inerentes a esses dispositivos, como a impossibilidade de se instalar um número ideal de sensores. Através da instalação de apenas um dispositivo adicional, uma câmera comum, foi possível extrair diversas informações importantes, como distância, orientação e deslocamento em relação a um dado marcador. Foi possível também, com o auxilio de alguns marcadores espalhados pelo ambiente, a construção de um pequeno sistema de localização para um mini-drone. O desempenho do algoritmo foi veloz o suficiente para se poder efetuar o processamento e envio dos comandos em tempo real, e os cálculos da localização gerou resultados satisfatórios.

Desenvolvimentos futuros nessa pesquisa incluem realização desse sistema de localização sem a utilização de marcadores Aruco, e a exploração e o mapeamento de ambientes desconhecidos. Para a retirada dos marcadores pode ser utilizada, em seu lugar, alguns marcadores naturalmente presentes no ambiente ou uma certa quantidade de objetos 3D previamente conhecidos e modelados. Para se realizar a exploração em ambientes desconhecidos, técnicas de SLAM serão desenvolvidas, como o PTAM-SLAM e o LDS-SLAM, já bastante utilizados na comunidade científica.

Referências

[1] D. Floreano and R. J. Wood, "Science, technology and the future of small autonomous drones," *Nature*, vol. 521, pp. 460–466, 2015.

[2] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, pp. 1067–1080, Nov 2007.

[3] S. A. Habsi, M. Shehada, M. Abdoon, A. Mashood, and H. Noura, "Integration of a vicon camera system for indoor flight of a parrot ar drone," in *2015 10th International Symposium on Mechatronics and its Applications (ISMA)*, pp. 1–6, Dec 2015.

[4] "Loco positioning system | bitcraze." `https://www.bitcraze.io/loco-pos-system/`. Acessado em 27 de Julho de 2018.

[5] K. O. Arras and N. Tomatis, "Improving robustness and precision in mobile robot localization by using laser range finding and monocular vision," in *Advanced Mobile Robots, 1999. (Eurobot '99) 1999 Third European Workshop on*, pp. 177–185, 1999.

[6] M. C. P. Santos, L. V. Santana, A. S. Brandão, and M. Sarcinelli-Filho, "Uav obstacle avoidance using rgb-d system," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 312–319, June 2015.

[7] T. Lemaire, C. Berger, I.-K. Jung, and S. Lacroix, "Vision-based slam: Stereo and monocular approaches," *International Journal of Computer Vision*, vol. 74, pp. 343–364, Sep 2007.

[8] J. Engel, J. Sturm, and D. Cremers, "Scale-aware navigation of a low-cost quadrocopter with a monocular camera," *Robotics and Autonomous Systems*, vol. 62, no. 11, pp. 1646 – 1656, 2014. Special Issue on Visual Control of Mobile Robots.

[9] S. García, M. E. López, R. Barea, L. M. Bergasa, A. Gómez, and E. J. Molinos, "Indoor slam for micro aerial vehicles control using monocular camera and sensor fusion," in *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 205–210, May 2016.

[10] L. von Stumberg, V. Usenko, J. Engel, J. Stückler, and D. Cremers, "From monocular slam to autonomous drone exploration," in *2017 European Conference on Mobile Robots (ECMR)*, pp. 1–8, Sept 2017.

[11] L. Kamrath and J. Hereford, "Development of autonomous quadcopter," in *SoutheastCon 2017*, pp. 1–6, March 2017.

[12] O. Dunkley, J. Engel, J. Sturm, and D. Cremers, "Visual-inertial navigation for a camera-equipped 25 g nano-quadrotor," 2014.

[13] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[14] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, pp. 1150–1157 vol.2, 1999.

[15] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346 – 359, 2008. Similarity Matching in Computer Vision and Multimedia.

[16] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), (Berlin, Heidelberg), pp. 430–443, Springer Berlin Heidelberg, 2006.

[17] J. Shi and C. Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 593–600, Jun 1994.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, Nov 2011.

[19] "Opencv python tutorials." `https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html`. Acessado em 03 de Agosto de 2018.

[20] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, (Berlin, Heidelberg), pp. 778–792, Springer-Verlag, 2010.

[21] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *2011 International Conference on Computer Vision*, pp. 2548–2555, Nov 2011.

[22] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38 – 47, 2018.

[23] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 930–943, Aug 2003.

[24] T. Ke and S. I. Roumeliotis, "An efficient algebraic solution to the perspective-three-point problem," *CoRR*, vol. abs/1701.08237, 2017.

[25] V. Lepetit, F.Moreno-Noguer, and P.Fua, "Epnp: An accurate o(n) solution to the pnp problem," *International Journal Computer Vision*, vol. 81, no. 2, 2009.

[26] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1052–1067, June 2007.

[27] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, (Washington, DC, USA), pp. 1–10, IEEE Computer Society, 2007.

[28] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," September 2014.