Courier Drones

Danilo Makoto Ikuta[1], **Gabriel Nopper Silva Rodrigues**[1], **Tales Carlos de Pádua**[1]

[1]Centro Universitário Universitário Senac
04.696-000 – São Paulo – SP – Brasil

danilo0195@hotmail.com, ganopper@gmail.com, talescpadua@gmail.com

Resumo. *Cada vez mais o uso de Drones torna-se comum no mundo. Dia após dia, mais noticias saem sobre drones, sejam elas sobre um acidente, ou publicidade de uma empresa vendedora de peça de Skates que entrega as peças com drones a domicilio.*
Embora ainda não muito populares, drones são uma tecnologia que eventualmente serão tão comuns quanto podemos pensar. Neste projeto, simulamos um drone - a inteligencia dele de reconhecimento grafico, na tentativa de pousá-lo adequadamente e seguramente, além de mapear os arredores durante o percurso. Este projeto permite aos alunos um estudo de métodos euristicos para um funcionamento rapido de um drone, em vários casos incluindo perda de precisão na leitura visual de um mapa, para obter velocidade ou otimizar o uso de uma memória limitada.
Simultaneamente, este projeto funciona através de simuladores se comunicando via socket com um servidor e diversos clientes, também possibilitando um estudo sobre comunicação via pacotes UDP.

1. Introdução

Este projeto é o Projeto Interativo V(Deste ponto em diante chamado apenas por PI5), do curso de Bacharelado em Ciência da Computação do Centro Universitário Senac, 2015. Ele consiste na simulação de voo de um drone inteligente, capaz de identificar-se com o ambiente e realizar um pouso. Para tal, nos utilizamos de alguns recursos bases que ajudam a definir o escopo deste projeto. São estes:

- Um servidor que carrega o simulador de um mundo no qual o drone está lendo, também chamado Simulador de Mundo Externo (SME), responsável por garantir que haja consistencia entre o movimento do drone e o mundo, e por tratar as chamadas do drone.
- Um client que carrega o simulador de um drone, capaz de mandar requests para o servidor, equivalentes ao que seria sua visão sobre o mundo. Com os dados desta visão, ele deve estimar um lugar plano para tentar um pouso.
- Um ou mais mapas no formato X3D (XML) de um ElevationGrid NxM simples, N sendo largura e M sendo altura.
- Um sistema de comunicação por pacote simples UDP entre drone e servidor. A comunicação deve ser executada através de um simples e unico pacote, com payload de 512 bytes.
- Um conjunto de regras criados por um comitê de padronização, contendo um membro de cada grupo responsavel por um projeto, que permita interoperabilidade entre um servidor e um drone quaisquer sejam os projetos usados.

Este projeto é produto de um dos grupos desenvolvendo o PI5 entitulado ”Courier Drones”. Nós nos utilizamos da linguagem Python para ambos Servidor e Drone, com auxilio da Biblioteca numPy para precisão do tamanho de Dados enviados nos pacotes UDP. Este artigo tem como objetivo descrever o funcionamento do sistema Courier Drones, e detalhar a linguagem e métodos usados.

2. Funcionamento da Simulação

Nossa simulação ocorre inicialmente no servidor. Com ele ligado, um drone pode dar um request UDP, inicial, e o servidor automaticamente posiciona o drone em um ponto do mapa aleatório com Y = 80, e então a simulação se inicia, com o drone recebendo sua visão, calculando uma rota aparentemente boa, movendo-se e enviando outro request de visão com o novo movimento. O server reposiciona o drone, verifica se não ouve colisão no movimento, e envia a visão - e o ciclo se repete até a simulação acabar.

Existem 3 causas para o encerramento da simulação. São estes:

- Colisão detectada no servidor. Sendo este o caso, o servidor avisará um problema no console, e não enviará mais mensagens, uma vez que o drone está em tese ”quebrado”.
- Sucesso no pouso, avisado pelo drone. Neste caso, ambos os programas encerram com mensagem de sucesso.
- Falha de conexão, causada por interrupções na internet.

Para qualquer destes casos, na versão atual, é necessário que ambos os programas sejam iniciados novamente para outra simulação. Versões posteriores permitirão que o servidor permança rodando.

2.1. Drone

O drone é um objeto rodado no cliente, contido numa esfera ciscunscrita em 10x10x10. Ele simula um voo por cima do mapa X3D, tentando pousar em qualquer ponto possivel do mapa. Quando a simulação se inicia, o drone é posicionado aleatóriamente no mapa, pelo servidor. O drone não sabe o ponto dele em relação ao mapa, tomando o ponto inicial como central. A cada iteração, ele mandará um request com sua posição e um zoom de camera, simulando sua visão. O servidor recebe este request e devolve uma matriz 15x15, cujo drone memoriza os pontos. O drone deve inferir, com certa inteligencia, pontos provaveis para o pouso. Avaliando um trecho aparentemente adequado, com base em média e moda em determinado trecho, o Drone tenta se aproximar deste local e aumentar o zoom gradativamente, até encontrar uma posição de pouso possivel. Uma vez que o pouso é realizado com sucesso, o Drone manda uma mensagem de Sucesso, encerrando a simulação.

A forma do drone é a de uma semiesfera no eixo y maior que 5, tendo 3 pernas projetadas abaixo na altura y = 2, nos pontos:

8, 8
3, 5
7, 5

2.2. Servidor

O servidor representa o mundo real, o SME, e é responsável por manipular e gerenciar a consistencia da simulação. O servidor inicializa serializando um mapa em .xml no formato X3D de um ElevationGrid. (Mais detalhaho abaixo, no setor mapa) Este mapa é colocado numa matriz, que representa o mapa da simulação.

O servidor espera um request de um drone qualquer, iniciando a simulação ao dar a ele uma posição aleatória com Y = 80 no mapa. A partir deste ponto, ele manda informações do mapa e recebe do drone via a comunicação UDP.

Alem de manipular estas mensagens, o mapa deve verificar consistencia da simulação. Isto envolve conferir se não ouve colisão entre o mapa e o drone. Para isso, usamos uma matriz que compara os espaços sobrevoados pelo drone. Traçamos duas ”reta”na matriz, represntando o trajeto do drone (Com o drone precisamente entre elas) e então executa-se um algoritmo simples de espalhamento em grafo, por largura.A cada iteração do grafo, ele verifica se a altura do drone -3 (ponto 2 da altura dele) é menor ou igual à altura do mapa, no ponto.

No caso postivo, existe colisão e a simulação se encerra.

3. Comunicação Drone-servidor

A comunicação entre o Simulador do Drone (SD) e o Simulador de Mundo Externo(SME) é provavelmente a parte chave deste projeto. Para esta comunicação, um comite de padronização foi organizado, com um membro integrante representando cada grupo desenvolvedor. Este comite tinha a tarefa de padronizar as regras de comunicação, de forma que houvesse intercomunicação entre os grupos, com um servidor operando em outro cliente, etc.

Este comite padroniza o formato dos pacotes UDP para envio de servidor ->cliente, e de cliente ->servidor.

Alem dos padrões de formato de pacote, também padroniza-se que em caso de colisão detectada, o servidor deixa simplesmente de enviar qualquer coisa, assumindo que o drone está quebrado. Portanto, o executável do Drone ficará perdido.

3.1. Servidor ->Cliente

O servidor envia um pacote com as seguintes informações:

- O ID do Drone Cliente, posteriormente usado para identificar para quem mandar a resposta, ao simular multiplos drones.
- O Zoom Aplicado no Mapa, enviado de volta para o drone preencher o mapa dele adequadamente.
- O Vento no instante, valor ainda não utilizado nesta versão
- Livre (Não utilizado)
- Mapa de pontos do sensor, representando uma matrix 15x15, em pares de valores. O primeiro Byte da dupla representa o tipo de terreno, ainda inutilizado. O segundo byte, por sua vez, representa a altura em y no mapa. A matriz é escrita da esquerda para a direita, de cima para baixo.

Byte	Informação	Formato
0	ID do drone cliente	Unsigned int 8 bits
1	Zoom aplicado no mapa	Unsigned int 8 bits
2-13	Vento	Não especificado
14-60	Livre (não utilizado)	
61+	Mapa de pontos do sensor	

Distribuição de bytes no pacote UDP, servidor para cliente

3.2. Cliente ->Servidor

Apos cada movimento executado pelo Drone, ele mandará um request para o servidor, com seu deslocamento nos eixos x, y, z, juntamente com seu ID, seu Zoom, e uma informação dizendo se ele pousou ou não.

- A Porta do cliente, para receber as respostas adequadamente. O servidor usa esta informação para enviar a resposta.
- O ID do drone, para o Servidor identificar qual dos multiplos drones está mandando a mensagem. Este valor ainda não é utilizado.
- O zoom do drone no momento da visão. Este valor define o quanto a visão alcançará no mapa. Quanto maior o alcance, maior a perda de informações e margem de erro.
- Deslocamento no eixo X. Este valor tem direção e um valor negativo significa sentido oposto.
- Deslocamento no eixo Y. Este valor tem direção e um valor negativo significa sentido oposto.
- Deslocamento no eixo Z. Este valor tem direção e um valor negativo significa sentido oposto.
- Espaço livre não utilizado.

Byte	Informação	Formato
0-3	Porta do cliente	Unsigned int 32 bits. Byte 0: MSB
4	ID do drone	Unsigned int 8 bits
5	Zoom do drone	Unsigned int 8 bits
6-17	Deslocamento em x	Signed int 32 bits. Byte 6: MSB
	Deslocamento em y	Signed int 32 bits. Byte 10: MSB
	Deslocamento em z	Signed int 32 bits. Byte 14: MSB
18+	Livre (não utilizado)	

Distribuição de bytes no pacote UDP, cliente para servidor

4. Resultados Parciais

Obtivemos neste percurso experiencia com Python efetiva. Mesmo os programadores habituados com Python, estavam habituados a uma versão anterior de Python. A experiencia obtida foi significativa. A experiencia de um comite de padronização mostrou-se interessante e não fosse a falta de presença de um dos membros, excelente. Indiferente dela também aprendemos muito, pois especialmente no final do desenvolvimento foram averiguadas varias formas normais que ainda faltavam, como o byte na mensagem para sinalizar fim de simulação, dentre outros problemas que foram rapidamente acertados, devido ao fato de apenas dois grupos restarem.

Fica a ansiedade para a segunda fase de desenvolvimento.

Referências

[1] Documentação oficial do Python https://www.python.org/doc/

[2] Documentação oficial do NumPy http://www.numpy.org