UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTAÇÃO E SISTEMAS

ÁLISON CORDEIRO DOS SANTOS

LOCALIZAÇÃO USANDO FUSÃO SENSORIAL DE UM GPS E ODOMETRIA VISUAL

João Monlevade
2017

ÁLISON CORDEIRO DOS SANTOS

Localização usando fusão sensorial de um GPS e odometria visual

Monografia apresentada ao curso Engenharia de Computação do Instituto de Ciências Exatas e Aplicadas, da Universidade Federal de Ouro Preto, como requisito parcial para aprovação na Disciplina “Trabalho de Conclusão de Curso II”.

Orientador: Victor Costa da Silva Campos

João Monlevade

2017

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DE ENGENHARIA DE COMPUTAÇÃO

ANEXO III – Ata de Defesa

ATA DE DEFESA

Aos 05 dias do mês de setembro de 2017, às 11 horas e 29 minutos, na sala C302 do Instituto de Ciências Exatas e Aplicadas, foi realizada a defesa de Monografia pelo aluno **Álison Cordeiro dos Santos,** sendo a Comissão Examinadora constituída pelos professores: Prof. M.Sc. Fabrício Javier Erazo Costa, Prof. M.Sc. Harlei Miguel de Arruda Leite e Prof. Dr. Víctor Costa da Silva Campos.

O candidato apresentou a monografia intitulada: *"Localização usando fusão sensorial de um GPS e odometria visual"*. A comissão examinadora deliberou, por unanimidade, pela aprovação do candidato, com nota 8,0 (oito vírgula zero), concedendo-lhe o prazo de 15 dias para incorporação das alterações sugeridas ao texto final.

Na forma regulamentar, foi lavrada a presente ata que é assinada pelos membros da Comissão Examinadora e pelo graduando.

João Monlevade, 05 de setembro de 2017.

Prof. Dr. Víctor Costa da Silva Campos
Professor Orientador/Presidente

Prof. M.Sc. Fabricio Javier Erazo Costa
Professor Convidado

Prof. M.Sc. Harlei Miguel de Arruda Leite
Professor Convidado

Álison Cordeiro dos Santos
Graduando

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DE ENGENHARIA DE COMPUTAÇÃO

ANEXO II – TERMO DE RESPONSABILIDADE

Curso Engenharia de Computação

TERMO DE RESPONSABILIDADE

Eu, Álison Cordeiro dos Santos, declaro que o texto do trabalho de conclusão de curso intitulado *"Localização usando Fusão Sensorial de um GPS e Odometria Visual"* é de minha inteira responsabilidade e que não há utilização de texto, material fotográfico, código fonte de programa ou qualquer outro material pertencente a terceiros sem as devidas referências ou consentimento dos respectivos autores.

João Monlevade, 12 de setembro de 2017

Álison Cordeiro dos Santos
Álison Cordeiro dos Santos

UNIVERSIDADE FEDERAL DE OURO PRETO
INSTITUTO DE CIÊNCIAS EXATAS E APLICADAS
COLEGIADO DE ENGENHARIA DE COMPUTAÇÃO

ANEXO IV – Folha de Aprovação

Curso Engenharia de Computação

FOLHA DE APROVAÇÃO DA BANCA EXAMINADORA

Localização usando fusão sensorial de um GPS e odometria visual

Álison Cordeiro dos Santos

Monografia apresentada ao Departamento de Computação e Sistemas da Universidade Federal de Ouro Preto como requisito parcial da disciplina CSI496 – Trabalho de Conclusão de Curso II, do curso de Bacharelado em Engenharia de Computação e aprovada pela Banca Examinadora abaixo assinada:

Prof. M.Sc. Fabricio Javier Erazo Costa
DEELT - UFOP

Prof. M.Sc. Harlei Miguel de Arruda Leite
DECSI - UFOP

Prof. Dr. Victor Costa da Silva Campos
DEELT - UFOP

João Monlevade, 05 de setembro de 2017

AGRADECIMENTOS

Primeiramente a Deus por ter me dado saúde, força e determinação para superar as dificuldades.

A esta universidade, seu corpo docente, direção e administração pela oportunidade que me foi dada de crescer humanamente e profissionalmente.

Ao meu orientador Victor Costa da Silva Campos, pela paciência, conselhos e pelo suporte em todo tempo que precisei.

Aos meus pais, pelo amor, incentivo e por sempre acreditarem em mim e no meu potencial.

Aos meus irmãos que, como meus pais, sempre foram à base para que nunca desanimasse diante das dificuldades encontradas.

Aos meus amigos, antigos e mais recentes, pelo apoio e pelo companheirismo durante a minha graduação.

E a todos que direta ou indiretamente fizeram parte da minha formação, o meu muito obrigado.

“Entrega o teu caminho ao Senhor; confia
nele, e ele o fará.”

(Salmos 37:5)

RESUMO

O presente trabalho tem como foco o estudo em torno do problema de localização, que é um clássico problema presente na área da robótica. Praticamente todas as pessoas usam ou já usaram uma ferramenta de localização, com isso é aceitável encontrar uma solução cujo objetivo seja melhorar a qualidade desse tipo de ferramenta. Portanto, o objetivo principal aqui é o uso de técnicas de fusão sensorial usando informações de odometria visual obtidas por uma única câmera juntamente com as informações do GPS. Para a captura das informações da odometria visual foi anexado uma câmera em um objeto em movimento e teremos assim uma trajetória tendo como base o fluxo de vídeo obtido por essa câmera. Com o conhecimento de todos os parâmetros intrínsecos obtidos com a calibração da câmera, foi utilizado um algoritmo de odometria visual para estimar essa trajetória. Para a detecção das coordenadas de latitude e longitude do GPS foi utilizado um software gratuito de detecção de trilhas onde o trajeto feito e os dados necessários para o experimento podem ser obtidos por meio do site desse mesmo aplicativo. Por fim para a realização da fusão dessas duas informações, foi escolhido dentre os vários filtros estudados, o filtro de partículas. O mesmo foi implementado no MATLAB com um total de 100 (cem) partículas e os resultados gerados podem ser adquiridos no próprio software. Para que a fusão sensorial tenha um resultado satisfatório, é preciso que as informações adquiridas ao longo do processo sejam as mais precisas possíveis. Pode-se conferir no decorrer do trabalho que apenas a odometria visual obteve resultados com erros que não interferiram tanto no trajeto final. O que não ocorre na obtenção dos dados das coordenadas do GPS. Isso pode ser explicado por diferentes causas. O trajeto feito durante a captura pode ter sido curto diante do erro que o GPS detecta. Esse erro também pode ser diminuído aumentando a velocidade do veículo usado para a captura das informações ou simplesmente usando um GPS diferencial, diferente do comum usado nesse trabalho. Após observar os resultados espera-se que futuramente consiga-se tratar esse erro do GPS para que assim, a fusão tenha resultados mais satisfatórios. Esse objetivo se torna muito próximo, pois como dito antes, o resultado final da odometria visual foi muito bom. Com isso pode-se focar mais na melhoria dos dados do GPS.

Palavras-chave: Odometria Visual. GPS. Fusão Sensorial.

ABSTRACT

This work's focus lies on the localization problem, which is a classic problem in robotics. Almost everyone uses or has made use of a localization tool, and as such it is aceptable to find a solution whose goal is to enhance the quality of this kind of tool. In that regard, the main objective of this work is to make use of sensor fusion techniques using visual odometry information obtained from a single camera together with GPS information. In order to capture the information for the visual odometry, a camera was attached to a moving object and we can find a trajectory taking as base the video flow obtained from this camera. Having knowledge of the intrinsec camera parameters, obtained with the camera calibration, a visual odometry algorithm was employedto estimate this trajectory. In order to detect and store the latitude and longitude GPS information, a free trail detection software was used. In this software, the trajectory made, as well as the necessary data for the experiment can be obtained from the software's web page. Finally, in order to fuse these two informations, we made use (among the filters described in this work) of a particle filter. This filter was implemented in MATLAB with a total of 100 particles. In order for the sensor fusion to have a meaningful result, it is necessary that the information acquired through the process be as accurate as possible. However, in the course of this work, only the visual odometry found the results we were expecting. The GPS data did not really represent the experiment that was carried out. This can be explained by different causes. The trajectory made during the experiment might have been too short (given the GPS precision), or the moving speed might have been too low. These erros might be reduced by employing a faster vehicle or by making use of a differential GPS. After observing the results, we hope that in the future we are able to deal with these GPS errors in order to yield better results. We believe that this objective can be attained given how good the visual odometry results were.

Keywords: Visual Odometry. GPS. Sensor Fusion

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

BRIEF	–	BINARY ROBUST INDEPENDENT ELEMENTARY FEATURES
FAST	–	FAST ACCELERATED SEGMENT TEST
FK	–	FILTRO DE KALMAN
FKE	–	FILTRO DE KALMAN ESTENDIDO
KLT	–	KANADE-LUCAS-TOMASE
GPS	–	GLOBAL POSITIONING SYSTEM
MATLAB	–	MATRIX LABORATORY
OPENCV	–	OPEN SOURCE COMPUTER VISION LIBRARY
ORB	–	ORIENTED FAST AND ROTATED BRIEF
RANSAC	–	RANDOM SAMPLE CONSENSUS
SDV	–	SINGULAR VALUE DECOMPOSITION
USB	–	UNIVERSAL SERIAL BUS

SUMÁRIO

1 - Introdução

A Visão Computacional é o campo da computação que estuda maneiras de extrair informação dos objetos de uma imagem, tais como suas formas, velocidades, etc. Mais especificamente, é a construção de descrições explícitas e claras dos objetos em uma imagem [Ballard e Brown, 1982]. Ela se difere do processamento de imagens porque enquanto o processamento trata apenas da transformação de imagens em outras imagens, a Visão Computacional trata explicitamente da obtenção e manipulação dos dados de uma imagem e do uso deles para diferentes propósitos.

Para que o reconhecimento dessas imagens seja feito de forma adequada, é preciso que haja um processo para tratar as informações referentes a elas. E isso inclui, entre outras coisas, saber remover o ruído que pode aparecer em imagens, saber lidar com diferentes configurações de iluminação e sombra, perceber a profundidade a partir de uma visão em três dimensões e das configurações das sombras, e reconhecer e isolar os objetos numa dada cena.

Essa grande taxa de informação extraída de uma imagem pode ser usada para que outros processos tenham resultados mais satisfatórios. Em teoria fazer a união dessas informações com outros métodos de localização pode resultar em resultados mais precisos e assim expandir os propósitos que a visão computacional nos permite chegar. A localização fornecida por um GPS, por exemplo, pode ser melhorada se unirmos ela com as informações da odometria visual. Esse processo é parte da visão computacional que será abordado com mais detalhes posteriormente.

É importante ter noção da grandeza dessa área da visão computacional, pois esses sistemas são elementos de uso crescente em ambiente industrial. Aplicações desse sistema ocorrem em diferentes áreas do conhecimento, tais como astronomia, medicina, sensoriamento remoto, perícias, robótica, etc.

1.1 - Problema

O problema de localização é um problema clássico em robótica, e cada vez mais aplicado no dia a dia das pessoas [*e.g.* Google Maps, Waze]. O uso de um GPS facilita a localização, entretanto tem uma taxa de atualização baixa e um erro que pode ser considerável em certas aplicações. Outro sensor muito comum e disponível nos dias de

hoje, é a câmera digital que pode ter uma taxa de aquisição muito maior do que um GPS. Logo, seria interessante aliar as informações disponíveis de uma câmera para melhorar a localização fornecida por um GPS.

1.2 – Objetivos

Este projeto tem como objetivo principal o uso de técnicas de fusão sensorial/localização usando um GPS e informações de odometria visual obtidas por uma única câmera.

Este trabalho irá lidar principalmente com as áreas de visão computacional (na parte da odometria visual/calibração da câmera), probabilidade e robótica (na parte da fusão sensorial).

Espera-se que, durante a realização do trabalho, seja possível implementar uma estratégia de fusão sensorial (Filtro de Kalman Estendido, Filtro de Kalman Unscented, Filtro de Partículas ou Filtragem Complementar) que permita aliar as informações obtidas por um método de odometria visual às informações de um GPS para melhorar a qualidade da localização obtida.

1.3 - Justificativa

O problema de localização é bem explorado na área da robótica. Como praticamente todos nós usamos o GPS como uma ferramenta de localização, é interessante o estudo de alguma solução que vise melhorar a qualidade da mesma. A área da visão computacional é bastante abrangente para assuntos de robótica, e seu estudo para esse propósito pode ser algo relevante e que desperte interesse no assunto.

1.4 - Estrutura do trabalho

Esse trabalho é dividido em capítulos. No capítulo 1, foi apresentada uma visão geral do assunto que será abordado, o problema central que se espera ser resolvido e o objetivo em geral que é buscado com a realização do mesmo.

No capítulo 2 está contida a revisão bibliográfica sobre o tema a que se refere o trabalho. Sendo abordada uma grande demanda do material teórico que existe sobre o assunto tratado. Dando assim ao leitor um entendimento melhor do que será visto a seguir.

No capítulo 3 são apresentados os métodos e procedimentos adotados para o desenvolvimento do trabalho.

No capítulo 4 é apresentada a análise dos dados coletados, a partir da realização do experimento realizado com base na metodologia aplicada.

No capítulo 5 são destacados os principais aspectos e contribuições alcançados no trabalho. É apresentado também, um resumo do que foi esperado e se o objetivo geral foi alcançado.

2 - Conceitos gerais e revisão da literatura

A fim de introduzir o leitor em diversos assuntos que serão tratados nesse trabalho, faz-se necessária à revisão de alguns temas que serão abordados. Todos relacionados à Visão Computacional.

Muitas vezes nota-se enquanto viajamos um pequeno aparelho no painel do carro que diz a distância que o carro viajou. Esse aparelho por sua vez é chamado de odômetro. Ele mede o numero de rotações da roda e multiplica isso pela circunferência para obter uma estimativa da distância percorrida pelo carro. Em robótica, odometria é um termo mais geral e muitas vezes se refere a estimar não apenas a distância percorrida, mas toda a trajetória de um robô e o seu movimento. Então podemos dizer que para cada instância existe um vetor que descreve a posição completa do robô naquela instância.

2.1 - Odometria Visual

Odometria Visual é o processo que a partir de imagens e parâmetros consegue-se obter a posição e orientação de uma determinada câmera. Com isso tem-se o conhecimento das características que ela possui. Assim, é necessário um sistema robusto e eficiente que só pode ser desenvolvido com o conhecimento desses parâmetros envolvidos no processo. Desse modo, o desenvolvimento de procedimentos de calibração e rotinas de correção de erros são elementos fundamentais na determinação de localizações e posições de elementos no plano da imagem.

2.2 - Calibração da Câmera

Calibração pode ser definido como o processo de determinação de características geométricas e ópticas internas da câmera assim como sua orientação e posição a um certo sistema de coordenadas do mundo [Marques, 2007].

A calibração consiste em determinar todos os parâmetros que caracterizam o sistema, o ambiente em que se insere e o modo como será utilizado [Meireles, 2002]. Estes parâmetros podem ser referentes tanto a características físicas de objetos como a coeficientes ou matrizes. A caracterização completa do modelo é feita pela determinação de todos os parâmetros e para que essa correspondência seja feita é necessário efetuar

antecipadamente a calibração da câmera. Esta etapa se resume em duas fases: a determinação dos seus parâmetros intrínsecos e extrínsecos.

Os parâmetros intrínsecos servem para poder relacionar às coordenadas internas das imagens com as coordenadas do espaço medidas no sistema referencial com origem no centro da lente da câmera. Estes parâmetros dependem exclusivamente das características físicas da câmera.

Os Parâmetros extrínsecos consistem na posição e na orientação da câmera referente a outro sistema de coordenadas. A determinação desses parâmetros nem sempre é necessária, caso se possa trabalhar exclusivamente com os parâmetros intrínsecos. Contudo, é recomendável a sua determinação, porque a casos em que esses parâmetros são bem utilizados.

A calibração da câmera envolve a determinação do grupo de parâmetros necessários para a reconstrução do feixe perspectivo gerador da imagem [Mazon, Zacchi e Martins, 2011]. Os parâmetros de calibração existentes são: ponto principal (X0, Y0), distância focal (f), coeficientes de distorção radial simétrica (K1, K2, K3) e os coeficientes de distorção descentrada (P1, P2).

O deslocamento do ponto principal (X0, Y0) é ocasionado pela não coincidência entre o eixo ótico da câmera e o centro fiducial da fotografia [Mazon, Zacchi e Martins, 2011] como mostra a Figura 1.

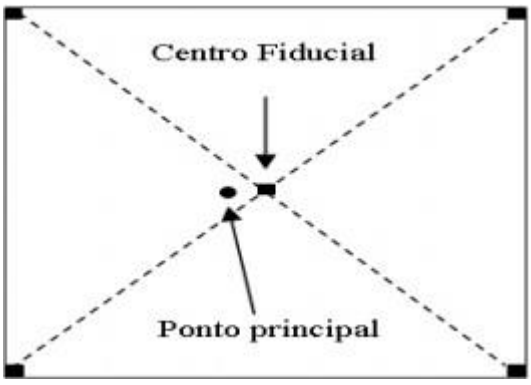

Figura 1 - Descocamento entre o ponto principal e o centro fiducial
Fonte: MAZON, ZACCHI e MARTINS, 2011, p. 2

A distância focal (f) é a medida que define o quando se consegue ver a partir de uma determinada lente. Ela nos diz o quanto da cena será capturado e o quanto foi ampliação dos grandes elementos detectados. Quanto maior a distancia focal, mas estreito o ângulo de visão e maior a ampliação. Quanto menor a distância focal, menor será a visão e menor a ampliação. A figura 2 representa de forma clara a distância focal.

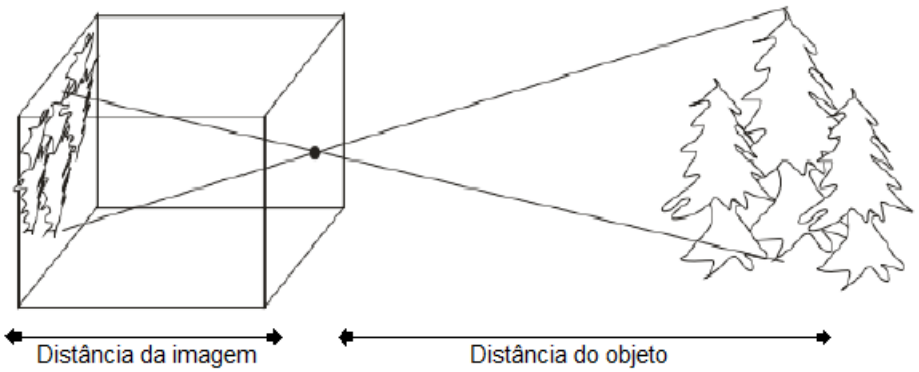

Figura 2 - Representação da distância focal (f)

A distorção radial simétrica (K1, K2, K3) pode ser encarada como sendo a parcela não desejável da refração sofrida por um raio de luz ao atravessar uma lente [Mazon, Zacchi e Martins, 2011]. Desta forma, um raio de luz que antes de penetrar na câmera, forma um ângulo α com o eixo óptico, ao atravessar o sistema de lentes irá resultar em um ângulo α + 2α causando um deslocamento r na posição da imagem no plano do negativo [Andrade, 1998]. A Figura 3 mostra o efeito da distorção radial simétrica.

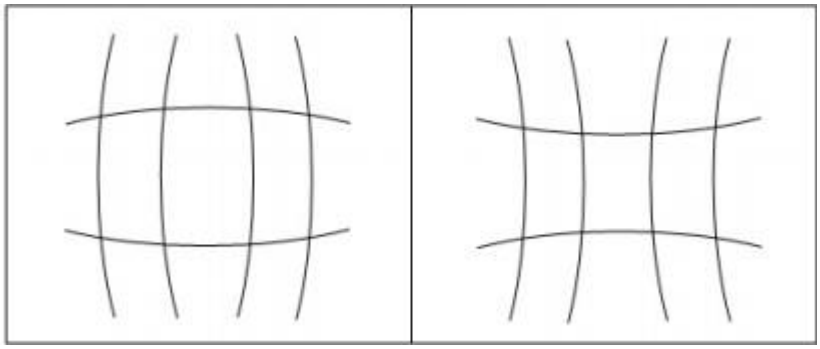

Figura 3 - Efeitos da distorção radial simétrica. À esquerda tem-se a distorção observada através de uma lente positiva e à direita tem-se a distorção observada através de uma lente negativa.
Fonte: Fonte: MAZON, ZACCHI e MARTINS, 2011, p. 3

A distorção descentrada (P1, P2) é gerada pela impossibilidade do fabricante em alinhar perfeitamente os eixos ópticos das lentes que compõe uma objetiva [Mazon, Zacchi e Martins, 2011]. Essa distorção resulta em um deslocamento na imagem composta pelas componentes tangencial e radial simétrica. A Figura 4 mostra o efeito da distorção descentrada.

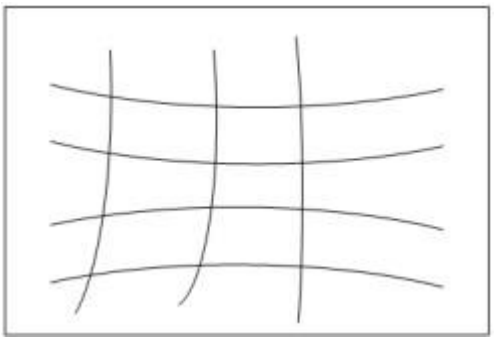

Figura 4 - Efeitos da distorção descentrada
Fonte: MAZON, ZACCHI e MARTINS, 2011, p. 3

2.3 – Robótica Probabilística

A robótica probabilística é uma nova abordagem da robótica que dá destaque às incertezas na percepção e ação do robô. A ideia-chave dessa abordagem é tentar representar essa incerteza explicitamente, usando cálculos da teoria da probabilidade. Assim, em vez de confiar sempre no melhor palpite sobre o que poderia ser no mundo real, os algoritmos probabilísticos representam a informação por distribuições de probabilidade em todo espaço de possíveis hipóteses. Fazendo isso, esses algoritmos podem representar ambiguidade e grau de crença de forma matematicamente sólida, permitindo unir todas as fontes de incertezas presentes na robótica, ou seja, informações críticas que o robô não consegue obter para realizar sua tarefa.

Thrun, Burgard e Fox (2000) ilustram uma abordagem probabilística com um exemplo motivador, a localização de um robô móvel. Essa abordagem pode ser vista na figura 5. Esse problema de localização é conhecido como localização global, onde um robô

é colocado em algum lugar no ambiente e tem que se localizar a partir do zero. No paradigma probabilístico, a estimativa momentânea do robô (também chamada de crença) é representada por uma função de densidade de probabilidade no espaço de todos os locais. Isso está ilustrado no primeiro diagrama da Figura 5, que mostra uma distribuição uniforme que corresponde à incerteza máxima.

Suponha que o robô tenha uma primeira medida de sensor e observa que está ao lado de uma porta. A crença resultante, mostrada no segundo diagrama na Figura 5, coloca alta probabilidade em locais ao lado de portas e baixa probabilidade em outros lugares. Observe que esta distribuição possui três picos, cada um correspondente a uma das portas (indistinguíveis) no ambiente. Além disso, a distribuição resultante atribui alta probabilidade a três locais distintos, ilustrando que a estrutura probabilística pode lidar com múltiplas e conflitantes hipóteses que, naturalmente, surgem em situações ambíguas. Finalmente, mesmo os locais chamados “não porta” possuem probabilidade não nula. Isto é explicado pela incerteza inerente à detecção: com uma probabilidade pequena e não nula, o robô pode errar e na verdade não estar ao lado de uma porta. Agora suponha que o robô se mova. O terceiro diagrama na Figura 5 mostra o efeito do movimento do robô em sua crença, assumindo que o robô é movido como indicado. A crença é deslocada na direção do movimento. Também é suavizado, para explicar a incerteza inerente no movimento do robô. Finalmente, o quarto e último diagrama na Figura 5 retrata a crença depois de observar outra porta. Esta observação leva nosso algoritmo a colocar a maioria da massa de probabilidade em um local próximo a uma das portas, e o robô agora está bastante confiante quanto a onde está.

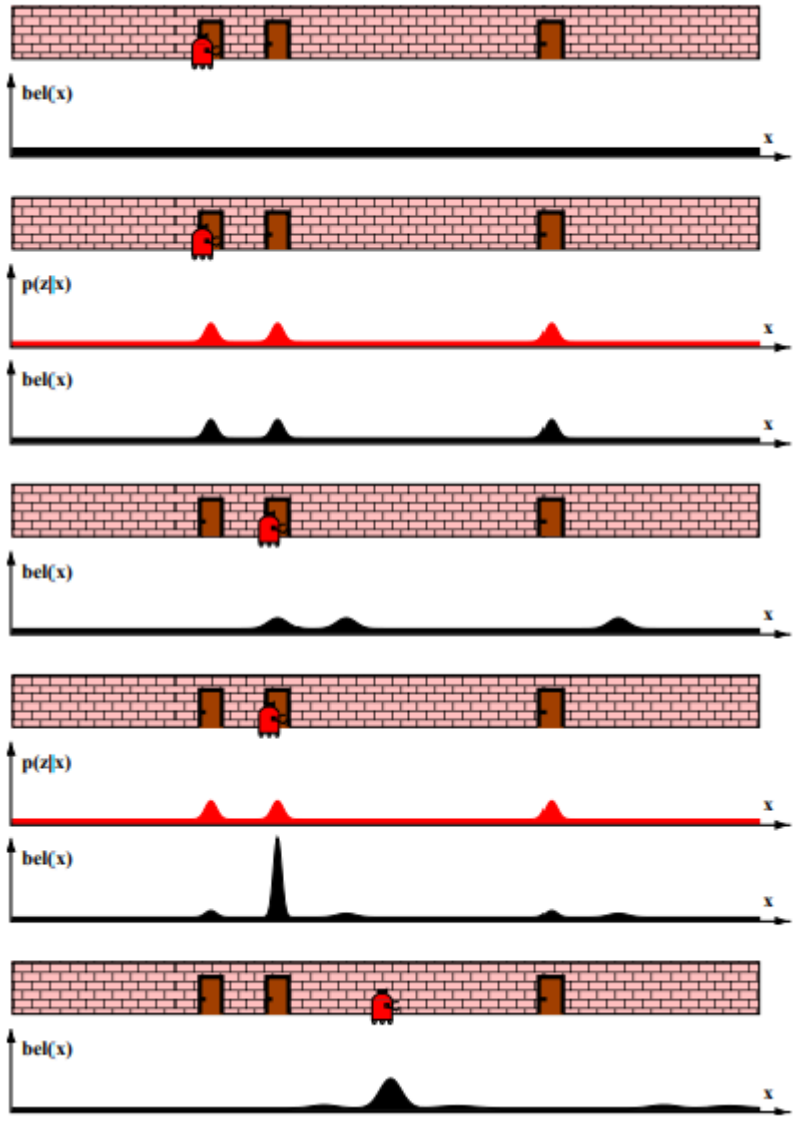

Figura 5 - A ideia básica da localização de Markov: um robô móvel durante a localização global.
Fonte: THRUM e BURGARD, 2000, p. 4

2.4 – Algoritmos Probabilísticos

Como vimos, os algoritmos probabilísticos são responsáveis por eliminar ou minimizar as incertezas ou erros encontrados. Em termos de localização, os algoritmos de filtros que reduzem essa incerteza na posição do robô.

2.4.1 – Filtro de Bayes

O algoritmo do filtro de Bayes é o mais geral para cálculo de crenças. Ele calcula a distribuição de crenças a partir de dados de medição e controle. O filtro de Bayes é feito de forma recursiva, isto é, os cálculos no tempo t são feitos a partir dos cálculos obtidos no tempo t-1. Sua entrada são os dados no tempo t-1, juntamente com o controle mais recente e a mais recente medida obtida. A figura 6 mostra de forma simples uma etapa do algoritmo de filtro de Bayes.

Algoritmo do filtro de Bayes (crença (x_{t-1}), u_t,), z_t):
para todo x_t faz
$\overline{crença}(x_t) = \int p\, x_t \mid u_t, x_{t-1})$ crença(x_{t-1}) dx
crença$(x_t) = \eta\ p(z_t \mid x_t)\ \overline{crença}(x_t)$
fim para
retorna crença(x_t)

Figura 6 - Algoritmo geral para o filtro de Bayes
Fonte: THRUM, BURGARD, 2000, p. 24

2.4.2 – Filtro de Partículas

O filtro de partículas é uma implementação alternativa não paramétrica do filtro de Bayes [Sebastian, 1991]. Assim como os filtros de histograma, os filtros de partículas aproximam o posteriori por um número finito de parâmetros. No entanto, eles diferem na forma como esses parâmetros são gerados, e no qual eles preenchem o espaço de estados. A ideia-chave do filtro de partículas é representar o posteriori por um conjunto de amostras de estados aleatórios tirado deste posteriori. Em vez de representar a distribuição por uma forma paramétrica, os filtros de partículas representam uma distribuição por um conjunto de amostras extraídas dessa distribuição. Essa representação é aproximada, mas não é paramétrica, portanto, pode representar um espaço muito mais

amplo de distribuições do que, por exemplo, filtros gaussianos. O algoritmo do filtro de partículas é representado na figura 7.

Algoritmo do Filtro de Partículas (X_{t-1}, u_t, z_t):
$\bar{X}_t = X_t = \varnothing$
para m = 1 até M fazer
amosta $x_t^{[m]} \sim p\,(x_t \mid u_t, x_{t-1}^{[m]})$
$w_t^{[m]} = p\,(z_t \mid x_t^{[m]})$
$\bar{X}_t = \bar{X}_t + (x_t^{[m]}, w_t^{[m]})$
fim para
para m = 1 até M fazer
amostrar i com probabilidade $\propto w_t^{[i]}$
adicionar $x_t^{[i]}$ até X_t
fim para
retorna X_t

Figura 7 - Algoritmo filtro de partículas
Fonte: THRUM, BURGARD, 2000, p. 78

Nos filtros de partículas, as amostras de uma distribuição futura são chamadas partículas e são denotadas na equação 1:

$$X_t = x_t{}^{[1]}, x_t{}^{[2]}, \ldots, x_t{}^{[M]} \quad [1]$$

Cada partícula $x_t{}^{[m]}$ (com $1 \leq m \leq M$) é uma instanciação concreta do estado no tempo t, ou seja, uma hipótese quanto ao que o estado verdadeiro pode ser no tempo t. Aqui, M indica o numero de partículas no conjunto de partículas X_t. Na prática, o número de partículas M é muitas vezes um número grande, por exemplo, M = 1000, pois assim os resultados podem ser mais satisfatórios. Em algumas implementações, M é uma função de t ou de outras quantidades relacionadas à variável futura.

A intuição por trás dos filtros de partículas é aproximar às amostras futuras pelo conjunto de partículas X_t. Idealmente, a probabilidade de uma hipótese de estado x_t ser incluída no conjunto de partículas X_t deve ser proporcional ao seu filtro de Bayes no futuro.

Esse algoritmo de filtro de partículas ainda se aproximaria do posteriori, mas muitas de suas partículas acabariam em regiões de baixa probabilidade futuras. Como resultado, exigiria muito mais partículas. O passo de reamostragem é uma implementação

probabilística da ideia Darwiniana de sobrevivência do mais forte. Ele reorienta o conjunto de partículas para regiões no espaço de estados com alta probabilidade futura. Ao fazê-lo, ele concentra os recursos computacionais do algoritmo de filtro nas regiões no espaço de estados onde eles mais importam.

2.5 – Fusão Sensorial

A fusão sensorial é um processo que combina informações de diferentes sensores gerando assim uma nova informação. Essa combinação de sensores varia conforme o tipo do resultado final que se espera, assim pode ser usada para diferentes tipos sensores. Tanto para sensores que fornecem as mesmas informações sobre um objeto de interesse quanto para os que fornecem diferentes informações, ela obtém os mesmos resultados.

A fusão sensorial também possui suas limitações. Apesar de obter resultados satisfatórios, sua precisão muitas vezes é limitada com uma razoável quantidade de falhas e redundância. Uma justificativa para sua utilização, além de obter resultados novos, é o alto preço de um sensor que resulta as mesmas informações que a fusão sensorial terá como saída. Assim, é mais viável a escolha de dois sensores mais baratos e realizar a fusão sensorial do que escolher um sensor com um preço não tão acessível.

.

3 - Metodologia

3.1 – Odometria Visual

Há varias maneiras de determinar a trajetória de um robô em movimento. Foi usada aqui uma câmera USB comum (poderia ser também mais de uma câmera) que estará anexada ao notebook em movimento e teremos assim uma trajetória usando o fluxo de imagens proveniente dessa câmera.

O primeiro procedimento a ser feito foi o processo de calibração da câmera, pois é necessário o conhecimento dos parâmetros intrínsecos da mesma para continuação do projeto. Para a realização da calibração da câmera foi usado à biblioteca multiplataforma OpenCV, originalmente desenvolvida pela Intel, em 2000. Com o auxílio da biblioteca foi possível estimar a matriz de projeção e com isso os parâmetros necessários para a realização da calibração. Foi capturado em tempo real um total de 15 (quinze) frames de um tabuleiro para a captura desses parâmetros. A figura 8 mostra como essa captura foi feita. Os cálculos para a obtenção da matriz de projeção com os determinados parâmetros é feita pelo OpenCV não sendo necessário para o usuário a implementação de nada. Só é necessária a mudança de algumas informações que caracterizam a calibração em si. No fim é gerado um arquivo em que as informações da câmera são informados, incluindo a matriz de projeção. Esse arquivo pode ser analisado na figura 9. A matriz de projeção encontrada pode ser vista na equação 2 (dois), em que f_x e f_y representam a distancia focal e c_x e c_y são as coordenadas do ponto principal.

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad [2]$$

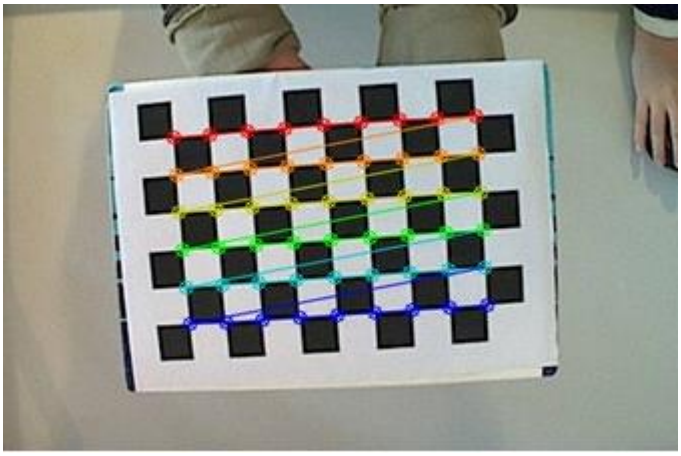

Figura 8 - Captura dos frames do tabuleiro em tempo real

Fonte:

http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html

camera - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

```
%YAML:1.0
---
calibration_time: "08/02/17 17:13:05"
image_width: 640
image_height: 480
board_width: 4
board_height: 5
square_size: 2.5000000372529030e-002
aspectRatio: 1.
flags: 2
camera_matrix: !!opencv-matrix
   rows: 3
   cols: 3
   dt: d
   data: [ 8.5497810138817135e+002, 0., 3.1940161869150671e+002, 0.,
       8.5497810138817135e+002, 2.4019126991364590e+002, 0., 0., 1. ]
distortion_coefficients: !!opencv-matrix
   rows: 5
   cols: 1
   dt: d
   data: [ -1.4466502038069935e+000, 9.9024898884282546e+000,
       3.0395314180155376e-002, 1.5909556608516454e-002,
       -5.4420051423384798e+001 ]
avg_reprojection_error: 2.4443478244820002e-001
```

Figura 9 - Arquivo gerado pelo OpenCV com os parâmetros de calibração

Para estimar a trajetória usando esse fluxo de vídeo usou-se como base um procedimento de odometria visual [Avi Singh's blog, 2015]. Para ficar mais claro vamos formular o problema descrevendo o que seria sua entrada e sua saída. Na entrada, temos um fluxo de imagens de escala de cinza provenientes de uma câmera. Assim, analisa-se frame a frame conforme os dados são captados. Tem-se o total conhecimento prévio de todos os parâmetros intrínsecos que foram obtidos através da calibração. Na saída temos para cada par de imagens uma matriz de rotação e um vetor de translação. Esses valores descrevem o movimento da câmera entre os dois frames. Para a realização desse procedimento foi necessário novamente a biblioteca OpenCV para que a compilação do código pudesse ser feita. O OpenCV disponibiliza uma grande quantidade de algoritmos que são usados na odometria visual. Os algoritmos citados no processo são encontrados nos tutoriais da biblioteca.

O procedimento utilizado pode ser dividido em etapas que são descritas abaixo.

1- Captura dos frames em tempo real.
2- Utilização do algoritmo ORB para detectar as características dos frames capturados em T e T+1.
3- Utilização do algoritmo de 5 (cinco) pontos de Nister com o RANSAC para calcular a matriz essencial.
4- Estimativa da matriz essencial que foi calculada no passo 3 (três).
5- Obter a trajetória do fluxo dos frames.

3.1.1 – Detecção das características

Nessa abordagem, usa-se o algoritmo ORB. Esse algoritmo é basicamente uma fusão do detector de canto FAST e do descritor BRIEF, mas com muitas modificações para melhorar seu desempenho. Mas como funciona? O algoritmo FAST encontra os chamados pontos chave (cantos) e, em seguida, é aplicada a detecção de canto de Harris para encontrar os melhores pontos entre eles. Mas um problema é que o FAST não calcula a orientação. Com isso o ORB apresenta uma modificação. Ele calcula o centroide de intensidade com o canto localizado no centro. A direção do vetor desse ponto de canto para o centroide dá a orientação. Para melhorar a invariância de rotação, os momentos são calculados com X e Y que devem estar em uma região circular de raio r, onde r é o tamanho do trecho analisado. Agora para os descritores, ORB usa o descritor BRIEF. Mas de acordo

com os criadores, BRIEF executa mal a rotação. Então o que o ORB faz é direcionar BRIEF de acordo com a orientação dos pontos chaves. ORB discretiza o ângulo com incrementos de $\frac{2\pi}{30}$ (12 graus) e constrói uma tabela de pesquisa de padrões BRIEF pré-computados. Enquanto a orientação do ponto chave Θ for consistente entre as visualizações, o conjunto correto de pontos será usado para calcular eu descritor. Esse algoritmo já vem pronto no OpenCV, e pode ser encontrado nos tutoriais disponíveis da biblioteca. As informações de como o ORB funciona internamente, tais como seus cálculos matemáticos, podem ser encontradas no artigo que o define [Rublee, Rabaud, Konolige et al., 2011]. O código apresentado na Figura 10 mostra como podemos utilizar o algoritmo ORB em C++ no OpenCV.

```
void featureDetection(Mat img_1, vector<KeyPoint>& Keypoints1, Mat& descriptors1)    {
  Ptr<ORB> orb = ORB::create();
  orb->detectAndCompute(img_1, noArray(), Keypoints1, descriptors1);

}
```

Figura 10 - – Algoritmo ORB

3.1.2 – Rastreamento das características

Os melhores pontos chaves detectados no passo anterior são usados no passo seguinte que usa um rastreador KLT. O KLT é uma implementação, na linguagem de programação C, de um rastreador de recursos para a comunidade de visão computacional. O código-fonte é de domínio público, disponível para uso comercial e não comercial. O rastreador KLT basicamente olha em torno de cada ponto chave analisado para ser rastreado e usa essa informação local para encontrar o ponto chave na próxima imagem. Os pontos chaves detectados em T são rastreados em T+1. A função descrita na figura 11 faz o rastreamento de recursos no OpenCV usando o rastreador KLT.

```
void featureTracking(Mat img_1, Mat img_2, vector<KeyPoint>& points1, vector<KeyPoint>& points2,
                     Mat& descriptors1, Mat& descriptors2, vector<Point2f>& pt1, vector<Point2f>& pt2) {
  Ptr<BFMatcher> bf = BFMatcher::create(NORM_HAMMING, false);
  featureDetection(img_2, points2, descriptors2);
  vector<DMatch> match;
  bf->match(descriptors2, descriptors1, match);

  vector<KeyPoint> apt1 = vector<KeyPoint>(match.size());
  vector<KeyPoint> apt2 = vector<KeyPoint>(match.size());

  for(int i=0; i<(int)match.size(); i++){
    apt1[i] = points1[match[i].trainIdx];
    apt2[i] = points2[match[i].queryIdx];
  }
  KeyPoint::convert(apt1, pt1);
  KeyPoint::convert(apt2, pt2);
}
```

Figura 11 - Função rastreador KLT

3.1.3 – Estimativa da Matriz Essencial

Uma vez que temos correspondências pontuais, temos várias técnicas para a computação de uma matriz essencial. A matriz essencial é definida da seguinte forma na equação 3:

$$y_1^T E\, y_2 = 0 \qquad [3]$$

Onde y_1 e y_2 são coordenadas de imagem normalizadas homogêneas. Enquanto um algoritmo simples requer oito correspondências pontuais. Uma abordagem mais recente que mostra resultados melhores é o algoritmo de cinco pontos de Nister. Ele resolve uma série de equações não lineares e requer o menor numero possível de pontos, já que a Matriz Essencial possui apenas cinco graus de liberdade.

3.1.3.1 – RANSAC

Se todas as nossas correspondências pontuais fossem perfeitas, teríamos apenas cinco correspondências entre dois quadros sucessivos para estimar o movimento com precisão. No entanto, os algoritmos de rastreamento de recursos não são perfeitos e, portanto, temos várias correspondências erradas. Uma técnica padrão de manipulação outliers ao fazer a estimativa do modelo é o RANSAC, um algoritmo iterativo. Em cada iteração, ele detecta aleatoriamente cinco pontos do conjunto de correspondências, estima a matriz essencial e, sem seguida, verifica se os outros pontos são coerentes ao usar essa matriz essencial. O algoritmo termina após um numero fixo de iterações, e a matriz essencial com a qual o número máximo de pontos detectados é usada. Vale ressaltar que o

uso do RANSAC ao invés de um método de mínimos quadrados não lineares é preferido devido à velocidade de solução usando o mesmo. Usar o RANSAC em OpenCV é bastante direto e é só chamar sua função. Tudo o que foi preciso é mostrado na figura 12.

```
E = findEssentialMat(points1, points2, focal, pp, RANSAC, 0.999, 1.0, mask);
```

Figura 12 - Função algoritmo RANSAC em openCV

3.1.4 – Computação de R e t a partir da Matriz Essencial

Outra definição da Matriz Essencial (consistente) com a definição mencionada anteriormente é a mostrada na equação 4:

$$E = R[t]_x \quad [4]$$

Onde R é a matriz de rotação, enquanto $[t]_x$ é a representação da matriz de um produto cruzado com t. Tomando a SVD da matriz essencial, e depois explorando as restrições na matriz de rotação, obtemos o seguinte nas equações 5,6 e 7:

$$E = U\Sigma V^T \quad [5]$$

$$[t]_x = VW\Sigma V^T \quad [6]$$

$$R = UW^{-1}V^T \quad [7]$$

A sua implementação no OpenCV é descrita na figura 14, onde só é preciso chamar sua função.

```
recoverPose(E, points1, points2, R, t, focal, pp, mask);
```

Figura 13 - Implementação da matriz essencial em openCV

Fonte: Elaborado pelo autor

3.1.5 – Construindo a trajetória

Denotando a posição da câmera por R_{pos}, t_{pos}. Podemos seguir a trajetória usando as equações 8 e 9:

$$R_{pos} = RR_{pos} \quad [8]$$

$$t_{pos} = t_{pos} + tR_{pos} \quad [9]$$

Observe que as informações de escala do vetor de translação t devem ser obtidas de alguma outra fonte antes de concatenar. A maioria dos algoritmos de Visão Computacional não está completa sem algumas heurísticas lançadas e a odometria visual não é uma exceção. Todo o algoritmo de odometria visual faz a suposição de que a maioria dos pontos em seu ambiente são rígidos. No entanto, se estamos em um cenário em que a câmera está parada e um trilho passa por ela, isso levaria o algoritmo a acreditar que a câmera se moveu de lado, o que é fisicamente impossível . Como resultado, se acharmos que a translação é dominante em uma direção diferente da frente, simplesmente ignoramos esse movimento.

Após a implementação do algoritmo descrito, foi efetuada a trajetória sendo gravado cada frame, cada matriz de rotação, cada vetor de translação e assim o caminho foi sendo guardado. A trajetória feita foi simples, um caminho de praticamente 50 (cinquenta) metros percorrido no sentido de ida e volta feito 3 (três) vezes. Totalizando no total aproximadamente 300 metros. Foi feito esse caminho, pois a bateria do notebook usado não suportaria um caminho mais longo. Outro notebook não foi usado pois para rodar o algoritmo é necessário ter o OpenCV instalado. Na figura 14 são apresentados alguns frames do trajeto feito para se ter uma noção do tamanho e do lugar em que o experimento foi feito.

Figura 14 - Amostra do caminho feito na odometria

3.2 – Obtenções das coordenadas GPS

Para obter as coordenadas GPS durante o percurso feito na odometria visual foi usado o aplicativo Wikiloc. Aplicativo criado pelo espanhol Jordi Ramot com o objetivo de proporcionar uma ajuda mútua para descobrir trilhas, caminhos, rotas e pontos de coordenadas ao ar livre. Assim as coordenadas de latitude e longitude eram salvas e podiam ser resgatadas a qualquer momento. O aplicativo está disponível e qualquer um pode ter acesso. Sua interface pode ser visualizada na figura 15.

Figura 15 - Aplicativo Wikiloc para obtenção das coordenadas GPS

Fonte: print screen do aplicativo Wikiloc

3.3 – Fusão Sensorial

Após a coleta dos dados necessários tanto da odometria quanto do GPS, foi feito a fusão sensorial utilizando o filtro de partículas. Diante de todos os filtros estudados decidiu-se por escolher o filtro de partículas, pois se percebe que ele se adequa bem ao objetivo que se espera alcançar. Como o objetivo deste trabalho é fazer a fusão sensorial utilizando as informações da odometria e do resultado da captação do GPS foi usado a odometria visual para a propagação e o GPS para a correção dessas informações. De início a captura das informações da odometria e do GPS seriam feitos de forma simultânea no mesmo algoritmo e a realização da fusão sensorial seria feita de forma online. Mas como essa metodologia gerou alguns problemas, foi decidido que a realização da fusão sensorial seria feita de forma off-line. Após essa decisão, foi usado o software MATLAB para a atribuição dos dados. Assim, foi possível implementar o filtro de partículas e com isso gerar os resultados desejados. Foi utilizado o montante de 100 (cem) partículas, pois um número maior gerou uma grande demanda de tempo no MATLAB, fazendo com que a compilação do código demorasse muito, chegando a ser inviável. Contudo 100 (cem) partículas também

é um número aceitável. A parte do código onde é mostrada a inicialização do filtro é mostrado na figura 16.

```
%% Filtro de Partículas
n = 100; %numero de paticulas
x = zeros(2166,n);
y = zeros(2166,n);
xm = zeros(2166,1);
ym = zeros(2166,1);
Ra = cell(n,1);
ta = cell (n,1);
for i=1:n
    theta = theta0 + 0.1*randn;
    Ra{i} = [cos(theta0) 0 sin(theta0); 0 1 0; -sin(theta0) 0 cos(theta0)];
    ta{i} = zeros(3,1);
end
```

Figura 16 - Parte do código no MATLAB do filtro de partículas onde é mostrada a inicialização do filtro de partículas .

Fonte: print screen da tela do MATLAB

.Como a fusão sensorial foi feita de forma off-line, foi obtido junto com as informações da odometria e GPS a hora exata em que esses dados foram captados. Assim tem-se a hora em que a odometria e o GPS se coincidem. Então é a partir dessa informação equivalente que o filtro de partículas se baseia para realizar a fusão. Ou seja, são nesses pontos que as partículas são analisadas e são calculados seus determinados pesos para que assim o filtro possa informar um melhor resultado do trajeto que foi realizado. O código em que é feito essa comparação da hora e o calculo dos pesos pode ser analisado na figura 17.

```
if(h(i) > hgps(auxgps))
    auxgps = auxgps+1;
    pesos = zeros(n,1);
    for j = 1:n
       pesos(j) = exp(-(x(i,j)-xg(auxgps-1))^2/1)*exp(-(y(i,j)-yg(auxgps-1))^2/1);
    end
```

Figura 17 – Parte do código do filtro de partículas em que é feito comparação da hora e o cálculo dos pesos.

Fonte: print screen da tela do MATLAB

4 - Apresentação e análise dos Resultados

Com o propósito de comparar os resultados obtidos, é mostrado na figura 18 o real trajeto feito com a câmera. Nota-se que o caminho de ida e volta pode ser observado e, portanto, a análise final pode ser feita.

Figura 18 – Real trajeto feito com a câmera

4.1 – Odometria Visual

Enquanto os dados eram gravados no percurso, era também mostrado na tela o trajeto sendo plotado em tempo real. A figura 19 mostra esse trajeto plotado no fim do experimento.

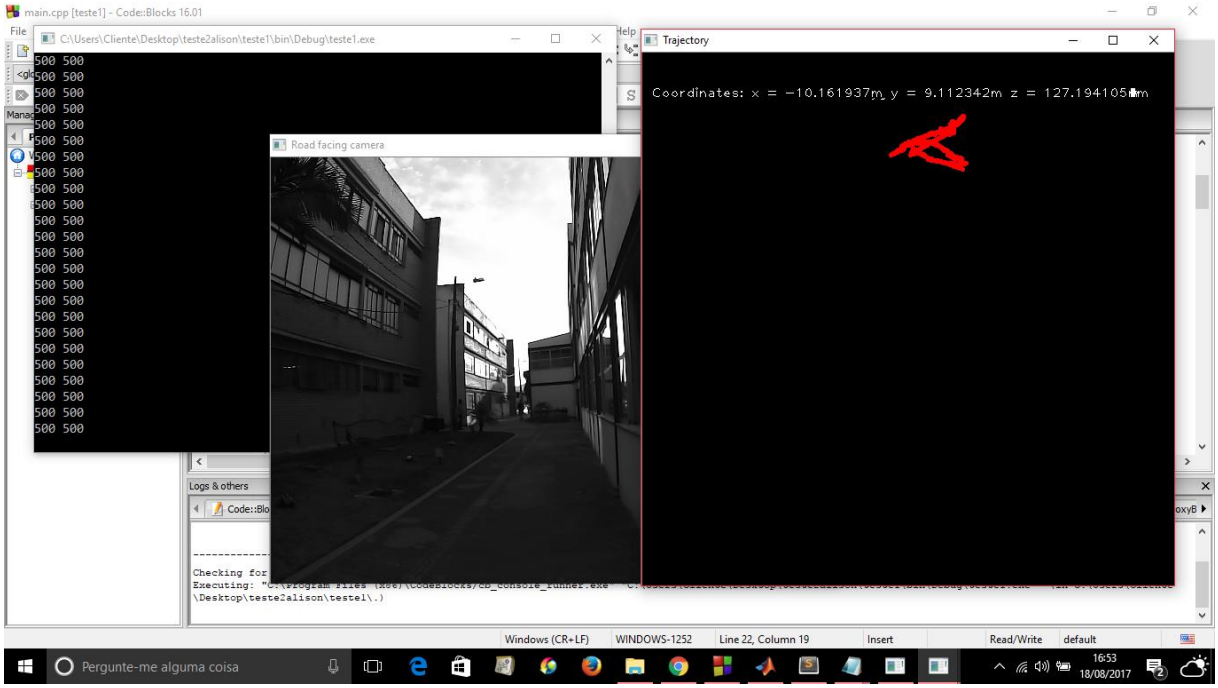

Figura 19 - Caminho plotado no fim do experimento da odometria

Fonte: print screen do programa rodando no sistema operacional Windows 10

Após a coleta dos dados da odometria, foi usado o MATLAB para que, a partir deles tentar gerar o mesmo resultado visto na figura 19. Assim, fazendo as mesmas contas que o programa no OpenCV, foi adquirido no MATLAB o caminho mostrado na figura 19.

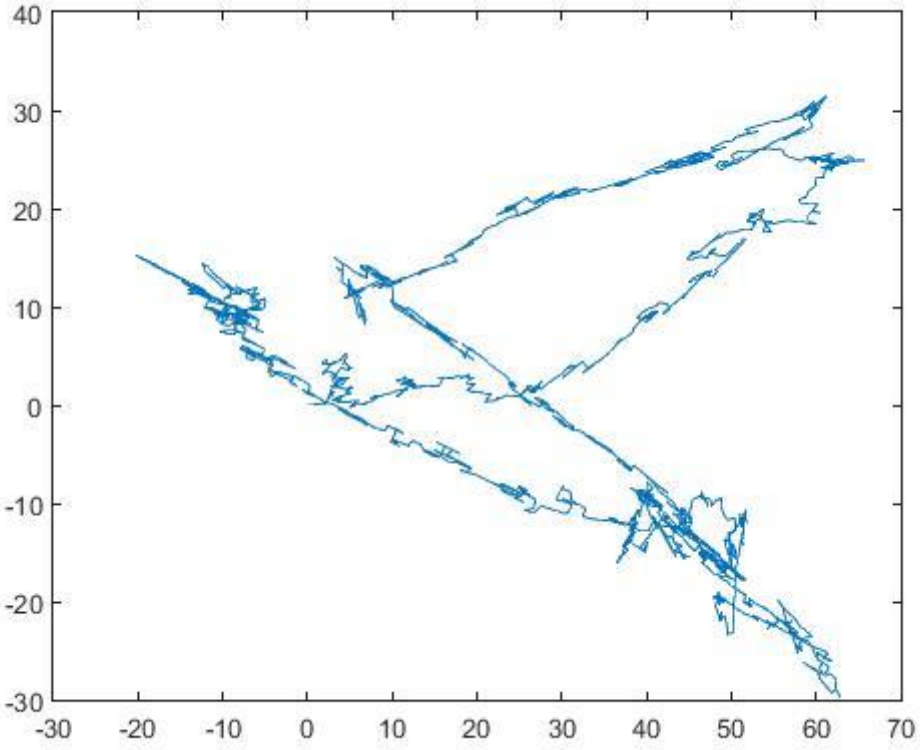

Figura 20 - Caminho plotado pelo MATLAB no fim do experimento da odometria visual
Fonte: print screen da figura plotada no software MATLAB

Analisando as figuras 19 e 20, pode-se observar que se chegou ao mesmo resultado. Nota-se que pelas figuras foi feito um caminho de ida e volta com alguns erros de direção e orientação. Isso pode ser explicado pelo alto acumulo de dados acumulados durante o experimento, o que aumenta a chance desses erros acontecerem. Portanto podemos classificar que o experimento de odometria visual foi satisfatório, pois o resultado é condizente com o que foi feito. Mesmo com essa pequena parcela de erros detectados.

4.2 - Obtenções das coordenadas GPS

Após o trajeto feito, o aplicativo Wikiloc gera um arquivo em que todos os dados de latitude e longitude coletados estão disponíveis, juntamente com a hora que esses dados foram detectados. O fuso horário do aplicativo é 5 (cinco) horas adiantado com o fuso horário de Brasília. Isso foi tratado quando esses dados foram usados. Essa parte do arquivo que é mostrado essas coordenadas é mostrado na figura 21.

```
<trkpt lat="-19.835638" lon="-43.168257">
  <ele>852.7</ele>
  <time>2017-08-18T21:45:48Z</time>
</trkpt>
<trkpt lat="-19.835732" lon="-43.168265">
  <ele>850.7</ele>
  <time>2017-08-18T21:46:00Z</time>
</trkpt>
<trkpt lat="-19.835793" lon="-43.168338">
  <ele>849.5</ele>
  <time>2017-08-18T21:46:08Z</time>
</trkpt>
<trkpt lat="-19.835863" lon="-43.168403">
  <ele>848.4</ele>
  <time>2017-08-18T21:46:15Z</time>
</trkpt>
<trkpt lat="-19.835812" lon="-43.168317">
  <ele>844.8</ele>
  <time>2017-08-18T21:46:49Z</time>
</trkpt>
<trkpt lat="-19.835773" lon="-43.168222">
  <ele>845.3</ele>
  <time>2017-08-18T21:47:46Z</time>
```

Figura 21 - Parte do arquivo em que mostra as coordenadas latitude e longitude e a hora de sua detecção com o fuso horário 5 (cinco) horas adiantado.
Fonte: Código gerado pela página do aplicativo Wikiloc

O aplicativo também gera o trajeto feito durante a detecção das coordenadas. Esse trajeto pode ser visualizado na figura 22.

Figura 22 - Trajeto plotado pelo aplicativo Wikiloc
Fonte: print screen tirado da página do aplicativo Wikiloc

Esses dados do GPS também foram jogados no MATLAB para que o trajeto fosse plotado também pelo mesmo. No MATLAB os dados de latitude e longitude foram convertidos em metros e assim puderam ser usados para que fossem analisados e comparados com o que foi dado pelo aplicativo Wikiloc. Após essa manipulação no MATLAB o caminho plotado pode ser visualizado na figura 23.

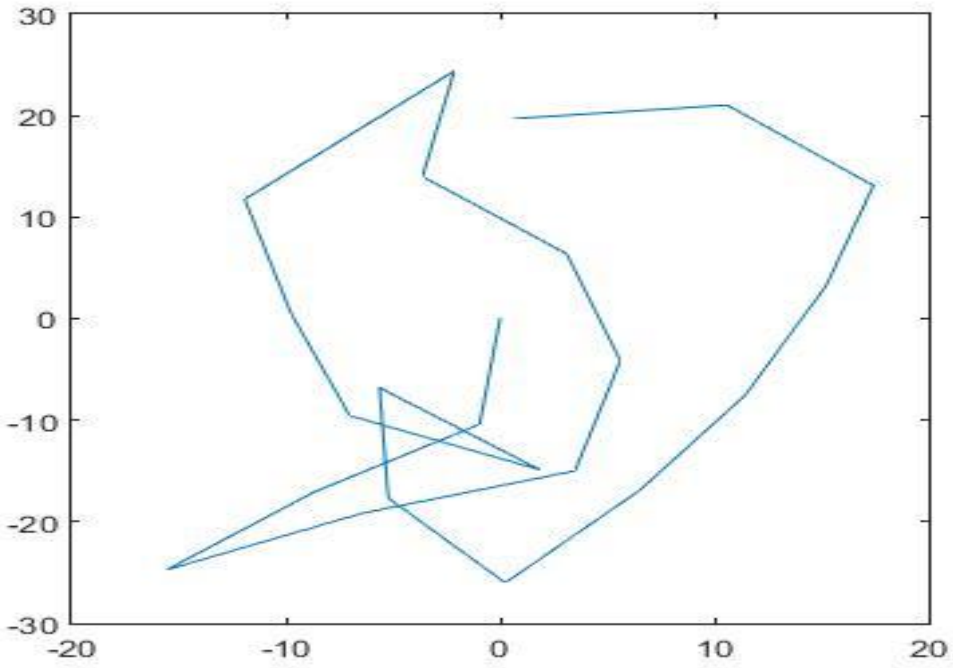

Figura 23 - Trajeto do GPS plotado no MATLAB
Fonte: print screen da figura plotada no software MATLAB

Em tese os dados do GPS seriam mais precisos, mas não é isso que podemos observar através dos resultados obtidos. Em ambas as imagens se vê um erro de direção e não notamos o trajeto de ida e volta sendo feito. Isso pode ser explicado pelo fato do erro do GPS ter interferido muito no resultado final. Pois o trajeto feito foi curto diante do erro que ele detecta. Esse erro poderia ser menor se a detecção fosse feita com um veículo mais rápido de modo que o erro não interferisse tanto ou se fosse usado para detectar a latitude e longitude um GPS diferencial. Mesmo com esse resultado não satisfatório, a fusão sensorial dos dois resultados foi feita e analisada.

4.3 – Fusão Sensorial

Após as informações da odometria visual e do GPS coletados foi feito uma fusão sensorial para que houvesse uma melhora significativa em que o resultado fosse melhor que os dois já mostrados anteriormente. Esse resultado final pode ser analisado na figura 24.

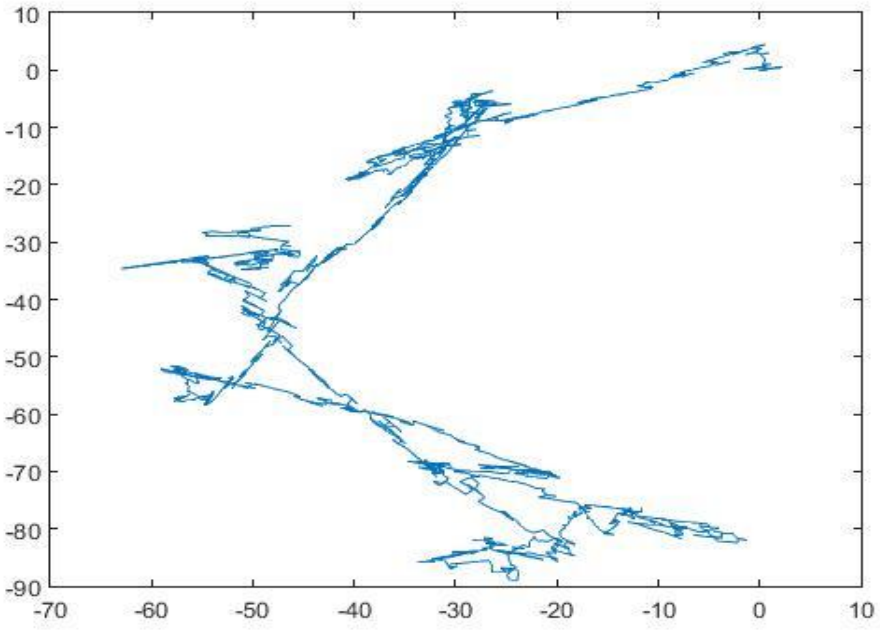

Figura 24 - Resultado da fusão sensorial plotado no MATLAB
Fonte: print screen da figura plotada no software MATLAB

Como podemos ver na figura 24, o resultado da fusão sensorial não foi melhor que o da odometria visual mostrado na figura 20. Isso porque, como foi notado antes, o resultado do GPS foi muito inferior ao da própria odometria visual. Com essa diferença de qualidade entre os dois resultados, o filtro não gera o resultado satisfatório que era esperado. Se os erros do GPS fossem menores, o resultado da fusão sensorial poderia ter sido mais satisfatório.

5 - Conclusões

Com relação ao objetivo do trabalho, faz-se necessário que ambos os experimentos realizados, sendo eles a obtenção dos dados da odometria visual e do GPS ocorram com o menor número de erros possíveis. Observando os resultados, percebe-se que a odometria teve um resultado satisfatório, em ralação ao acumulo de erros apresentados. Mas também é notório que o resultado da captura das coordenadas de latitude e longitude do GPS não foi satisfatório para que pudéssemos fazer a fusão destes dados com os coletados na odometria visual.

Essa improvável incoerência dos dados do GPS foi resultado do alto grau de erros obtidos na captura da latitude e longitude com base no trajeto realizado. Um dos motivos em que esse erro é explicado se deve ao fato de que o trajeto feito foi curto em relação à distancia detectada pelo erro do GPS. Com isso os erros se sobressaíram sobre os dados de longitude e latitude que estavam sendo detectados. Outro motivo para explicar o erro se dá pela velocidade que o trajeto foi percorrido. Esse erro poderia ter sido menor se a detecção dos dados fosse realizada por meio de um veículo mais rápido, de modo que o erro não interferisse tanto assim na detecção. Por fim, outro motivo seria a utilização de um GPS diferencial para a detecção da latitude e longitude, pois poderia, também, diminuir o erro que foi encontrado no GPS comum.

Com base nisso, espera-se a realização de um trabalho futuro em que esse erro detectado pelo GPS fosse tratado, utilizando algum método para tentar melhorar a captura da latitude e longitude referente ao trajeto feito no experimento do trabalho. Assim, tanto os dados da odometria visual e do GPS seriam válidos para que a fusão sensorial por meio do filtro de partículas pudesse ser feito. Com isso espera-se gerar um resultado melhor que ambos os realizados nesse trabalho, de forma a melhorar as informações de localização que temos atualmente.

Referências

THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. **Probabilistic Robotics.** Cambridge: The MIT Press, 2000.

THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. **Probabilistic Robotics.** Cambridge: The MIT Press, 2006.

NISTÉR, David; NARODITSKY, Oleg; BERGEN, James. **Visual Odometry for Ground Vehicle Applications.** New Jersey: Sarnoff Corporation, 2005.

FORSYTH, O.; PONCE, J. **Computer Vision: a modern approach.** New Jersey: Prentice Hall, 2012.

BALLARD, Dana; BROWN, Christopher. **Computer Vision.** New York: Prentice-Hall, 1982.

RUBLEE, Ethan; RABAUD, Vincent; KONOLIGE, Kurt; BRADSKI, Gary. **ORB: na eficiente alternative to SIFT or SURF.** Menlo Park: Willow Garage, 2011.

NISTÈR, David. An Efficient Solution to the Five-Point Relative Pose Problem. **IEE Transactions on pattern analysis and machine inteligence,** 26, 6, 756 a 770, junho, 2004.

MARQUES, C.C.S.C. **Um sistema de Calibração de Câmera.** 2007. Dissertação de mestrado – UFAL, Maceió, 2007.

MILESKI, Y.R; KOPACEK, M.C. M; AMORIM, H.J. Calibração de sistemas ópticos para uso em visão computacional. Porto Alegre: UFRGS. 2014. Disponível em: < https://www.lume.ufrgs.br/bitstream/handle/10183/112608/Poster_37319.pdf?sequence=2 >. Acesso em: 10 de maio de 2017.

MEIRELES, Rui. Interface com computador por Controlo Visual de Cursores. Disponível em: < https://paginas.fe.up.pt/~ee97107/Relatorio_de_Projecto_FINAL_PARTE_2.pdf >. Acesso em: 12 de maio de 2017.

MAZON, Hugo; ZACCHI, Giancarlo; MARTINS, Ricardo. Calibração de câmeras e fontes de erros para triangulação fotogramétrica. Florianópolis: UFSC. 2011. Disponível em: < http://www.dsr.inpe.br/sbsr2011/files/p1602.pdf >. Acesso em: 13 de maio de 2017.

SINGH, Avi. Monocular Visual Odometry using OpenCV. Avin Singh's blog. 2015. Disponível em: < https://avisingh599.github.io/vision/monocular-vo/ >. Acesso em: 8 de junho de 2017.