Instituto Politécnico de Beja

Estudo do funcionamento e da complexidade computacional do algoritmo A*

Estruturas de Dados e Algoritmos

Escola Superior de Tecnologia e Gestão
Engenharia Informática

Marco Filipe do Carmo Sacristão (11907)
5 de Junho de 2014

Índice

1. Introdução

No âmbito da unidade curricular de Estruturas de Dados e Algoritmos foi definido como trabalho prático pelo docente Prof. Jasnau Caeiro a realização de uma aplicação que visa a utilização do algoritmo de pesquisa heurística por grafos **A*[1]**, numa imagem de formato **PGM[2]**.
A utilização desse mesmo algoritmo deve ser efetuada sobre uma imagem, de forma a encontrar o melhor caminho entre dois pontos nessa imagem, através do seu custo em distância, e utilizando a cor mais próxima entre os pontos iterados. A leitura da imagem é efetuada de modo a saber o seu tamanho e conteúdos, utilizando a biblioteca **OpenCV[3]**, a qual permite a leitura da imagem em formato *PGM* permitindo a sua transformação numa imagem de outro formato de modo a saber o valor da cor em cada pixel.

1.1. Objetivos e motivação

Este trabalho prático tem como objetivo a programação do algoritmo **A*** a ser utilizado, na linguagem de programação **Java[4]**, sendo esta a linguagem de programação escolhida é também necessário a implementação básica de uma **GUI[5]** de maneira a demonstrar os resultados obtidos sobre a imagem digital a ser lida em conjunto com a biblioteca **OpenCV**.
Após a programação do algoritmo **A*** e demonstração do seu correto funcionamento, deverá ser feita a exportação do tempo que o mesmo demorou a encontrar o melhor caminho possível, analisando assim os resultados obtidos.

1.2. Contributos

O trabalho prático foi apenas desenvolvido por um elemento como descrito na capa do mesmo, sendo assim não existiu divisão de tarefas para resolução do mesmo, logo, todo o produto resultante é de mérito do autor apenas. (Excluindo informações obtidas através de terceiros, e a referencias bibliográficas enumeradas na bibliografia, sendo estas contribuições externas)

---

[1] Portable Graymap, Netpbm - *http://netpbm.sourceforge.net/doc/pgm.html*
[2] A Star search algorithm - *http://pt.wikipedia.org/wiki/A**
[3] Open Source Computer Vision Library - *http://opencv.org/*
[4] Java programming language - *http://pt.wikipedia.org/wiki/Java_programming_language*
[5] Graphical User Interface - *http://pt.wikipedia.org/wiki/Graphical_user_interface*

1.3. Estrutura do documento

Este documento de relatório encontra-se divide-se em quatro partes essenciais, sendo as mesmas:

→ **Parte teórica:** Abordagem teórica á resolução do trabalho prático
→ **Parte experimental:** Explicação dos algoritmos realizados e análise dos resultados dos mesmos.
→ **Conclusão:** Síntese de todo o contendo do trabalho prático.
→ **Bibliografia:** Enumeração de locais consultados para obtenção de informação pertinente ao trabalho prático.

2. Parte teórica

2.1. Introdução

A resolução deste trabalho resume-se na sua essência a duas partes principais, a primeira será então, a leitura de uma imagem com o formato *PGM*, retirando a informação necessária da mesma, sendo essa a intensidade da cor em escala de cinzento nesse **pixel**[1] *(PGM **Parsing**[2])*. Com esta informação é então possível passar á segunda parte, calcular através do algoritmo *A** qual o melhor caminho caminho entre dois pontos pré definidos andando de *pixel* em *pixel* com o menor distância ao ponto final e com um valor de cor semelhante ao seu.

2.2. *PGM parsing*

O formato *PGM* permite-nos saber através do seu cabeçalho (por esta ordem de leitura em linhas):

→ **Magic number:** Permite-nos saber qual o tipo de ficheiro que estamos a lidar, o *magic number* de uma imagem com o formato *PGM* é "**P5**".
→ **Altura e Largura:** Diz-nos através do formato de dois números inteiros separados por um espaço qual a altura e largura da imagem.
→ **Valor Máximo:** Qual o valor máximo possível que um pixel pode assumir em cor, em que o valor mais próximo de zero será branco, e o valor mais próximo do valor máximo apresentado será preto (pois trata-se de uma imagem em *Grayscale*).
→ **Valores:** As restantes linhas do ficheiro contém então um valor de zero até ao valor máximo, e existem portanto Altura × Largura valores contidos até ao fim do ficheiro.

Portanto, a primeira parte do nosso trabalho prático seria então analisar o ficheiro *PGM* como se de texto se tratasse, apenas obtendo apenas o seu cabeçalho.

---

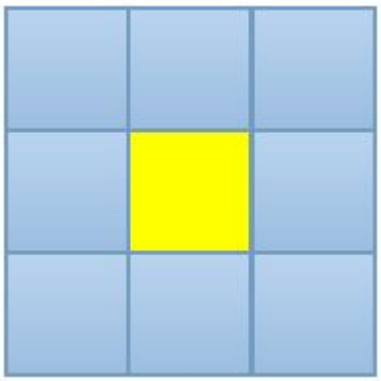
[1] *http://pt.wikipedia.org/wiki/Pixel*
[2] *http://pt.wikipedia.org/wiki/Parsing*

2.3. Conversão PGM para PNG

Não existindo suporte para nativo para *Java* no que toca a ficheiro *PGM* é necessário recorrer à biblioteca *OpenCV* e utilizar a mesma para gravar a imagem em *PNG* de modo a ser possível utilizar a imagem e apresenta-la na janela. A leitura da cor do *pixel* apesar de ser feita na *matriz* realizada pelo ficheiro *parsing* do ficheiro *PGM*, o mesmo não dispõe de uma maneira que seja possível ler o *pixel* numa determinada coordenada *x,y* portanto, teremos de utilizar novamente o formato *PGM* (usando a biblioteca *OpenCV*) para ler *pixel* a *pixel* depois de recebida a lista que contém os *pixeis* (descritos por coordenadas em *x,y).*

2.4. A*

O A* é um algoritmo de *pathfinding*[1], permite-nos calcular o melhor caminho entre dois pontos (evitando eventuais paredes ou obstáculos), neste caso, tendo em conta o valor da cor, o mesmo procura vizinhos que rodeiam o pixel num valor desse ponto e *x-1* até *x+1,* e y-1 até *y+1,* como demonstrado na imagem abaixo (*pixel* atual representado pela cor amarela, sendo os azuis a vizinhança do mesmo).



A formula base para o calculo é (F = G + H) em que H representa distância do ponto inicial até ao ponto final.

[1] http://en.wikipedia.org/wiki/Pathfinding

3. Parte experimental

3.1. Dispositivos utilizados

A compilação deste projeto em *Java* teve como alvo uma máquina com sistema operativo *Linux* tendo sido gerido o mesmo no **IDE**[1] ***Eclipse***[2].
As características da máquina são as seguintes:

→ **CPU:** Intel Core i7 3537U
→ **RAM:** 8GB
→ **Disco:** Crucial m4 128GB (6Gb/s)

3.2. Sistema experimental

O sistema experimental em *Java* está divido em duas classes *GUI.java* e *AStar.java*. Esta é a divisão mais simples possíveltendo em conta o paradigma da **orientação a objectos**[3]. A classe *GUI* é responsável pela criação e apresentação da janela, assim como de geração das imagens necessárias.

4. Conclusão

5. Bibliografia

[1] *http://en.wikipedia.org/wiki/Integrated_development_environment*
[2] *http://en.wikipedia.org/wiki/Eclipse_%28software%29*
[3] *http://pt.wikipedia.org/wiki/Orienta%C3%A7%C3%A3o_a_objetos*