Fundamentos da Programação - Projeto 5

Autores

Gabriel Franco: 2240297

Dahan Schuster: 2301792

Sobre a contribuição dos integrantes

A equipe se reuniu para discutir a elaboração do projeto e juntos contribuímos para criação de seu algoritmo. Como o trabalho foi feito no Replit, o código poderia ser testado e alterado por ambos da dupla sempre que algum problema surgisse ou caso uma das partes imaginasse uma forma de incrementá-lo.

Ideia geral do programa

A ideia central do projeto é separar a imagem em *clusters* (aglomerações) e fazer a média das posições (x, y) dentro dessa aglomeração para encontrar o centro, e usar desses centros para calcular o ângulo usando arco tangente.

Para identificar os *clusters*, o programa percorre a matriz até achar um elemento que tem valor maior ou igual à constante **MIN_VAL**[1]. Quando um elemento desses é encontrado, seu valor é zerado (a fim de evitar repetições) e é iniciada uma busca recursiva por outros valores maiores ou iguais ao **MIN_VAL** perto dele [do elemento]. Essa busca estende-se através de um raio igual à constante **N_VIZINHOS**[2], partindo do centro. Caso existam valores que passem pela marca de claridade, o *cluster* aumenta de tamanho e continua a procurar por outros valores por recursão até que não exista mais nenhum pixel com intensidade maior ou igual a **MIN_VAL** nas suas redondezas.

Para obter o ângulo basta calcular o arco tangente entre os dois centros. Para isso é necessário subtrair as coordenadas. y/x da direita – y/x da esquerda pois o eixo vertical possuí referencial positivo para baixo.

1. **MIN_VAL**: Esta constante indica o valor mínimo que um elemento (píxel da imagem) deve ter para ser considerado “claro”, uma vez que o objetivo do programa é buscar pontos de claridade. Seu valor foi refinado com base em testes e na análise dos resultados.

2. **N_VIZINHOS**: Outra diretiva que determina o comportamento do programa, informando o raio de busca de píxeis claros ao redor de um elemento. Quanto maior esse valor, mais tolerante o algoritmo será em relação à distância entre um píxel claro e outro, permitindo *clusters* com mais ruídos. Por outro lado, valores menores dificultam a busca de *clusters* em imagens de menor nitidez.

Sobre os defafios

O maior desafio foi obter valores precisos. Como este programa varia muito de acordo com **MIN_VAL** e com **N_VIZINHOS** escolhidos, então dependendo do valor das Macros o programa pode obter tanto

resultados imprecisos e lentos quanto razoavelmente precisos e rápidos. Através de alguns testes, conseguimos estabelecer os valores que melhor se adéquam ao cálculo, entretanto, observamos que é bem raro obter centros precisos (zero de erro).

Em um dos encontros da equipe, tentamos alterar o programa para incrementar sua performance, sobretudo, sua precisão. Após aplicarmos estratégias distintas, obtivemos um algoritmo mais preciso, porém com a desvantagem de ser extremamente demorado em relação aos anteriores (6 vezes mais lento!). Então optamos por utilizar o código mais rápido, pois acreditamos que um erro médio de 1.2 pixeis seja aceitável para que o controle ainda possua boa precisão de movimento e a maior velocidade de processamento diminui o *delay*, compensando, de algum modo, sua imperfeição.

Sobre a superação dos desafios

Para encontrar valores mais precisos, testamos alguns valores de **MIN_VAL** e **N_VIZINHOS** e obtivemos (baseando-se em tentativa e erro) 128 para o valor de intensidade (**MIN_VAL**) e 5 para o raio da busca (**N_VIZINHOS**) como os valores mais optimizados. Esse conjunto fornece um algoritmo que obtém erros mínimos em relação às outras configurações e uma melhor performance, gerando resultados constantes e próximos ao esperado em tempo satisfatório, porém erros são cometidos em quase 100% dos testes, variando entre 1.00 e 1.41 na maioria das vezes.

A versão mais precisa, feita como uma tentativa de melhorar a primeira versão, conseguiu reduzir o valor do score em até aproximadamente 0.5 pontos, o que nos deixou animados com a ideia. Porém, após mais algumas horas de esforço, vimos que a precisão alcançada não valia a perda de performance: ao passo que o melhor score foi de ~1.4, o tempo de processamento nesse cenário foi 6 vezes maior que a versão inicial, que chegou a alcançar ~1.9 pontos com tempo de 361 unidades de medida