Instituto Federal do Norte de Minas Gerais

IFNMG – Campus Montes Claros

Bacharelado em Ciência da Computação

Disciplina de Visão Computacional

Computação Gráfica

Transformações Geométricas 2D

Iarah Gonçalves de Almeida

David Walter Jansen (davidwalterjansen@gmail.com)

Última atualização: 26 de maio de 2019

Sumário

Capítulo 1

Descrição do Problema

A seguinte descrição de problema é referente ao trabalho prático para transformações geométricas 2D em imagens, proposto pelo Professor Wagner Ferreira de Barros na Disciplina de Computação Gráfica do curso de Ciência da Computação (IFNMG - Campus Montes Claros):

Escrever um programa em **C/C++** que realize as seguintes transformações sobre imagens: translação, rotação, escala (não uniforme, isto é, com fatores independentes para as direções x e y) e cisalhamento.

O programa deve funcionar com mapeamento reverso e implementar pelo menos dois dos três tipos de amostragem vistos em aula (pontual, bilinear (ou triangular) ou gaussiano). A imagem resultante pode ter tamanho diferente da imagem original e, portanto, antes de realizar a transformação é preciso calcular o tamanho da imagem resultante.

O sistema desenvolvido deve apresentar um menu que permita ao usuário:

(1) ler uma imagem do disco.
(2) realizar sobre ela quaisquer transformações desejadas.
(3) escolher o tipo de amostragem.
(4) gravar a imagem resultante.

Note que o usuário poderá aplicar varias transformações em sequência antes de gravar o resultado. Para ler e gravar os arquivos, utilize a biblioteca de imagens, disponibilizado no material suplementar da disciplina.

Capítulo 2

Solução Proposta

Para atender as especificações do problema, desenvolvemos um programa em **C/C++**, utilizando a IDE Code::Blocks na versão 17.12, integrada ao compilador g++ para Windows (MingGW GCC-8.2.0). O projeto pode ser encontrado anexo ao envio desse documento ou em https://github.com/DWalterJansen/CG/tree/master/GeometricTransformations2D. Para o segundo caso, basta solicitar aos autores desse trabalho a permissão de acesso ao repositório.

2.1 Bibliotecas Utilizadas

Além das bibliotecas da linguagem C++, no projeto incluímos duas bibliotecas fornecidas pelo professor da disciplina: image.h [2], wMatrix.h [3]. Outras duas bibliotecas foram criadas: transformation2D.h e colormod.h.

transformation2D.h

A função dessa biblioteca é fornecer uma classe para gerar e compor matrizes necessárias para transformação geométricas em 2D. A Classe **Transformation2D** utiliza duas matrizes: mt_show é a matriz que mostra a composição de transformações a nível do que o usuário deseja fazer; mt_real contém composições adicionais que movem a imagem para o quadrante de visualização, ou seja, $x \in [0, width)$ e $y \in [0, height)$. Logo, se o usuário deseja realizar uma rotação de θ, por exemplo, as matrizes serão dadas por 2.1 e 2.2:

$$mt\_show = \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

$$mt\_real = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

onde tx e ty correspondem ao valor da translação necessária, em seus respectivos eixos, para a imagem não ter nenhum pixel com coordenada negativa. Após gerada a matriz mt_real, calculasse o novo tamanho da imagem após a transformação. Esse procedimento é realizado pelo método **compositionReal**.

É importante salientar que, para fins de otimização, os cálculos de correção das coordenadas e do novo tamanho são sempre realizados com base nos 4 pontos que definem o perímetro da imagem.

colormod.h

Essa biblioteca apenas fornece uma classe para modificar a coloração dos textos de saída. Usamos a cor vermelha para restrições de entrada; azul para ressaltar os parâmetros definidos pelo usuário, tais como: nome do arquivo de leitura, matriz de transformação (mt_show), tipo de amostragem; verde para alertas de ordem nos passos, por exemplo, uma imagem só pode ser salva após definir seu tipo de amostragem.

2.2 main.cpp

A função principal do nosso programa contém o menu pedido para esse trabalho, além das funções para tratar cada operação do menu. As funções tem o padrão de nome como *option*@, onde o @ é substituído pelo número correspondente ao número da opção no menu (2.1).

Menu

A figura 2.1 mostra o resultado do menu logo após o início da execução do programa. Note que o menu começa com a matriz identidade para a transformação, ou seja, o equivalente a criar uma cópia da imagem de leitura. As informações exibidas no menu são atualizadas conforme o usuário define as configurações das transformações, como pode ser visto no capítulo 3.

Option1

A opção de leitura da imagem solicita do usuário o nome do arquivo para ser aberto. Sempre que essa função é chamada, a matriz de transformação sofre um *reset* (volta ser igual a identidade).

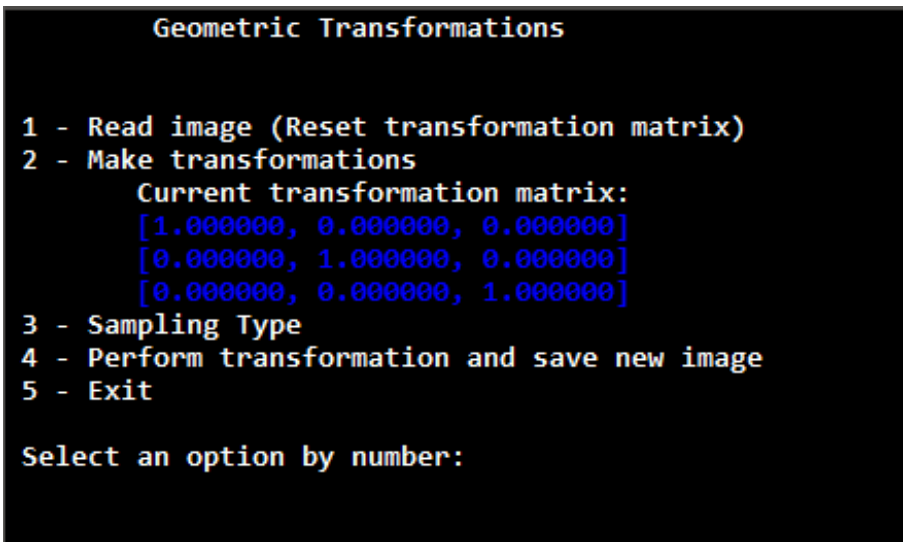

Figura 2.1: Menu logo após o início da execução

Isso é importante porque a *mt_real* faz correções com base nas dimensões da imagem de entrada, e esse processo só pode ser feito partindo-se da matriz inicial, ou seja, da matriz identidade.

Option2

A opção de criar a matriz de composição solicita do usuário a quantidade de transformações, bem como o tipo de cada transformação. Para cada tipo é necessário informar os parâmetros da transformação. Dentro dessa opção ocorre a chamada de métodos para construir as duas matrizes de composição (*mt_show* e *mt_real*).

Option3

A opção para o tipo de amostragem permite ao usuário selecionar, dentre os dois tipos de mapeamento reverso implementados, qual deseja utilizar para preenchimento dos pixels na imagem transformada. Foram implementados os mapeamentos reversos: Pontual e Bilinear.

Option4

A opção de realizar as transformações e salvar o resultado, aplica sobre a imagem de entrada uma única vez a *mt_real*, que contém a composição de todas as transformações pedidas. A forma de mapeamento reverso dos pixels é aquela escolhida pelo usuário. Além disso, a transformação aplicada é diferente para o caso da imagem colorida e imagem em tons de cinza. Essa diferença está expressa nas funções: **reversMapColorful** e **reversMapNotColorful**. O resultado de todos esses procedimentos é salvo num arquivo com o nome informado pelo usuário. A formato do arquivo de saída tem a mesma extensão do arquivo de leitura.

Capítulo 3

Testes

Antes de mostramos os testes, é importante ressaltar que as estruturas matemáticas computacionais para representar imagens, são diferentes da representação matemática no plano cartesiano. Em outras palavras, **matrizes** em computação apresentam sua origem, ou seja, o ponto de coordenadas $(0, 0)$ no canto superior esquerdo da imagem. No entanto, representamos imagens no espaço Euclidiano com sua origem no canto inferior esquerdo da imagem. Essa simples diferença apresenta impactos na orientação das transformações. As figuras 3.1, 3.2, 3.3 e 3.4 retiradas da referência [1], mostram o resultado esperado para cada transformação.

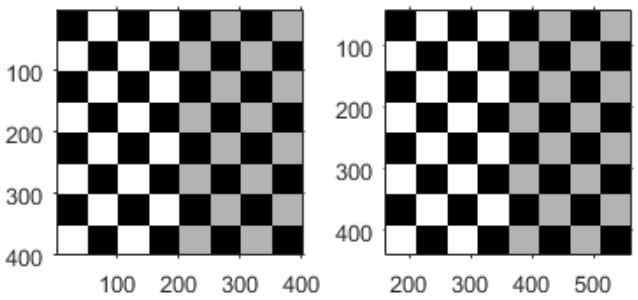

Figura 3.1: Translação em x de -150 unidades, em y de -50 unidades

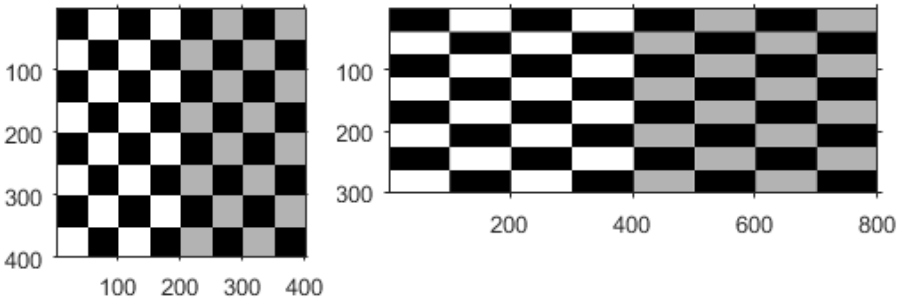

Figura 3.2: Escala por um fator de 2 em x

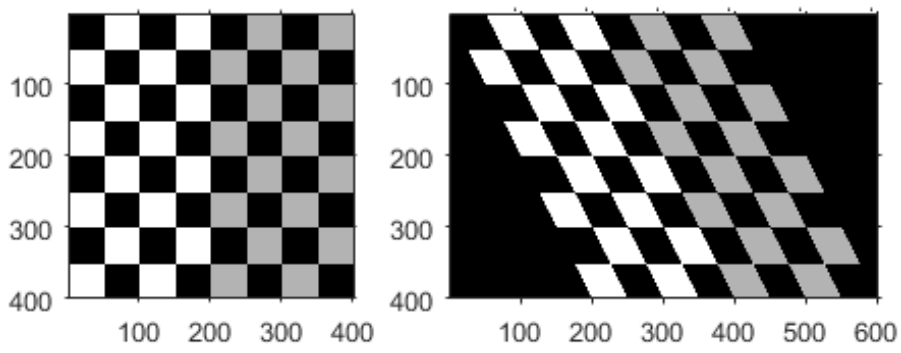

Figura 3.3: Cisalhamento na direção horizontal por um fator 0.5*y

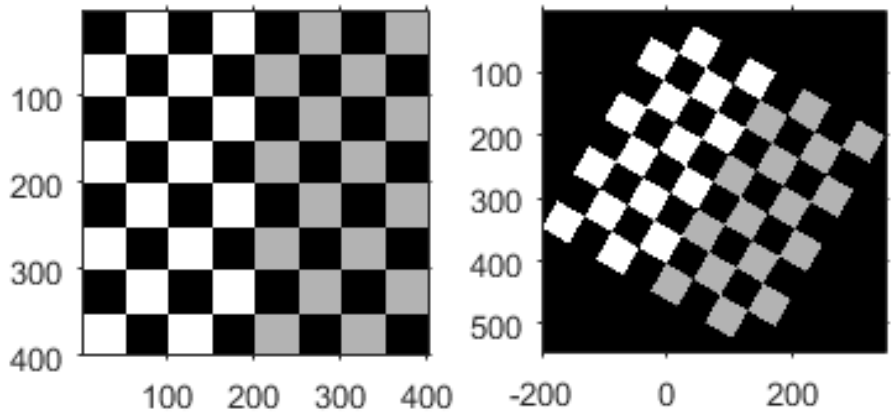

Figura 3.4: Rotação de 30º

3.1 Parâmetros dos Testes

Utilizamos a figura 3.5 no formato .jpg como base. A partir dela foram geradas duas imagens para as quais os testes serão aplicados de fato: a primeira no formato ppm raw e a segunda no formato pgm raw. Para cada uma dessas imagens realizamos a seguinte sequência de transformações geométricas em 2D: Translação de $tx = 100$ e $ty = 100$, rotação de $\theta = 45\,°$, escala de $sx = 0.5$ e $sy = 1$, cisalhamento de $shx = 0.25$ e $shy = 0$. Além disso, para cada uma das duas imagens transformadas, será feito o mapeamento Pontual e Bilinear.

A figura 3.6 exibe o resultado da matriz gerada pelo programa, que representa a composição das transformações mencionadas nessa seção. Os resultados no capítulo 4 mostram a interface do programa e as imagens obtidas.

Figura 3.5: Imagem da Lamborghini. Resolução 1920x1200

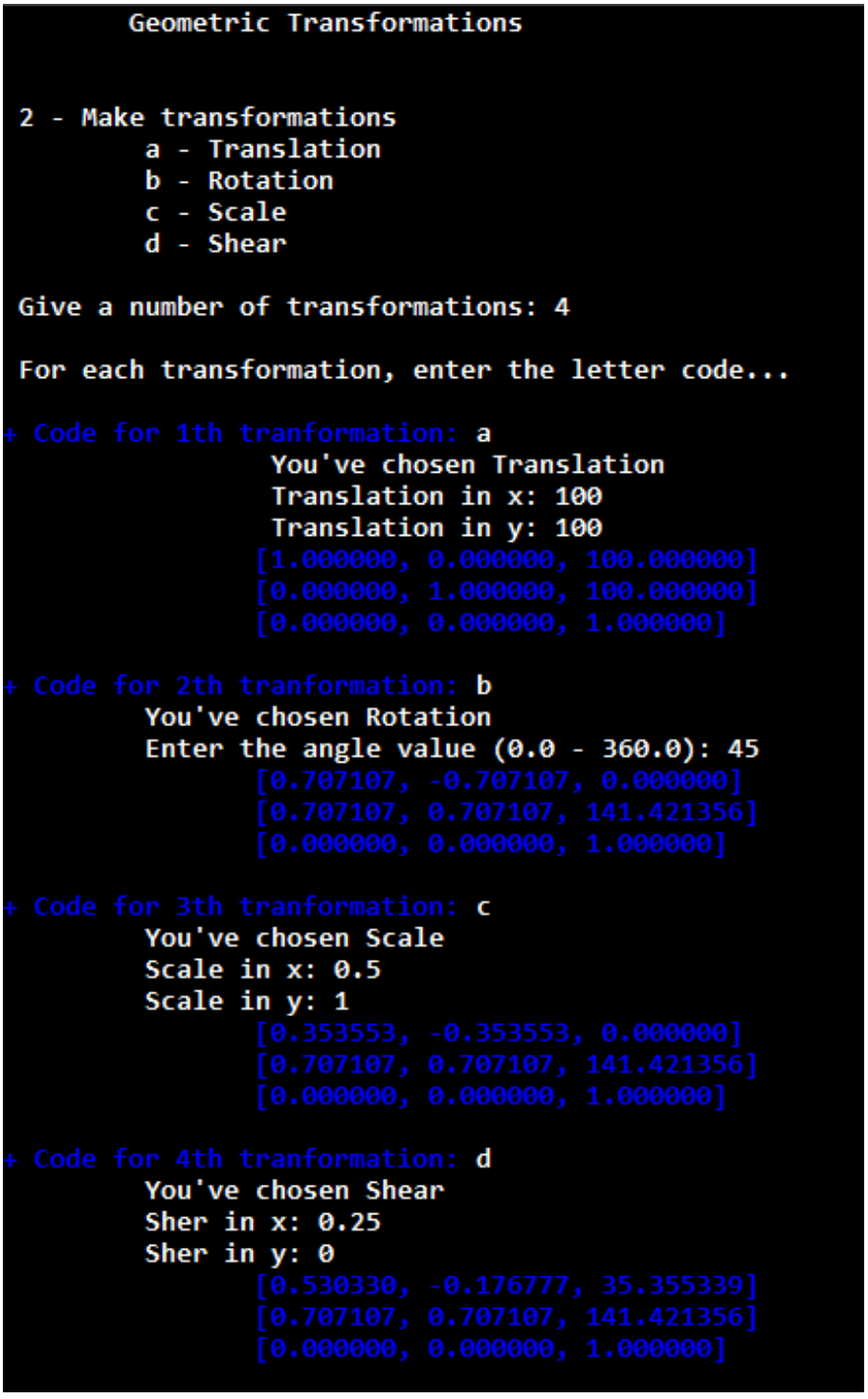

Figura 3.6: Composição de transformações geométricas em 2D para os testes

Capítulo 4

Resultados

A figura 4.1 mostra 4 interfaces do programa com os parâmetros necessários para produzir os seguintes resultados:

- Resultado para: imagem ppm, mapeamento pontual 4.2
- Resultado para: imagem ppm, mapeamento bilinear 4.3
- Resultado para: imagem pgm, mapeamento pontual 4.4
- Resultado para: imagem pgm, mapeamento Bilinear 4.5

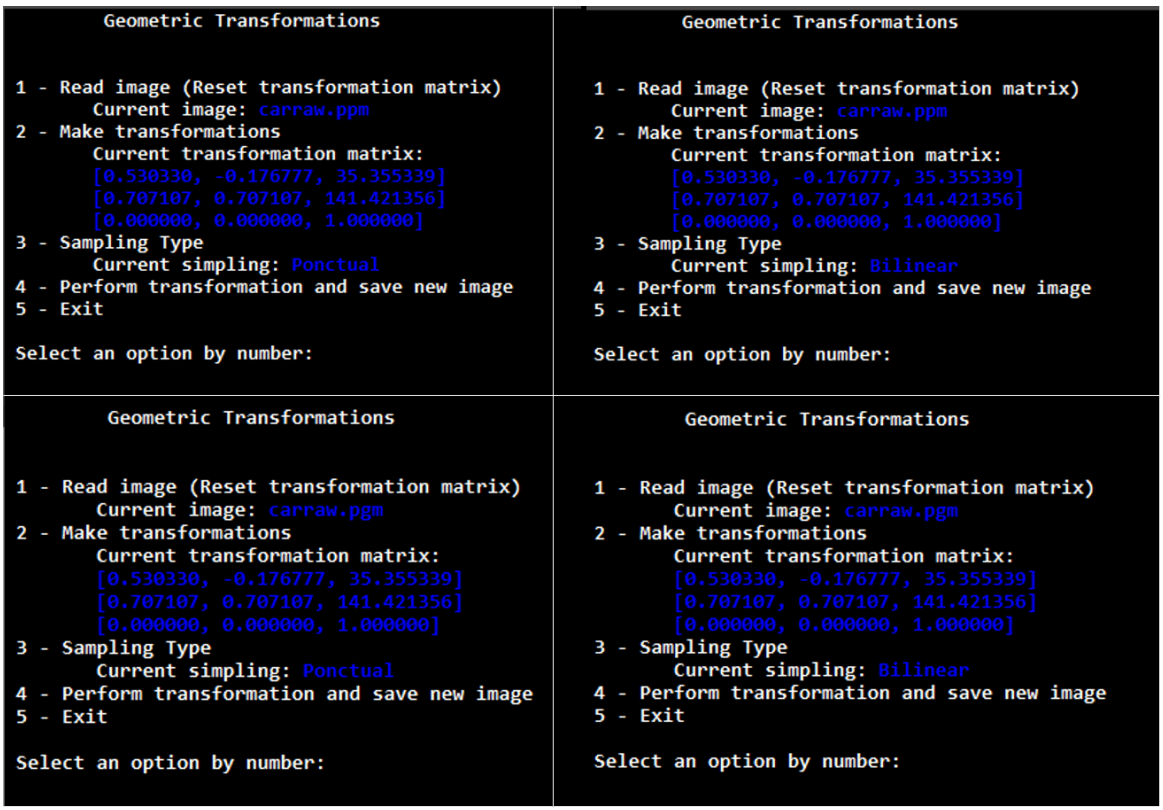

Figura 4.1: Todas as interfaces usadas para gerar os resultados

Figura 4.2: Resultado: Imagem colorida no formato ppm, usando mapeamento Pontual

Figura 4.3: Resultado: Imagem colorida no formato ppm, usando mapeamento Bilinear

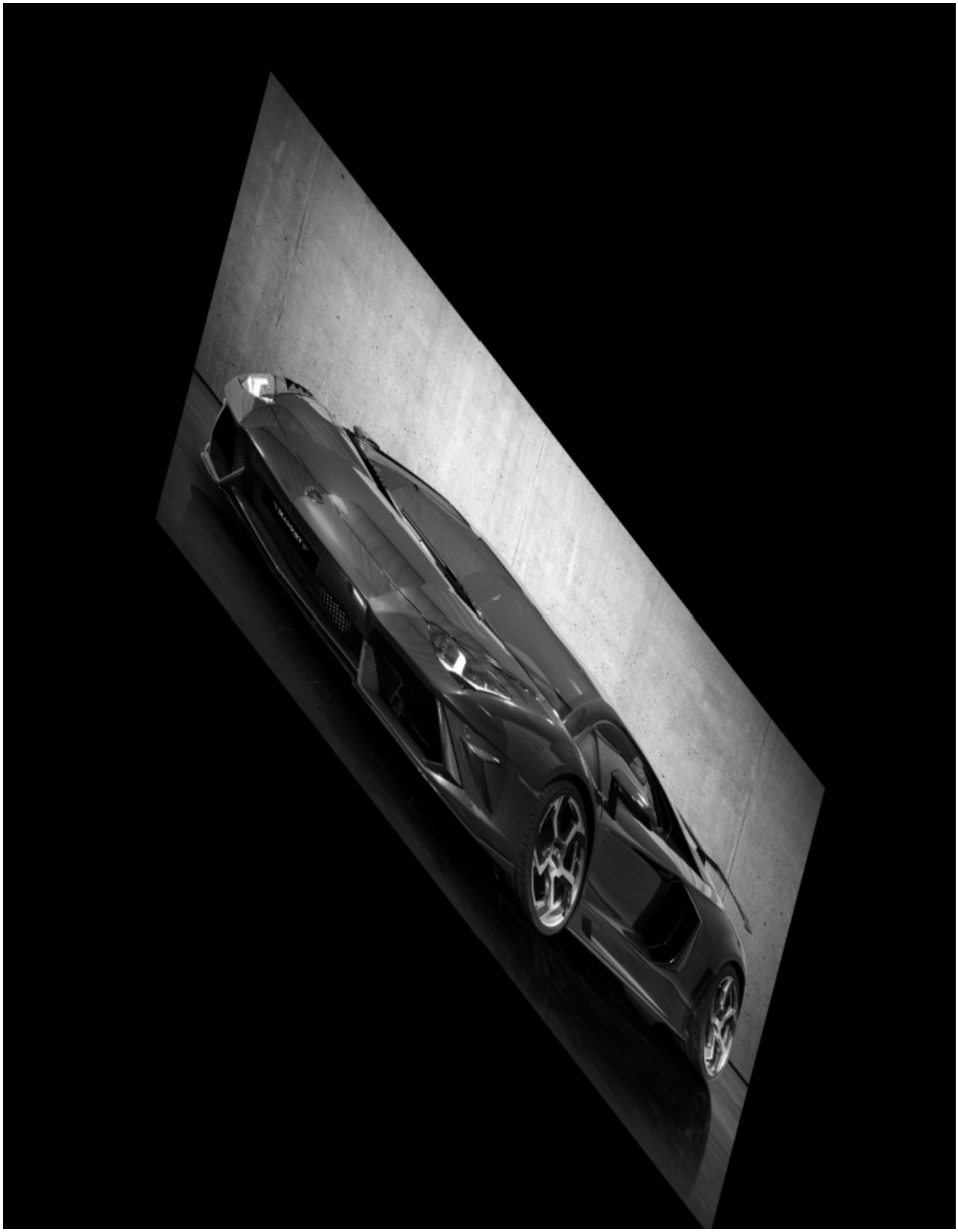

Figura 4.4: Resultado: Imagem tons de cinza no formato pgm, usando mapeamento Pontual

Figura 4.5: Resultado: Imagem tons de cinza no formato pgm, usando mapeamento Bilinear

Referências Bibliográficas

[1] Matrix representation of geometric transformations. https://www.mathworks.com/help/images/matrix-representation-of-geometric-transformations.html. Accessado: 2019-05-25.

[2] W. F BARROS. Códigos em c/c++ para leitura e escrita de imagem, 2004.

[3] W. F BARROS. Códigos em c/c++ para operações com matriz 3x3, 2004.