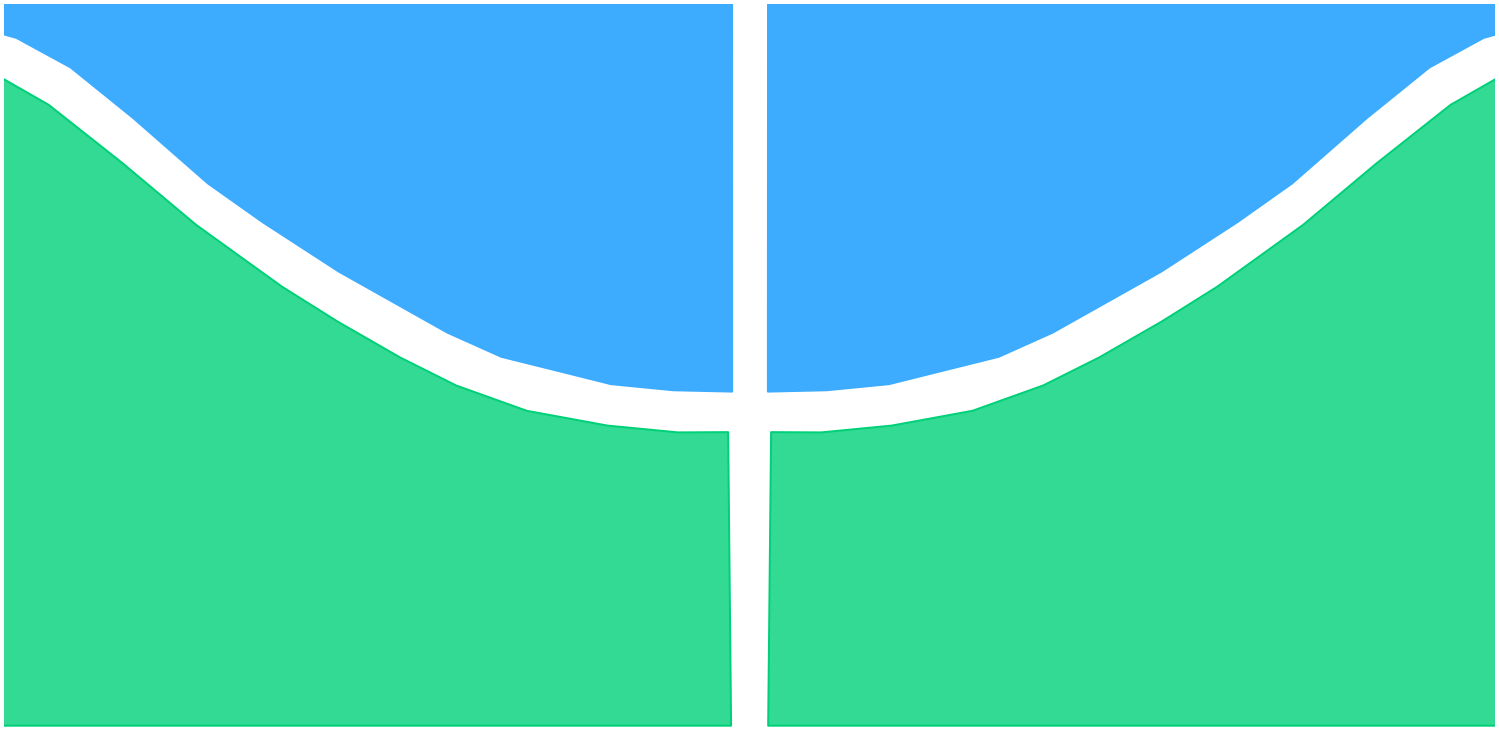

TRABALHO DE GRADUAÇÃO

FI^2 – Ferramenta de Investigação de Imagem Φ^2

Por,

André Pereira Henriques

Tauan de Oliveira Naves

Brasília, Julho de 2011

ENGENHARIA MECATRÔNICA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia
Curso de Graduação em Engenharia de Controle e Automação

TRABALHO DE GRADUAÇÃO

FI^2 – Ferramenta de Investigação de Imagem Φ^2

POR,

André Pereira Henriques
Tauan de Oliveira Naves

Relatório submetido como requisito parcial para obtenção
do grau de Engenheiro de Controle e Automação.

Banca Examinadora

Díbio Leandro Borges, UnB/ CIC (Orientador) _____

Flávio de Barros Vidal, CIC, UnB _____

Jorge de Albuquerque Lambert, SEPAEL, DPF _____

André Luiz da Costa Morisson, SEPAEL, DPF _____

Brasília, Julho de 2011

FICHA CATALOGRÁFICA

ANDRÉ, HENRIQUES, TAUAN, NAVES
FI^2, Ferramenta de Investigação de Imagem,

[Distrito Federal] 2011.

ix, 42p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2011). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Ferramenta de Investigação de Imagens 2.Software de Processamento de Imagens

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

HENRIQUES, A.P., NAVES, T.O., (2011). FI^2 , Ferramenta de Investigação de Imagens. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº 010/2011 , Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 42p.

CESSÃO DE DIREITOS

AUTOR: André Pereira Henriques, Tauan de Oliveira Naves.

FI^2 , Ferramenta de Investigação de Imagens: Software de processamento de imagens especializado em investigação forense.

GRAU: Engenheiro ANO: 2011

_____

André Pereira Henriques
SQN 206 Bloco K – Asa Norte.
70844-110 Brasília – DF – Brasil.

_____

Tauan de Oliveira Naves
SCES Trecho 2 Lote 2/41 Bloco A – Asa Sul.
70200 - 002 Brasília – DF – Brasil.

RESUMO

Trata-se do desenvolvimento de um software de processamento de imagens para investigação forense, destinado a peritos criminais e denominado FI^2 (Ferramenta de Investigação de Imagens). A ferramenta foi desenvolvida em cima das bibliotecas *OpenCV,* para o processamento de imagens, e *QT*, para a interface da aplicação, com a linguagem C++. Por se tratar de uma aplicação voltada a investigações criminais, a ferramenta registra detalhamento de todos os passos das técnicas empregadas durante a análise forense de uma imagem, de modo a robustecer sua validade em processos judiciais.

Palavras Chave: *Ferramenta de Investigação de Imagens, Software de Processamento de Imagens.*

ABSTRACT

It is presented an image processing software for forensics, to criminal experts and it is called FI² (*Ferramenta de Investigação de Imagens* – Image Investigation Tool). The tool was developed on top of *OpenCV* library, for image processing, and *QT* library, for software interface, with the language C++. Because it is an application oriented to criminal investigation, it records detailing all the steps of the techniques used during the forensic analysis of an image in order to strengthen its validity in court.

Keywords: *Tool Research Images, Software Image Processing.*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1 – INTRODUÇÃO

Fotos e vídeos são atualmente uma das principais fontes de informações para investigações. No entanto, nem sempre é possível conseguir imagens ou vídeos de boa qualidade, geralmente devido a problemas de equipamento e do ambiente. Com a tecnologia atual esses fatores não são impedimentos para se conseguir boas imagens e vídeos, para se ter provas de identificação válidas, e assim também informações úteis para as investigações.

Para a utilização dessa tecnologia é necessário que imagens e vídeos sejam digitalizados a fim de serem processados por um software dedicado a melhorar as caraterísticas úteis para a análise do material.

Características que podem afetar a qualidade da imagem:

- Baixa resolução de imagens, necessária para o aumento do tamanho dos detalhes úteis;
- Falta de contraste;
- Diferentes tipos de ruídos e perturbações;
- Tremido causado por movimento ou falta de foco;
- Distorções geométricas limitando, consequentemente, a reconstrução das dimensões dos objetos dentro da imagem (como, p. ex., dados biométricos dos indivíduos);

Com o intuito de superar essas adversidades o software FI^2 (Ferramenta de Investigação de Imagens) disponibiliza diversas ferramentas a fim de se obter uma imagem com significado relevante para o usuário. Cada operação realizada é registrada, certificada, além fornecer um conhecimento completo das ferramentas junto ao algoritmo para obter plena objetividade e garantir que o resultado possa ser reproduzido por outra pessoa.

Nos próximos capítulos será mostrada uma breve amostra do estado de arte (Capítulo 2), o sistema proposto, com sua arquitetura, ferramentas implementadas e futuras ferramentas (Capítulo 3) e, por fim, os resultados de sua aplicação em vários exemplos (Capítulo 4), concluindo com o futuro do FI^2.

CAPÍTULO 2 – ESTADO DA ARTE

Atualmente, existem vários produtos que visam o processamento de imagens e vídeos, mas poucos são destinados à área forense, dos quais apenas alguns conseguem realizar os objetivos a que se propõem. Podem ser dados como exemplo de produtos voltados a área forense: dTective by Avidand Oceans Systems[3], Impress by Imix[4], Star Witness Video by Signalscape[5], Video Analyst by Intergraph[6], Video Investigator by Cognitech[7] e Amped Five[8].

A maioria desses programas tem uma solução completa de hardware e software. Alternativamente, o FI^2 visa apenas a área de software, destinando-se ao processamento de imagem com algoritmos e técnicas que são reconhecidas na literatura especializada ou descritas e detalhadas pela própria aplicação, diferente de alguns daqueles que usam técnicas próprias sem prover acesso ao código ou ao algoritmo.

Todos esses programas têm ferramentas como ajuste de contraste e brilho, equalização de histograma e edição, zoom, espelhamento, rotação das imagens. Alguns deles oferecem ferramentas onde se pode personalizar filtros. Assim como o software aqui apresentado, quase todos armazenam um log de auditoria para cada caso realizado a fim de torná-los apropriados para apresentações em audiências. Apesar de oferecerem a capacidade de registrar as operações realizadas junto com seus parâmetros, esses softwares não dão acesso ao algoritmo aplicado.

A biblioteca de processamento de imagens OpenCV, criada pela Intel e atualmente em desenvolvimento ativo pela WillowGarage, apesar de não ser uma ferramenta destinada a área forense, é uma ferramenta inestimável para o software desenvolvido, pois além de dispor as ferramentas básicas para manipulação de imagens – e vídeos – de vários formatos, também contém modernos algoritmos de processamento nela, continuamente, implementados.

CAPÍTULO 3 – SISTEMA PROPOSTO

3.1. PROPOSTA

FI^2 é uma ferramenta de processamento de imagens especializada em investigação forense. A ferramenta foi desenvolvida em cima da biblioteca OpenCV e da biblioteca e ambiente de desenvolvimento Qt com a linguagem C++, mantendo a portabilidade entre os sistemas operacionais Linux, Windows e MacOS. Por se tratar de uma aplicação voltada para autoridades investigativas, o sistema proposto tem um registro das técnicas e manipulações realizadas nas imagens, incluindo datas, algoritmos e parâmetros, de modo a conferir sua validade em processos legais e judiciais.

Apesar de não se tratar de um software com o modelo open-source, o detalhamento das intervenções nas imagens, com indicação de documentação e artigos das técnicas aplicadas, deverão conferir a confiabilidade do sistema proposto.

As tabelas a seguir explicitam os recursos que foram implementados e os que serão futuramente implementados:

Tabela 1 – Ferramentas implementadas

Módulos	Tarefas	Observações
Módulo 1	Ferramentas Básicas	Cortar Luminância e Contraste Conversão em escala cinza Girar/Rotacionar 90°
Módulo 2	Interpolação	Vizinho mais próximo Bilinear Bicúbica Interpolação Lanczos [9]
Módulo 3	Operação de Pontos	Equalização de histograma
Módulo 4	Restauração de Imagem	Correção de perspectiva

Tabela 2 – Ferramentas a serem implementadas.

Módulos	Tarefas	Observações
Módulo 1	Ferramentas Básicas	
Módulo 2	Interpolação	
Módulo 3	Operação de Pontos	Limiarização Limiariazação adaptativa Alongamento de Contraste Operação Logarítmica Operação Exponencial
Módulo 4	Filtros	Mediana Unsharp máscara (USM) Passa-altas Rational sharpening Média Passa-baixas Suavização de ruído Redução de artefatos de bloqueio Adicionar ruídos (testes) Laplace da Gaussiana Frequência

3.2. ARQUITETURA DO PROGRAMA

O programa foi projetado com três principais objetivos norteadores: desempenho, portabilidade e escalabilidade.

3.2.1. DESEMPENHO

O desempenho é um fator crítico pelo fato de algumas técnicas de processamento de imagens terem um custo computacional elevado. Associado a isso, deseja-se resposta rápida, com o intuito de o operador obter um resultado o mais célere possível para cada alteração de parâmetro de uma determinada manipulação, facilitando, assim, a comparação entre as diversas configurações possíveis de uma determinada técnica. Para satisfazer esse requisito, decidiu-se pelo projeto de uma aplicação nativa, utilizando-se a linguagem C++, que é servida de compiladores nas mais diversas plataformas.

3.2.2. PORTABILIDADE

A portabilidade é um aspecto importante tendo em vista o reaproveitamento de códigos e a alcançabilidade da ferramenta. Para obter a almejada portabilidade, estudou-se, dentre as possibilidades, ferramentas que pudessem ser interoperadas indistintamente entre os três principais sistemas operacionais: *Windows*, *MacOS* e *Linux*. Com isso, optou-se pelas bibliotecas *OpenCV*, para o auxílio no processamento de imagens, e *QT* para o desenvolvimento da interface.

3.2.3. ESCALABILIDADE

A escalabilidade é importante pelo fato de se desejar uma ferramenta que englobe as mais diferentes e recentes ferramentas voltadas à investigação forense. Tais ferramentas têm os mais diversos requerimentos, tanto a interface quanto as características internas do programa.

Para isso, desenvolveu-se uma interface única para o código de representação gráfica de todas as ferramentas e outra interface, também única, para encapsular o código de processamento de imagens das ferramentas, flexíveis o suficiente para atenderem as demandas que as ferramentas que se deseje implementar requerem.

Na figura 3.1, diagrama UML da arquitetura simplificada do programa, representa-se apenas duas ferramentas, porém todas as disponíveis no programa seguem o mesmo modelo.

Enfim, procurou-se desenvolver uma aplicação que seja uma plataforma para que novas técnicas de processamento de imagens, em particular na área forense, fossem facilmente implementadas.

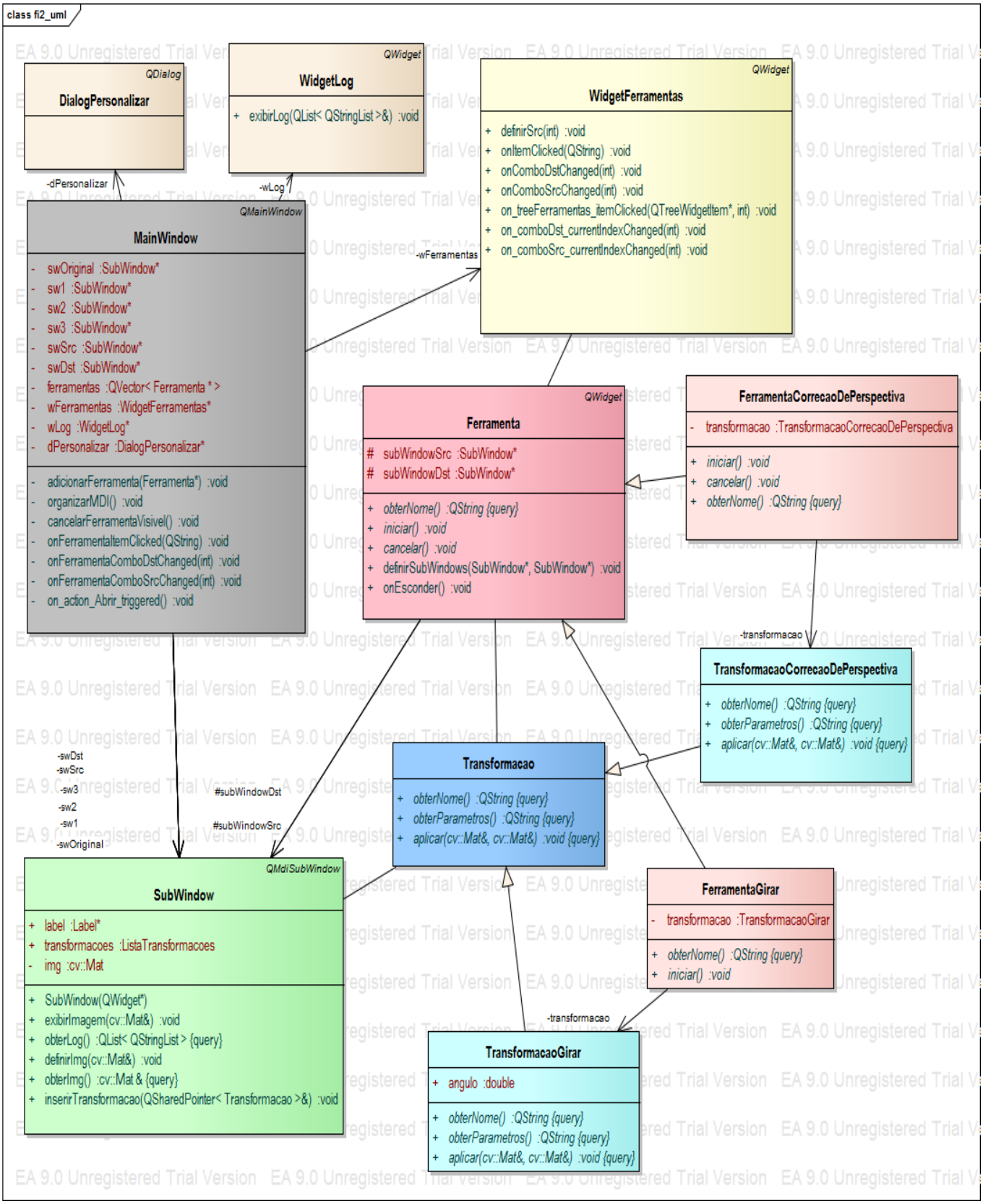

Figura 3.1. Diagrama de blocos UML simplificado do programa com dois exemplos de ferramentas.

Como é mostrado na figura 3.1, as ferramentas do programa seguem um modelo básico para a sua codificação.

A ferramenta deve conter o código de processamento de imagens. Trata-se da definição de uma subclasse de *Transformação* que requer a definição de 3 métodos virtuais:

- *obterNome() : QString* – nome da ferramenta que aparece no log;
- *obterParametros() : QString* – representação textual dos parâmetros da transformação aplicada, bem como possíveis observações;
- *aplicar(cv::Mat&, cv::Mat&) : void* – aplicação da transformação, onde o primeiro parâmetro é a representação da imagem (formato da biblioteca OpenCV) fonte, e o segundo é a imagem destino transformada.

Para a parte de interface gráfica a ferramenta deve definir uma subclasse de *Ferramenta*, implicando em definir três ou dois – dependendo das necessidades da ferramenta – métodos virtuais:

- *obterNome() : QString* – nome da ferramenta que é exibido na janela de ferramentas;
- *iniciar() : void* – rotina de inicialização da ferramenta (configuração inicial dos parâmetros gráficos);
- *cancelar() : void* – algumas ferramentas necessitam de um código de encerramento (e.g desalocação de recursos e sinais). Não é, portanto, obrigatória a definição desse método.

Além disso, a ferramenta deve conter o código de inserção da sua transformação, segundo a sua interface gráfica na janela de destino – método *inserirTransfomação(...)* da janela destino (do tipo *SubWindow*) que é definida pelo gerenciador de ferramentas para a ferramenta.

Uma vez definidas essas classes, basta se adicionar a ferramenta ao gerenciador de ferramentas – na *MainWindow* – com a chamada do método *adicionarFerramenta(Ferramenta *) : void*.

3.2.4. EXEMPLO DE IMPLEMENTAÇÃO DE FERRAMENTA

Para ilustrar a metodologia desenvolvida para a inserção de ferramentas no programa, segue o exemplo da implementação da ferramenta luminância, presente nos códigos contidos nos arquivos TransformacaoLuminancia.h, TransformacaoLuminancia.cpp FerramentaLuminancia.h e FerramentaLuminancia.cpp,– anexos 1, 2, 3 e 4 respectivamente. As elipses – "**...**" – presentes em trechos de códigos apresentados, representam código omitido, que não têm relevância na discussão.

Definições da classe *TransformacaoLuminancia* como subclasse de *Transformacao*:

```
class TransformacaoLuminancia : public Transformacao { ... };
```

Definições dos métodos necessários pela classe pai, *Transformacao,* na classe *TransformacaoLuminancia*:

```
virtual QString obterNome() const {
   return QString::fromUtf8("Lumin\303\242ncia");
}
virtual QString obterParametros() const {
   return QString("+Y = ") + QString::number(incrementoLuminancia);
}
virtual void aplicar(const cv::Mat &src, cv::Mat &dst) const { ... }
int incrementoLuminancia;
```

O parâmetro *incrementoLuminancia* é utilizado pelo método *aplicar(...)* para realizar a operação de incremento de luminância na imagem.

A seguir, as definições da interface visual da ferramenta.

Definições da classe *FerramentaLuminancia* como subclasse de *Ferramenta*:

```
class FerramentaLuminancia : public Ferramenta { ... }
```

Atributos utilizados pela classe:

```
TransformacaoLuminancia transformacao;
cv::Mat dstTemp;
```

Que representam, respectivamente, a transformação a ser aplicada na imagem e a imagem que é exibida temporariamente na janela destino – essa imagem não é a imagem destino definitiva.

Definições dos métodos necessários pela classe pai, *Ferramenta,* na classe *FerramentaLuminancia*:

```
virtual QString obterNome() const {
   return transformacao.obterNome();
}
virtual void iniciar() {
   dstTemp = subWindowSrc->obterImg().clone();
   qt_util::setValueSemSinais(ui->spinBox, 0);
   qt_util::setValueSemSinais(ui->horizontalSlider, 0);
```

```
    transformacao.incrementoLuminancia = 0;
    subWindowDst->exibirImagem(dstTemp);
}
```

O método *iniciar()* realiza as seguintes operações, na ordem em que se foi apresentado no código:

- copia a imagem de exibição temporária da imagem fonte;
- inicia os parâmetros da interface gráfica;
- inicia o parâmetro da transformação;
- exibe imagem destino temporária.

Além desses métodos, que devem ser definidos por conta da interface *Ferramenta*, devem se definir métodos que são dependentes da configuração gráfica da ferramenta. Esses métodos são específicos para cada ferramenta, mas, em geral, seguem um formato semelhante. São os métodos de alteração de parâmetro e confirmação e cancelamento de transformação.

Método de alteração de parâmetro:

```
void on_spinBox_valueChanged(int value) {
    ...
    transformacao.incrementoLuminancia = value;
    transformacao.aplicar(subWindowSrc->obterImg(), dstTemp);
    subWindowDst->exibirImagem(dstTemp);
}
```

Método de cancelamento da ferramenta:

```
void on_cancel_clicked() {
    cancelar();
    emit onEsconder();
}
```

Ao se cancelar a ferramenta, deve-se chamar o método *cancelar()* e emitir o sinal *onEsconder()*, ambos de *Ferramenta,* para se fechar a ferramenta.

Método de confirmação da ferramenta:

```
void on_ok_clicked() {
    subWindowDst->definirImg(dstTemp);
    subWindowDst->transformacoes = subWindowSrc->transformacoes;
```

```
            subWindowDst->inserirTransformacao(QSharedPointer< Transformacao
>(new TransformacaoLuminancia(transformacao)));
            emit onEsconder();
        }
```

O método de confirmação realiza as seguintes operações, na ordem em que se foi apresentado no código:

- define a imagem da janela de destino;
- copia as transformações – inclui o histórico – da janela fonte para a janela destino;
- insere a transformação da ferramenta na lista de transformações da janela destino;
- emite sinal para fechar ferramenta.

Finalmente, deve-se adicionar a ferramenta na janela principal com uma simples chamada do método de registro de ferramenta dentro do construtor da classe da janela principal *MainWindow* (figura 3.1):

```
...
    adicionarFerramenta(new FerramentaLuminancia());
...
```

CAPÍTULO 4 – PROCESSAMENTO DE IMAGENS

4.1. CORREÇÃO DE PERSPECTIVA

Dada uma imagem em perspectiva de um plano global, o objetivo é obter a retificação métrica desse plano, ou seja, obter a imagem equivalente onde o plano da câmera – observador – e o plano observado – a imagem – estejam paralelos (Figura 4.1).

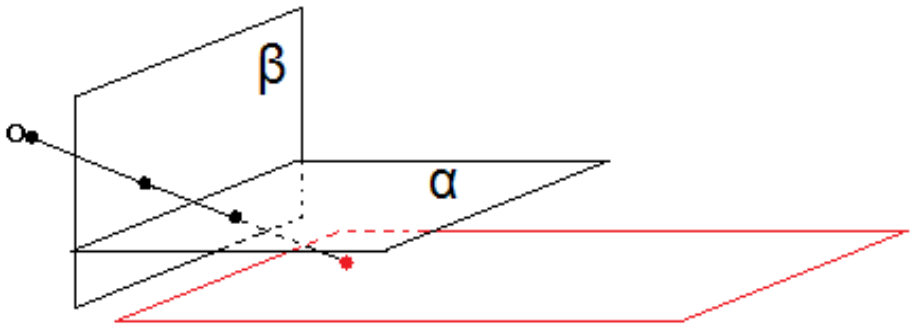

Figura 4.1. Retificação métrica, onde o plano β representa o plano de projeção original e o α, o plano de projeção almejado.

Um meio de se proceder é a partir da própria orientação relativa entre os planos global e o da câmera, o que acarretaria na disponibilidade de alguns parâmetros da câmera, não se enquadrando no modelo para o *software* proposto.

O método abordado utiliza um mapeamento entre o plano global e a imagem em perspectiva, a homografia. Essa transformação pode ser obtida pela correspondência entre pontos com posições conhecidas – ou estimadas – entre o plano de projeção e o plano global.

A homografia pode ser representada como uma matriz 3x3 no espaço projetivo, o que implica na utilização de coordenadas homogêneas para a representação dos pontos da imagem. O mapeamento, portanto, fica conforme a equação 1.

$$P' = H_{3x3} \cdot P, \tag{1}$$

$$\text{ou } \begin{bmatrix} x_i' \\ y_i' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}, \tag{2}$$

onde *P'* é o ponto no plano retificado, *H* é a matriz de homografia e *P* é o ponto no plano da imagem.

No modelo de homografia adotado – conforme LIEBOWITZ, CRIMINISI e ZISSERMAN, 1999 – o elemento das últimas coluna e linha de *H* é igual a 1 (eq. 2). Como consequência, um conjunto – não singular – de 4 correspondências $P' \leftrightarrow P$ é suficiente para se determinar *H*. No caso de um conjunto maior de pontos – sobredeterminado – pode-se utilizar mínimos quadrados para se obter a matriz de homografia – o que *ainda* não foi implementado no programa.

Uma vez obtida a homografia, basta aplicá-la na imagem para se obter a imagem com a correção de perspectiva.

4.2. EQUALIZAÇÃO DE HISTOGRAMA

O histograma de uma imagem digital é uma função discreta $h(i) = n_i$, onde *i* é o nível de intensidade de um componente de cor da imagem, e n_i, o número de pixels dessa imagem com nível de intensidade *i*. Em outras palavras, um histograma é uma distribuição de intensidades de cores em uma imagem.

O histograma sintetiza informações estatísticas muito ricas a respeito da imagem em um formato simples de se interpretar, daí sua utilidade como base para diversas técnicas de processamento de imagens. Conforme é evidenciado na figura 4.2.

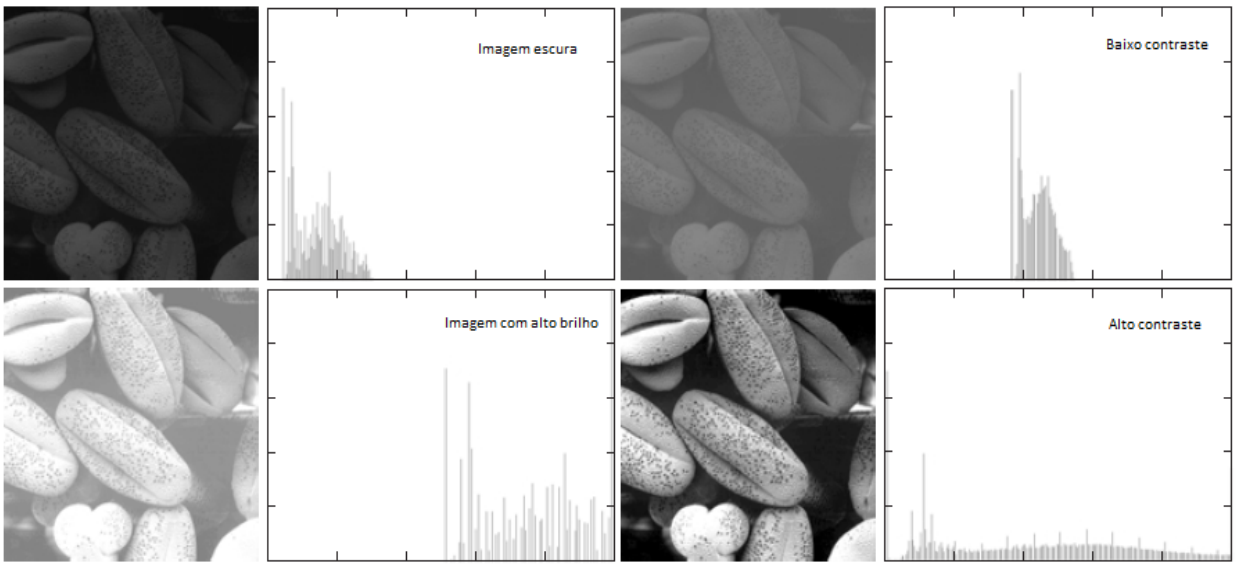

Figura 4.2. Correspondência entre brilho e contraste com o histograma da imagem.

A equalização de histograma é uma técnica de processamento de imagens voltada para normalização do brilho e aumento do contraste. O processamento permite manipular imagens de modo que áreas de baixo contraste ganhem mais contraste. A equalização de histograma alcança esse objetivo redistribuindo as intensidades de maneira que as mais frequentes sejam espalhadas (figura 4.3).

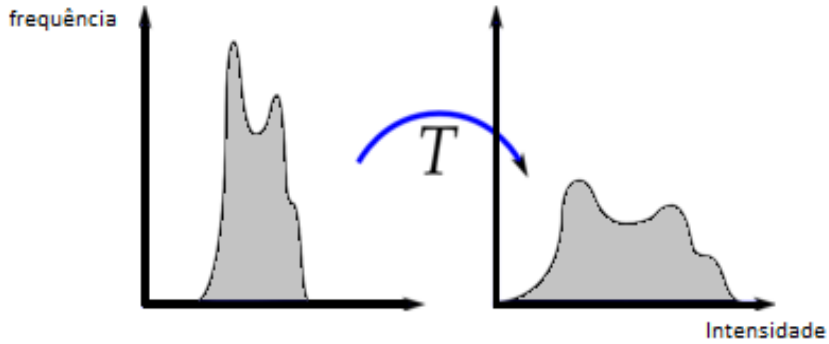

Figura 4.3. Operação realizada em histograma redistribuindo as intensidades mais frequentes.

Seguem os passos do algoritmo de equalização de histograma de uma imagem:

Normaliza-se o histograma *h* da imagem para se obter a função de distribuição *g*:

$$g(i = x) = \frac{h(x)}{n} = \frac{n_x}{n}, \tag{3}$$

onde *n* é número de pixels da imagem.

Calcula-se então a função distribuição acumulada *fdc*:

$$fdc(i) = \sum_{j=0}^{i} g(j). \tag{4}$$

A imagem *src(x, y)* = i_{src} é então mapeada para *dst(x, y)* = i_{dst}, por meio de *fdc* com um fator de escala *k* = *max(i)*, isto é, o maior valor de *i* possível – 255 para imagens de 8bits de profundidade:

$$dst(x, y) = fdc\big(src(x, y)\big) \cdot k. \tag{5}$$

A técnica é aplicada para uma componente de cor da imagem – geralmente imagens em escala de cinza. Para imagens coloridas a equalização de cada componente tem resultado imprevisível ou sem significado, por alterar o balanço de cores e consequentemente seu brilho e contraste. Como alternativa, converte-se a imagem para outro espaço de cor: Lab ou HSL/HSV, e então se aplica a equalização na componente de luminância ou valor.

4.3. INTERPOLAÇÃO

A ampliação e redução de imagem acarretam problema de como gerar e retirar amostras da imagem. Por exemplo, ao se aplicar uma simples transformação linear de escala em cada ponto para ampliar uma imagem de 10 x 10 pixels – 100 pixels no total – para 20 x 20 pixels – 400 pixels no total – surgirão 300 novos pontos não preenchidos na nova imagem. Analogamente, ao se reduzir uma imagem, pontos devem ser retirados, pois são sobrepostos na transformação.

Existem métodos para se criarem os pontos faltantes na ampliação, assim como para a escolha de como os pixels são retirados na redução, porém, em ambos os casos, as técnicas geram resultados que podem, ou não, serem adequados ao que se objetiva com elas.

Descreve-se, em seguida, algumas dessas técnicas.

4.3.1. VIZINHO MAIS PRÓXIMO

A interpolação pelo vizinho mais próximo é bem simples de ser implementada computacionalmente, e é tal qual o nome indica. Ao se fazer a ampliação de uma imagem os pontos não definidos têm os valores atribuídos iguais ao ponto transformado proveniente da imagem original mais próxima.

A vantagem dessa técnica é que ela preserva a morfologia original da imagem como ela está na imagem original, com o revés de se ter um resultado com transições pouco suaves – efeito tabuleiro de xadrez, em que a imagem fica serrilhada e composta de vários quadrados – o que pode dificultar na interpretação do que a imagem representa, especialmente para imagens com sinais de alta-frequência.

4.3.2. BILINEAR

Essa interpolação resulta em transições mais suaves na imagem ampliada, de modo que não se verifica o efeito xadrez. A técnica fica mais adequada, então, quando se deseja contornar esse efeito.

O método se baseia na do vizinho mais próximo e em uma composição de interpolações lineares – o que constitui uma interpolação bilinear; assim, finalmente, se obtém um gradiente suave entre os pontos.

Nos passos descritos do método, que se seguem, denomina-se valor de um ponto a componente de cor de interesse a que se deseja interpolar.

Sejam Q_{11}, Q_{12}, Q_{21} Q_{22}, R_1, R_2 *e* P tais como a figura 4.4. Interpolação bilinear:

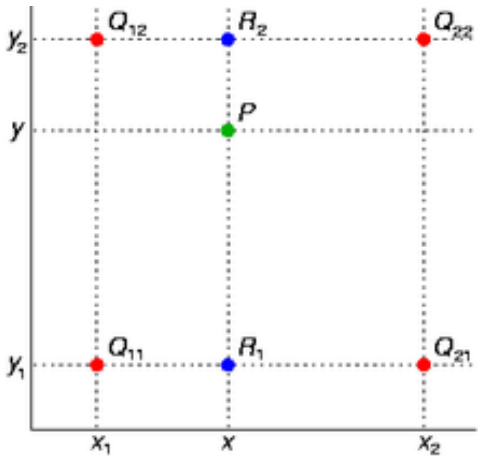

Figura 4.4. Interpolação bilinear.

Tomam-se os quatro pontos conhecidos, Q_{11}, Q_{12}, Q_{21} e Q_{22}, mais próximos numa grade 2x2 regular do ponto de interesse P – que se deseja interpolar.

Seja R_1 o ponto obtido pela interseção da reta formada pelos pontos Q_{11} e Q_{21} e a reta perpendicular a essa que contém o ponto P. Obtém-se o valor v de R_1 pela interpolação linear do valor dos pontos Q_{11} e Q_{21}:

$$v(R_1) = \frac{x_2+x}{x_2+x_1} v(Q_{11}) + \frac{x+x_1}{x_2+x_1} v(Q_{21}). \quad (6)$$

De forma análoga, obtém-se o valor v de R_2:

$$v(R_2) = \frac{x_2+x}{x_2+x_1} v(Q_{12}) + \frac{x+x_1}{x_2+x_1} v(Q_{22}). \quad (7)$$

Finalmente, interpolam-se R_1 e R_2 para se obter o valor de P.

$$v(P) = \frac{y_2+y}{y_2+y_1} v(R_1) + \frac{y+y_1}{y_2+y_1} v(R_2). \quad (8)$$

Numa forma mais compacta:

$$v(P) = \sum_{i=1,j=1}^{i=2,j=2} \frac{v(Q_{ij})}{(x_2+x_1)(y_2+y_1)} (x_2 + x)(y_2 + y). \quad (9)$$

Segundo GONZALES e WOODS, 2002, p. 64, outra abordagem para se alcançar o resultado da equação 9 é resolver a equação 10 a seguir para os pontos Q_{11}, Q_{12}, Q_{21} e Q_{22} para se obter os coeficientes *a*, *b*, *c* e *d*, para aplicá-la ao ponto *P*.

$$v(x, y) = ax + by + cxy + d. \tag{10}$$

4.3.3. BICÚBICA

A interpolação bicúbica gera uma imagem ainda mais suave que a bilinear. Em contrapartida, demanda um custo computacional maior. Um dos motivos pelo qual o resultado se mostra mais suave, é o fato de se utilizarem 16 outros pontos – vizinhança de uma grade regular 4x4 – para a estimativa do ponto interpolado.

4.3.4. INTERPOLAÇÃO LANCZOS

A interpolação Lanczos é um método utilizado para calcular os novos valores para os dados amostrados. Ele é frequentemente usado em interpolação multivariada, por exemplo, para dimensionamento de imagens. Ele é o mais lento dos filtros de interpolação disponíveis, podendo produzir imagens nítidas, mas também pode introduzir alguns ruídos.

4.4. ROTAÇÃO

Girar uma imagem consiste na aplicação de uma matriz de rotação em uma imagem. Essa matriz de rotação é uma matriz 2x2 mostrada a seguir:

$$M = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{11}$$

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix}. \tag{12}$$

Na aplicação dessa matriz de rotação a imagem resultante tem um dimensionamento maior que a imagem original, existindo pixels que não foram mapeados pela imagem original. Assim é necessário interpolação para o preenchimento desses pixels.

4.5. LUMINÂNCIA

Em um espaço de cores, como XYZ, a luminância é representada por Y. No espaço RGB, que é o comumente utilizado, o valor da luminância pode ser encontrado por uma média ponderada dos valores RGB, conforme a equação x.

$$Y = 0{,}2126R + 0{,}7152G + 0{,}0722B$$

Essa média representa a luminosidade, onde a luz verde é a intensidade mais percebida pelos seres humanos e a azul é a menor. O aumento e a diminuição desta fazem com que a imagem fique mais clara ou mais escura, respectivamente. Apesar de se assemelhar muito ao brilho da imagem e ter algumas relações, não são iguais. Duas imagens com diferentes valores de luminância podem ter valores de brilhos iguais.

CAPÍTULO 5 – RESULTADOS E ANÁLISES

5.1. INTERFACE

A aplicação foi desenvolvida de modo que novas ferramentas fossem futuramente incluídas no software, possibilitando que se preserve um modelo de interface entre todas elas.

Foi utilizado o *QtCreator* para a criação da interface. A interface foi feita com o objetivo de facilitar ao máximo a interação *software*/usuário. Seguem abaixo figuras do *software*.

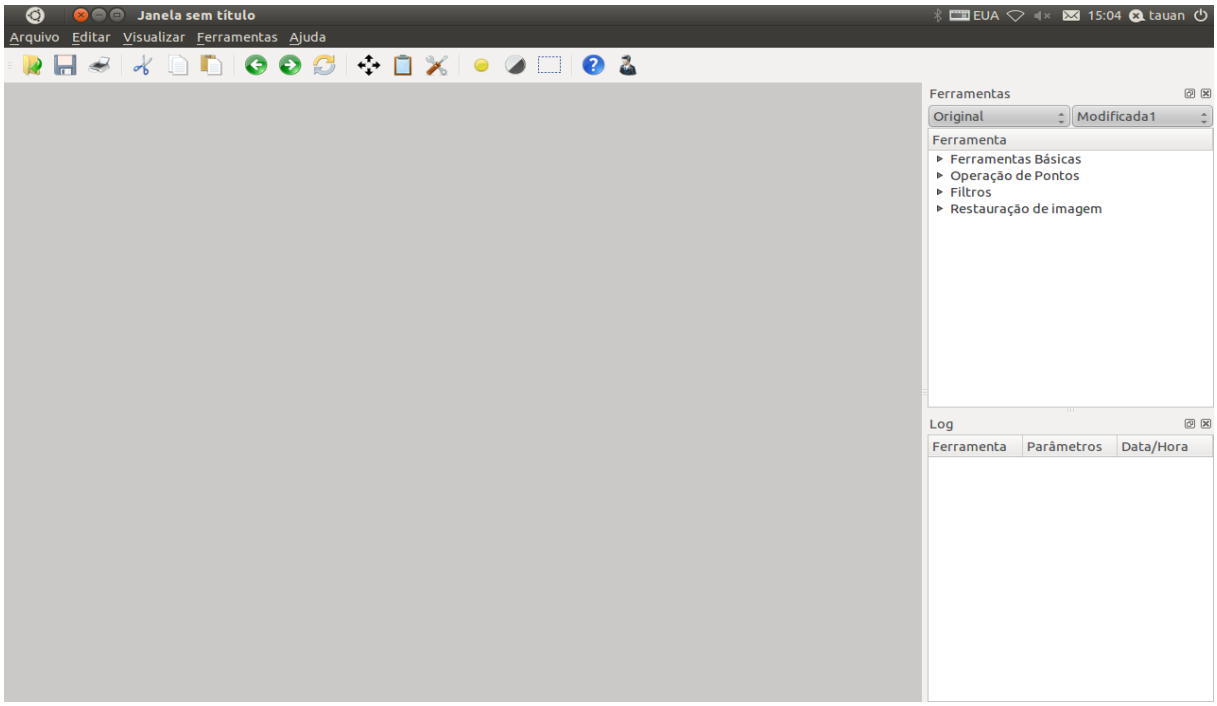

Figura 5.1. Interface do FI^2.

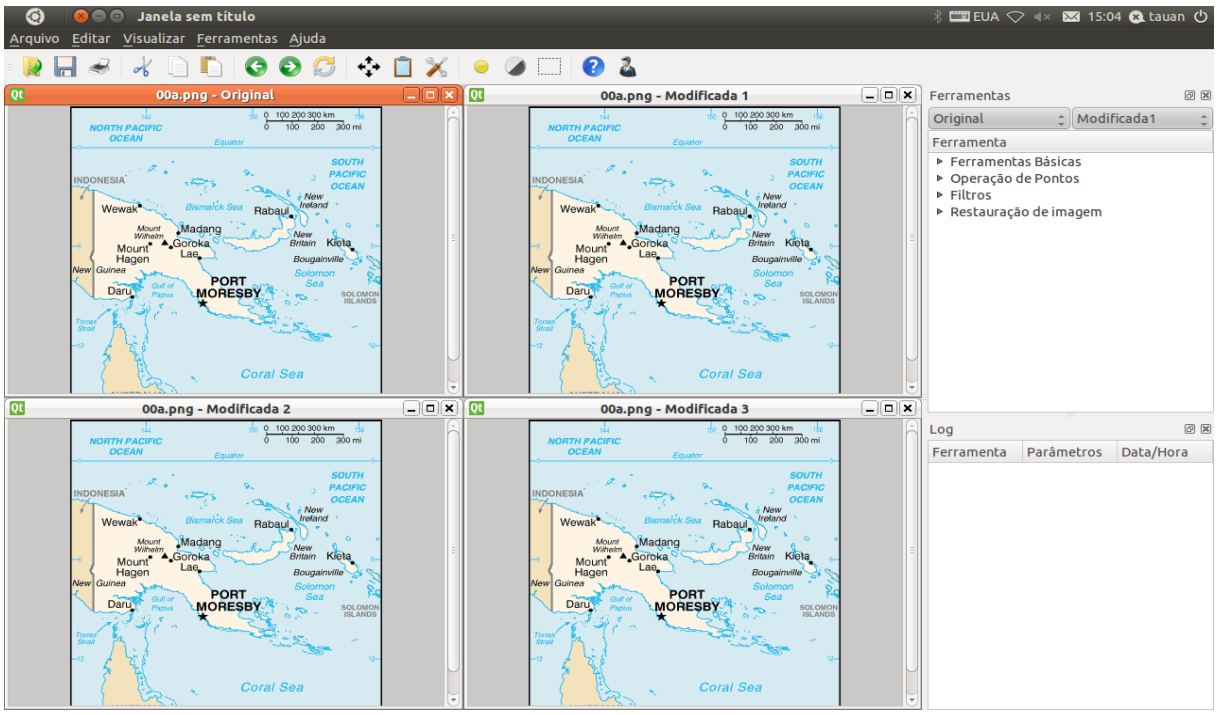

Figura 5.2. Interface do FI² com imagem.

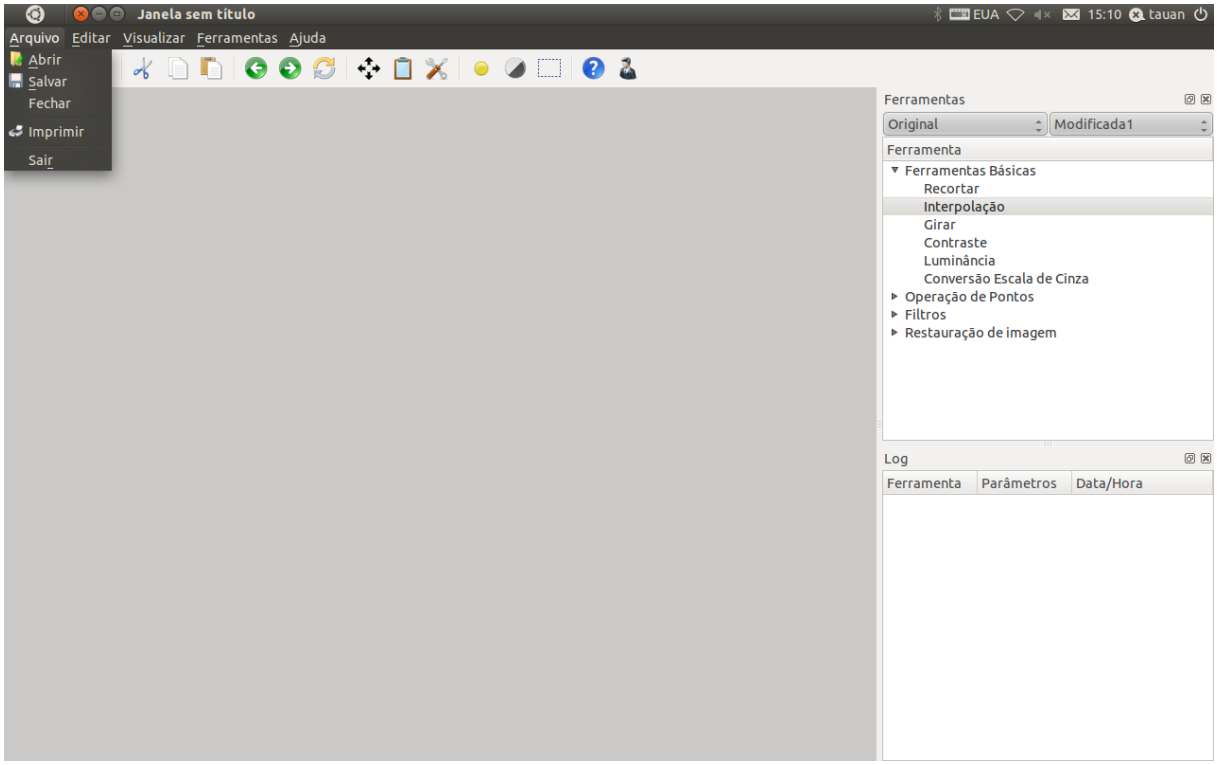

Figura 5.3. Menu Arquivo.

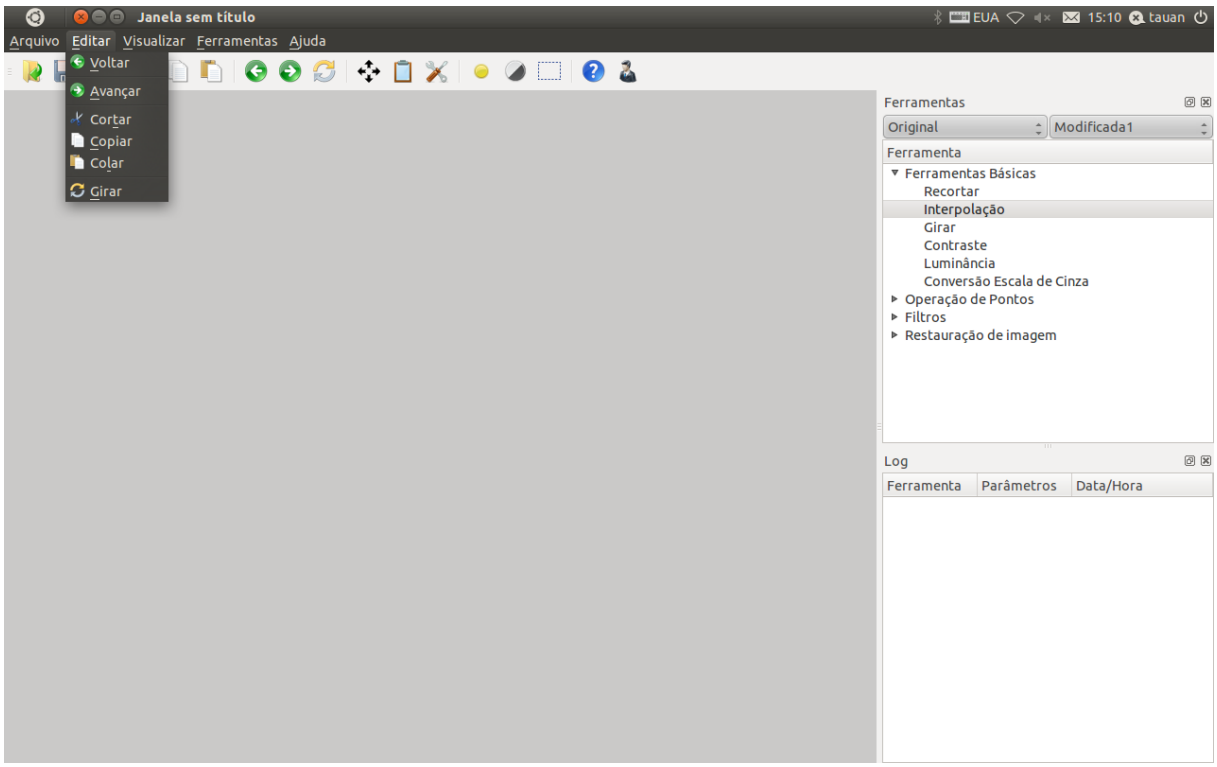

Figura 5.4. Menu Editar

Quando se utiliza o botão voltar, ele desfaz a última ação feita, ao contrário do avançar que refaz alguma ação que foi desfeita pelo voltar. Girar é a uma ferramenta que se encontra também na janela de ferramentas em Ferramentas Básicas e tem a simples função de girar a imagem.

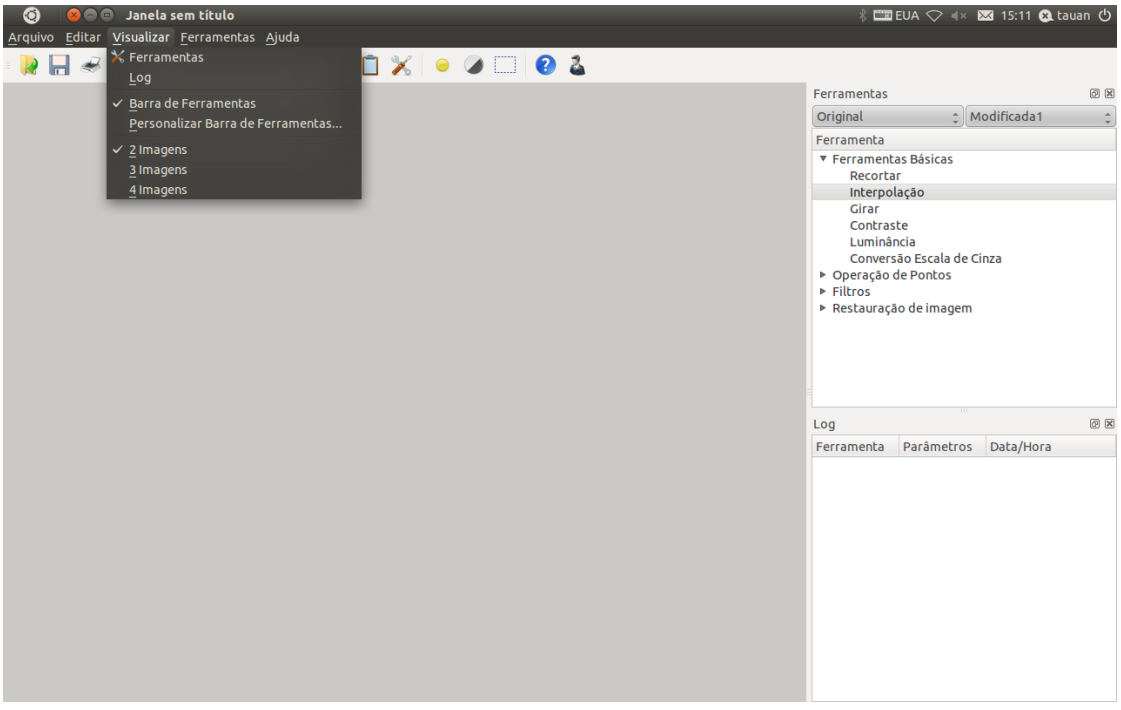

Figura 5.5. Menu Visualizar.

No Menu Visualizar, podemos ocultar ou mostrar algumas das janelas ou barras. Como pode ser visto, uma diferença para outros softwares é que existe a opção de escolher entre ter na tela 2, 3 ou 4 imagens para facilitar aplicação de várias ferramentas e comparação entre elas, com objetivo de escolher a melhor ferramenta utilizada. Como pode se ver, há uma Barra de Ferramentas abaixo do Menu, onde na opção Visualizar pode-se ocultar ela ou até personalizá-la com os itens desejados.

Figura 5.6. Barra de Ferramentas.

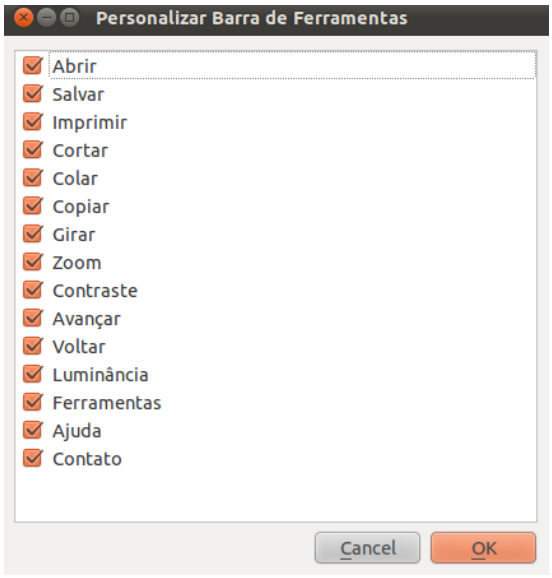

Figura 5.7. Janela para Personalizar a Barra de Ferramentas

O próximo menu é Ferramentas, ele é idêntico a janela inicial aberta à direita na parte de cima da figura 5.1.

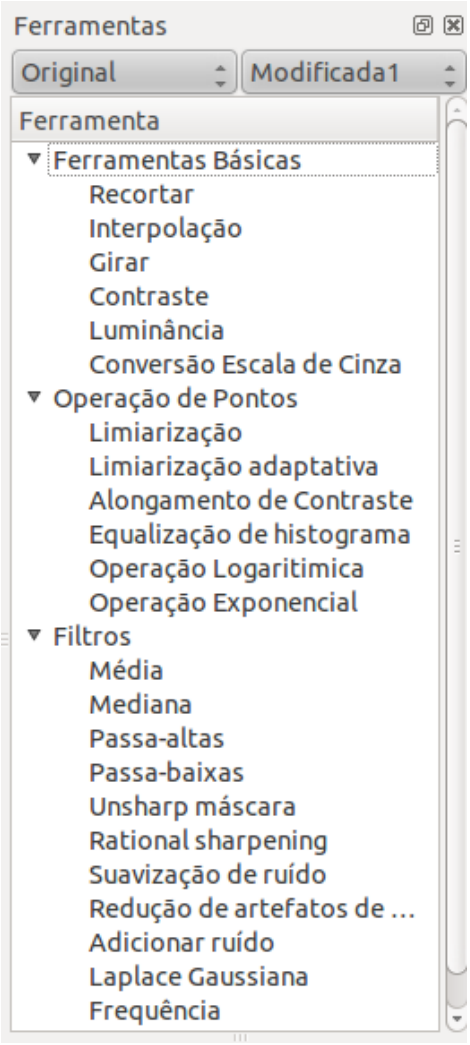

Figura 5.8. Janela de Ferramentas parte 1.

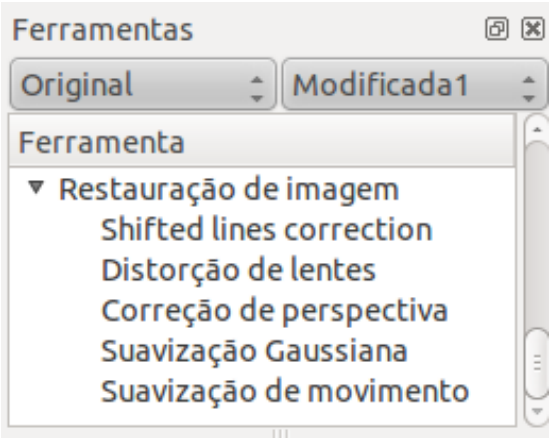

Figura 5.9 - Janela de Ferramentas parte 2.

Essa janela é a que tem todas as ferramentas aplicáveis nas imagens, para o processamento delas para a investigação. E pode ser selecionada a imagem que será modificada e a imagem de destino.

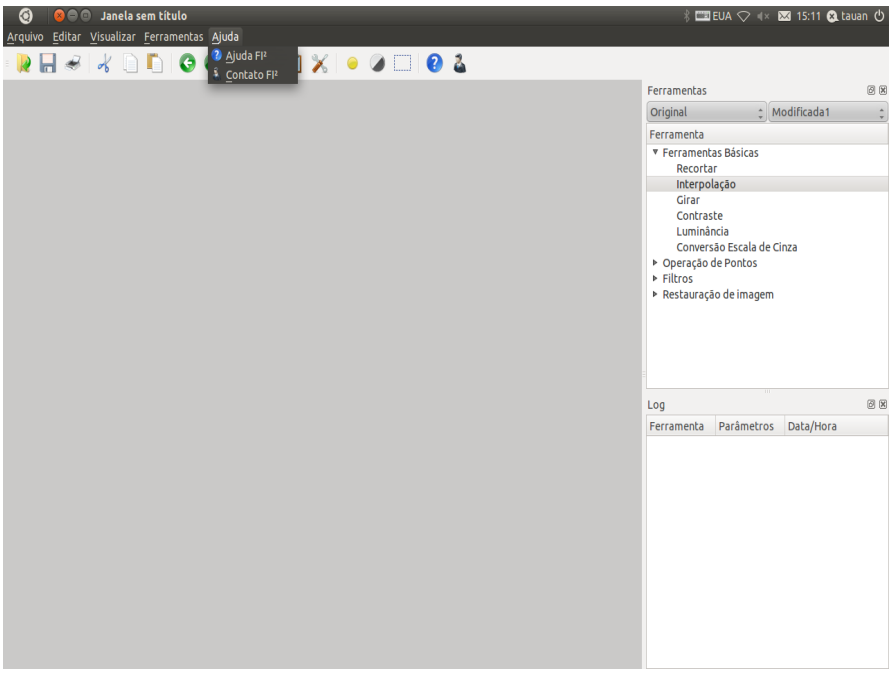

Figura 5.10. Menu Ajuda

Essa Ajuda tem com objetivo ter todos os artigos e como usar as ferramentas disponíveis para tornar o software mais utilizável.

Embaixo da Janela de Ferramentas na figura 5.1 pode se visualizar outra janela chamada *Log*. Essa janela é onde se armazena o *log* das aplicações feitas nas imagens seguido dos parâmetros e da Data e Hora feita. Esse *log* é importante para conferir a validade da imagem modificada, como também a validade das ferramentas usadas para a modificação da imagem. Cada imagem modificada tem o seu próprio *log* com as ferramentas que foram aplicadas.

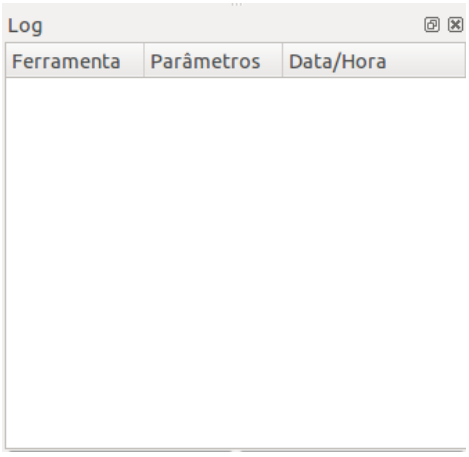

Figura 5.11. Janela do *Log.*

5.2. FERRAMENTAS

5.2.1. RECORTAR

Para o recorte da imagem foi utilizado uma área, selecionada com o mouse, assim pegando os pontos da imagem modificada e recortando imediatamente a imagem de destino. Também é aberto uma janela Recortar sobrepondo a janela Ferramentas com os 4 pontos, podendo ser modificado ponto por ponto.

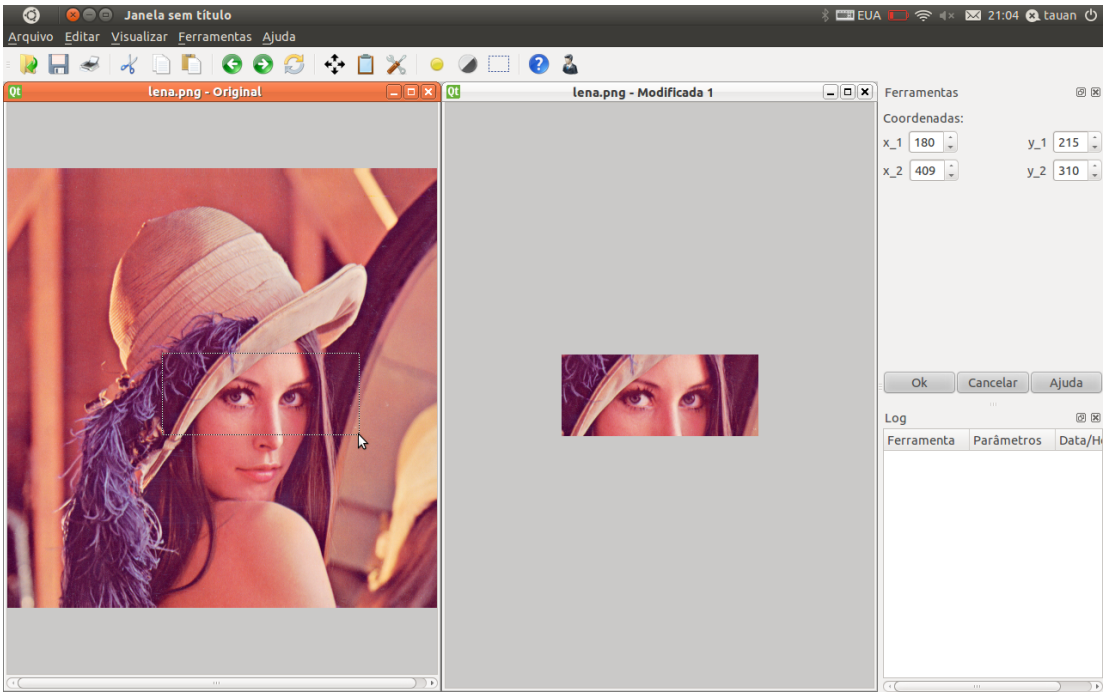

Figura 5.12. Ferramenta Recortar.

5.2.2. GIRAR

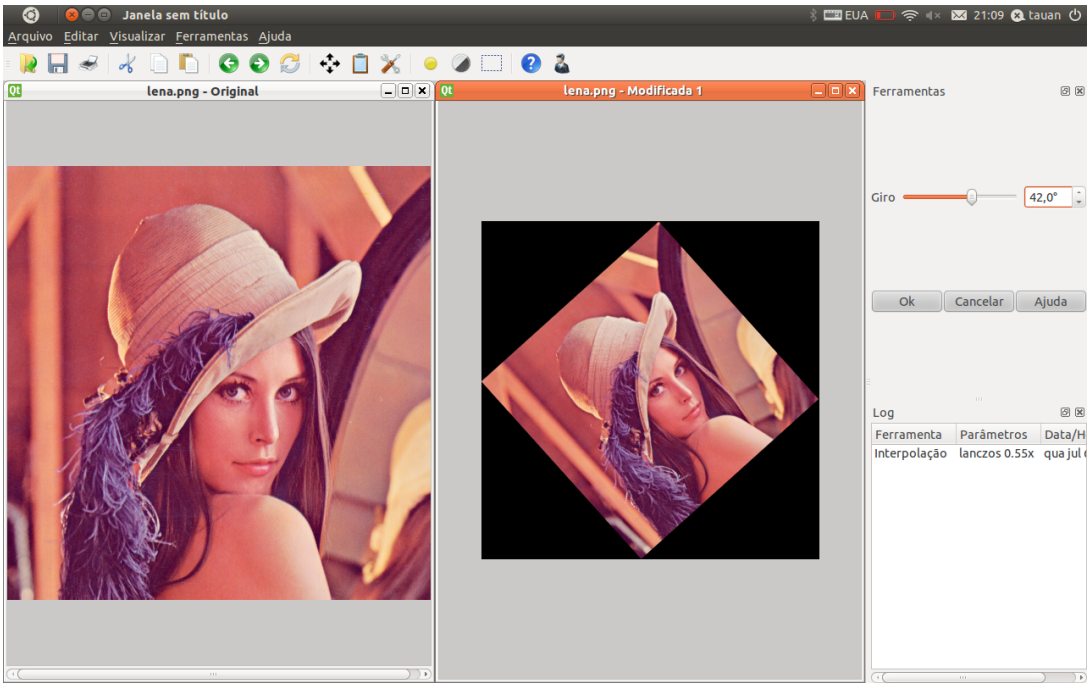

Figura 5.13. Ferramenta Girar.

Antes de aplicar a ferramenta Girar, foi aplicada uma interpolação para reduzir a imagem e assim ela ficar dentro da janela, sem barra de rolagens, facilitando a visualização do resultado. Para a própria ferramenta Girar usou a interpolação Bilinear.

5.2.3. LUMINÂNCIA

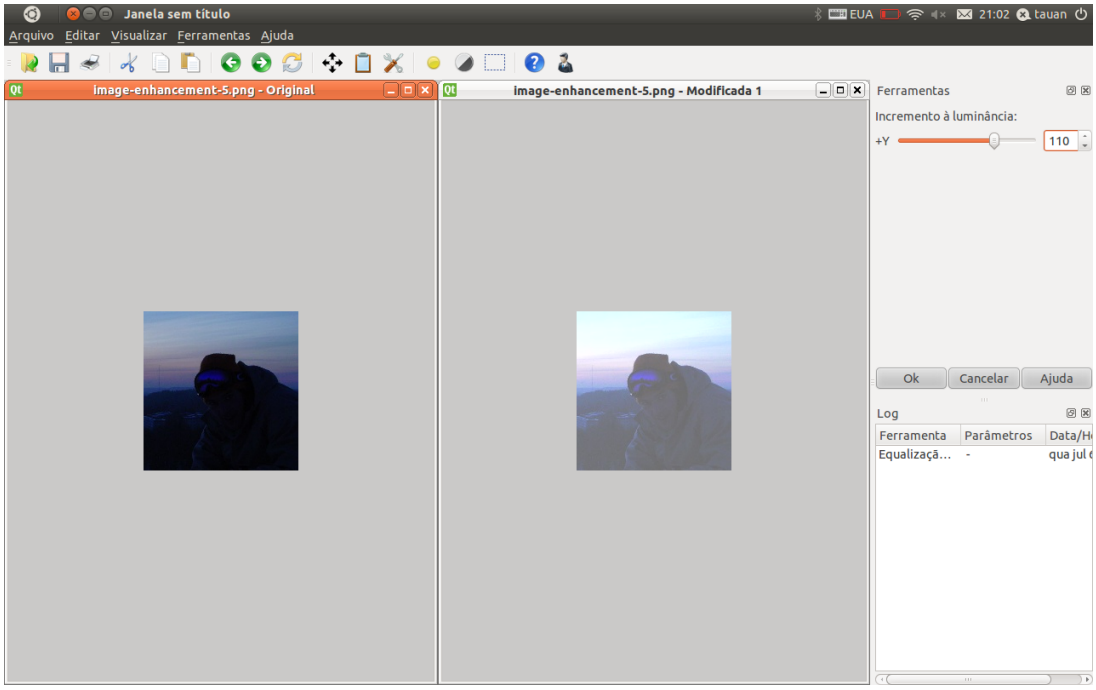

Figura 5.14. Ferramenta Luminância Claro.

Essa ferramenta transforma a imagem mais clara ou mais escura, sendo mais eficiente que brilho, pois leva em consideração o fato de que o olho humano é muito mais sensível a luz verde do que a vermelha ou a azul. Como resultado da figura 5.14 pode-se identificar melhor quem é o individuo da foto.

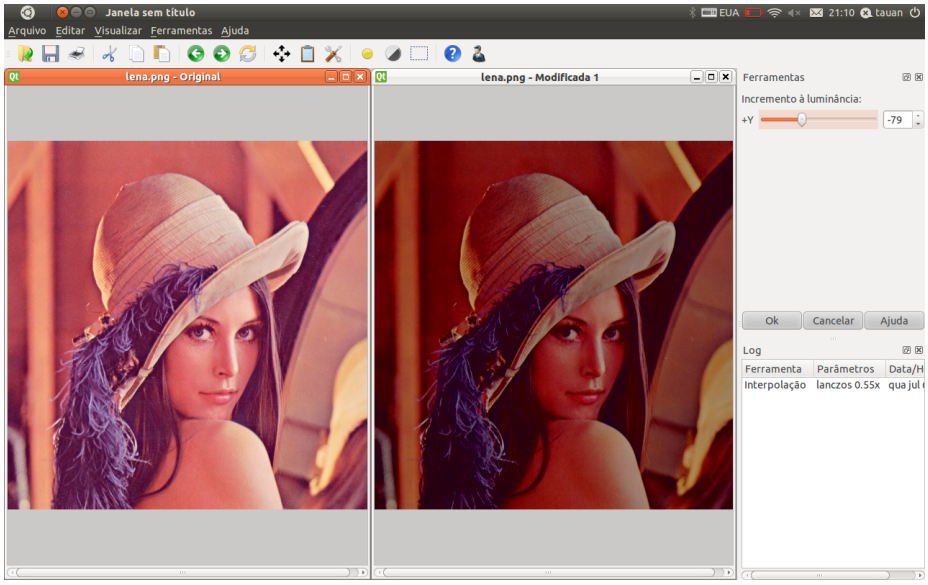

Figura 5.15. Ferramenta Luminância Escuro.

5.2.4. CONVERSÃO EM ESCALA CINZA

Ferramenta que transforma uma imagem colorida em uma imagem preto e branco. Sua funcionalidade melhora quando se têm as ferramentas de limiarização.

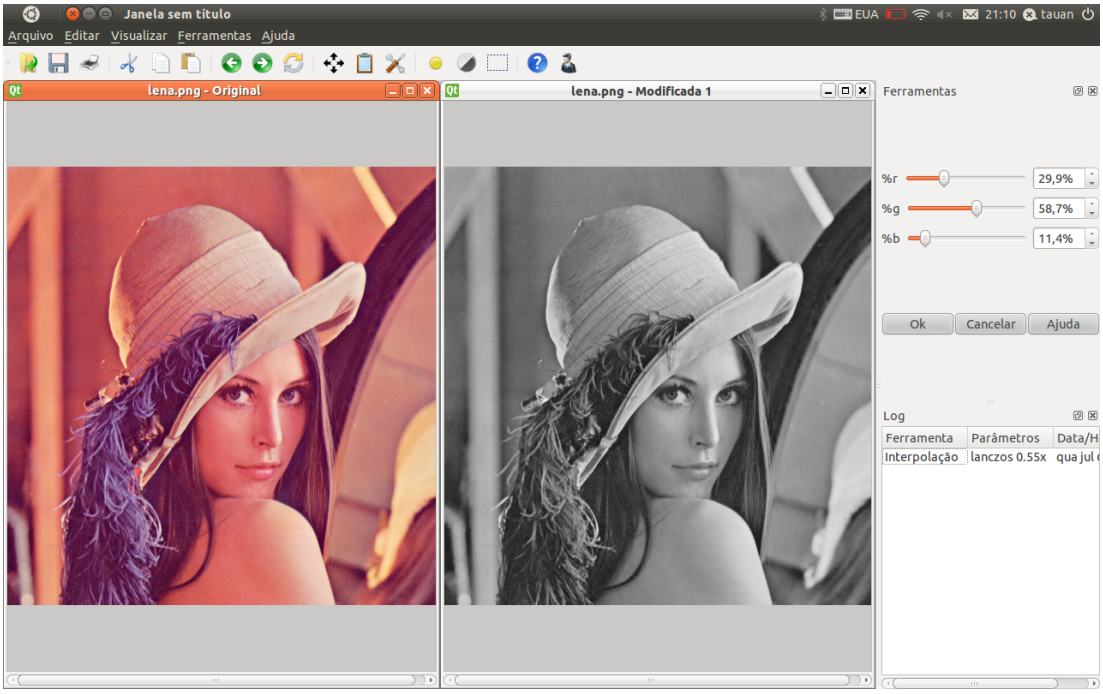

Figura 5.16. Ferramenta Conversão em Escala Cinza.

Conversão em escala cinza representa uma média ponderada das componentes RGB de uma imagem, assim na ferramenta existem 3 parâmetros para se escolher onde cada um representa o peso de cada componente nessa média.

5.2.5. CORREÇÃO DE PERSPECTIVA

Nessa ferramenta é necessário selecionar 4 pontos, onde esse 4 pontos são pontos da imagem onde na imagem destino esses pontos seriam um retângulo. Após selecionados, os 4 pontos geram a imagem de destino, existem 2 barras de rolagem na janela da ferramenta que servem para configurar a largura e a altura do retângulo, dando uma flexibilidade para o usuário ajustar as proporções do retângulo. Na figura abaixo, o resultado mostra o reconhecimento de uma placa de um carro inicialmente não identificável.

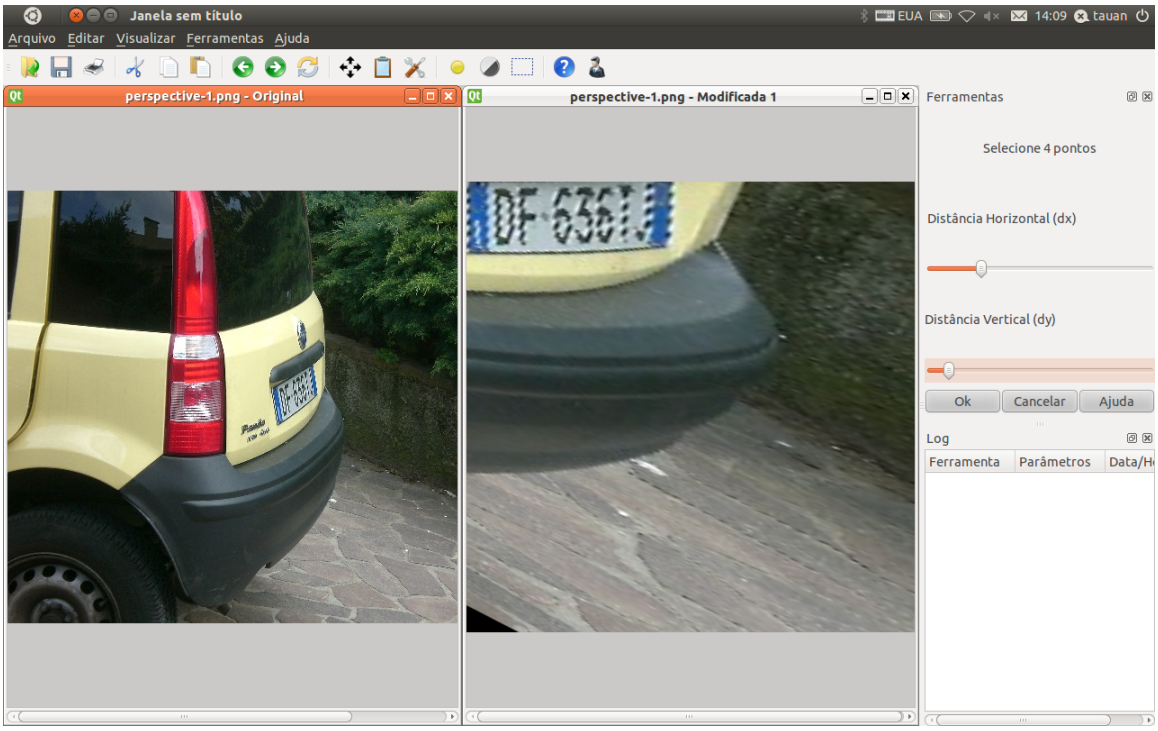

Figura 5.17. Ferramenta Correção de Perspectiva.

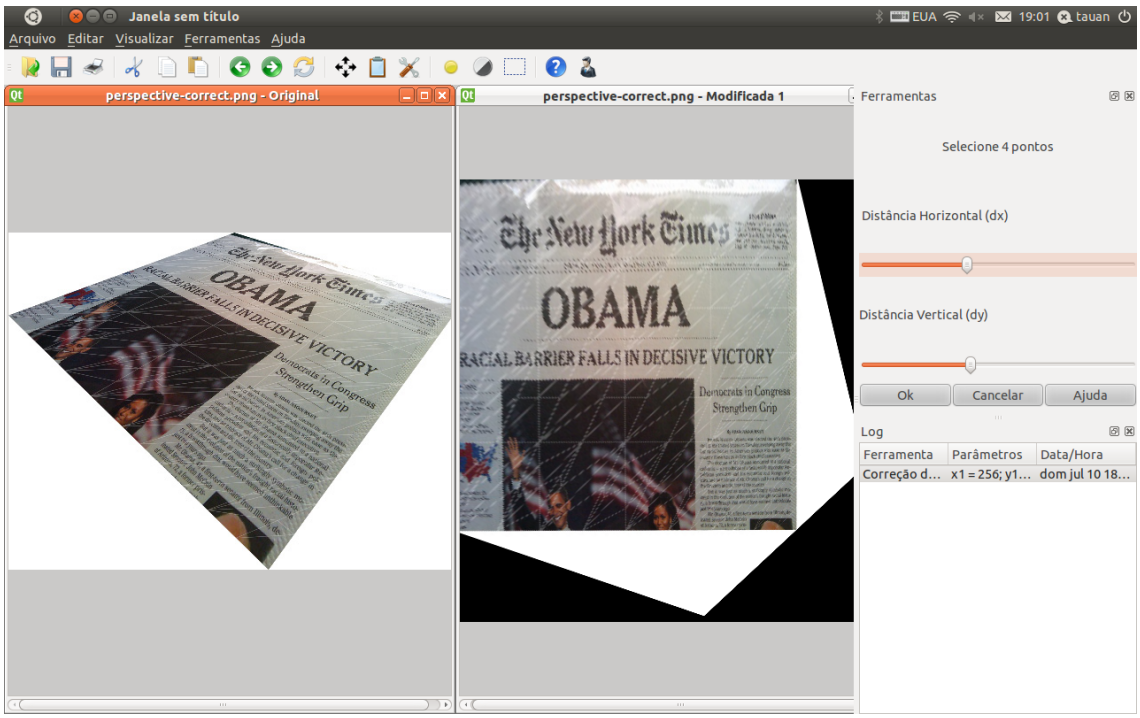

Figura 5.18. Ferramenta Correção de Perspectiva.

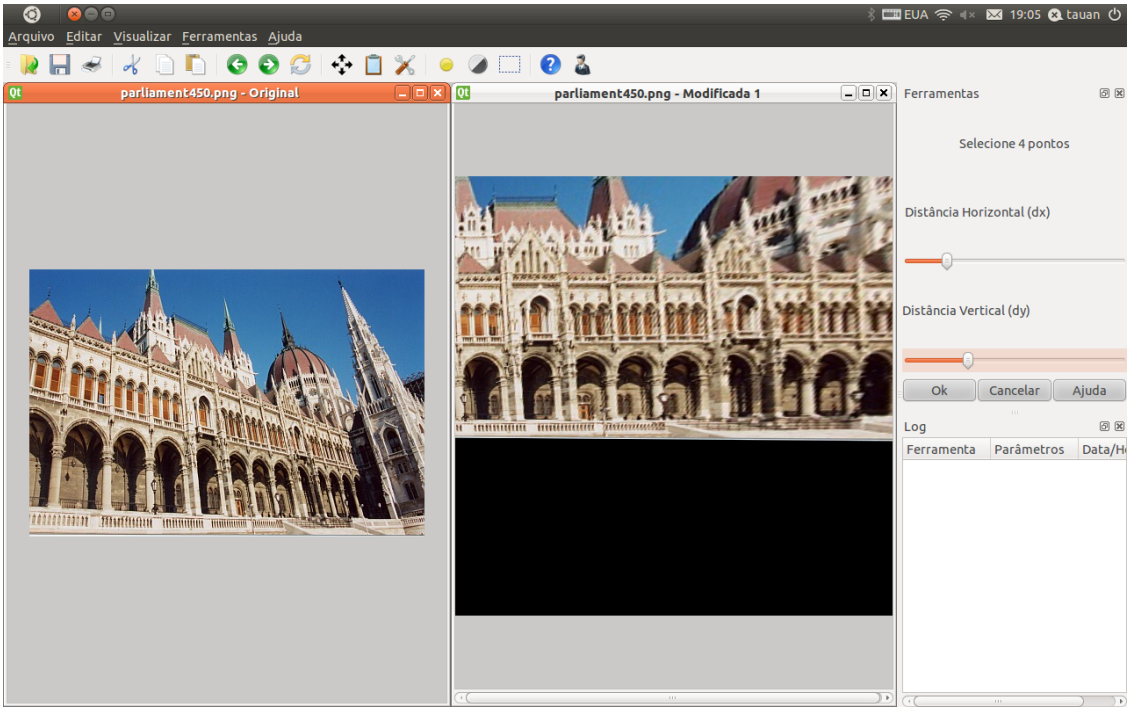

Figura 5.19. Ferramenta Correção de Perspectiva.

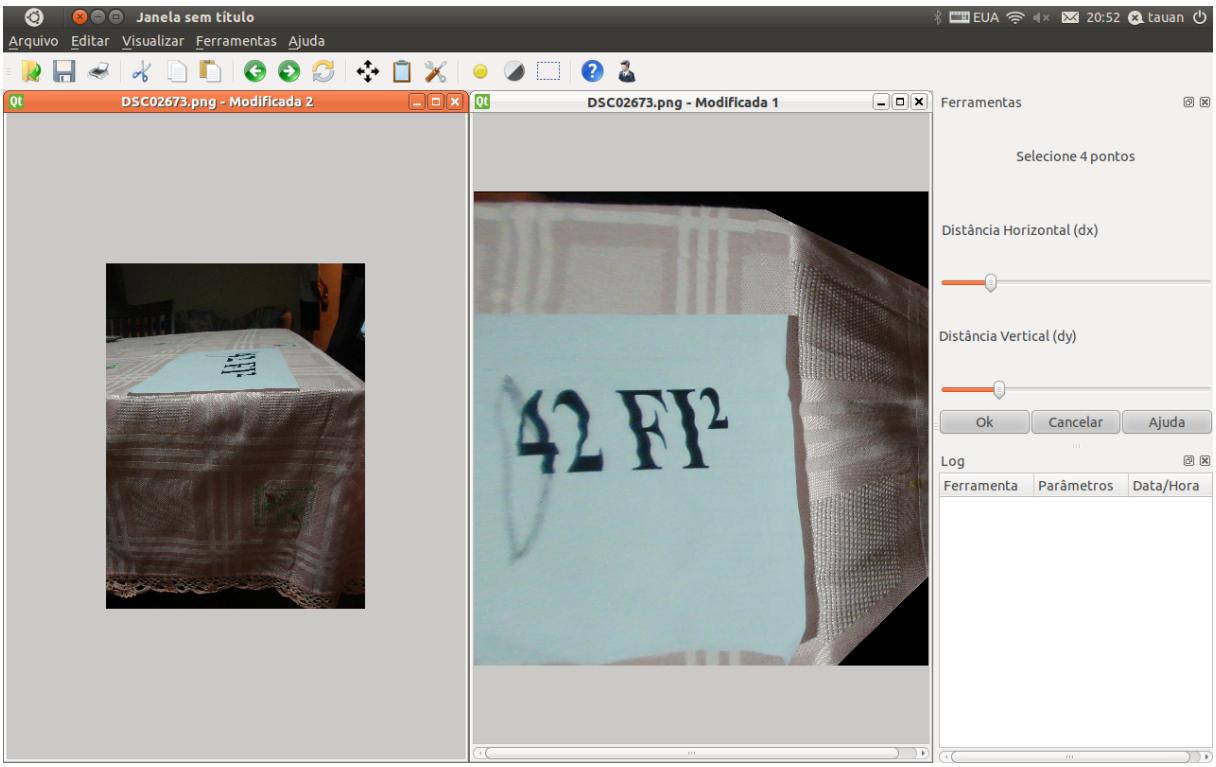

Figura 5.20. Ferramenta Correção de Perspectiva.

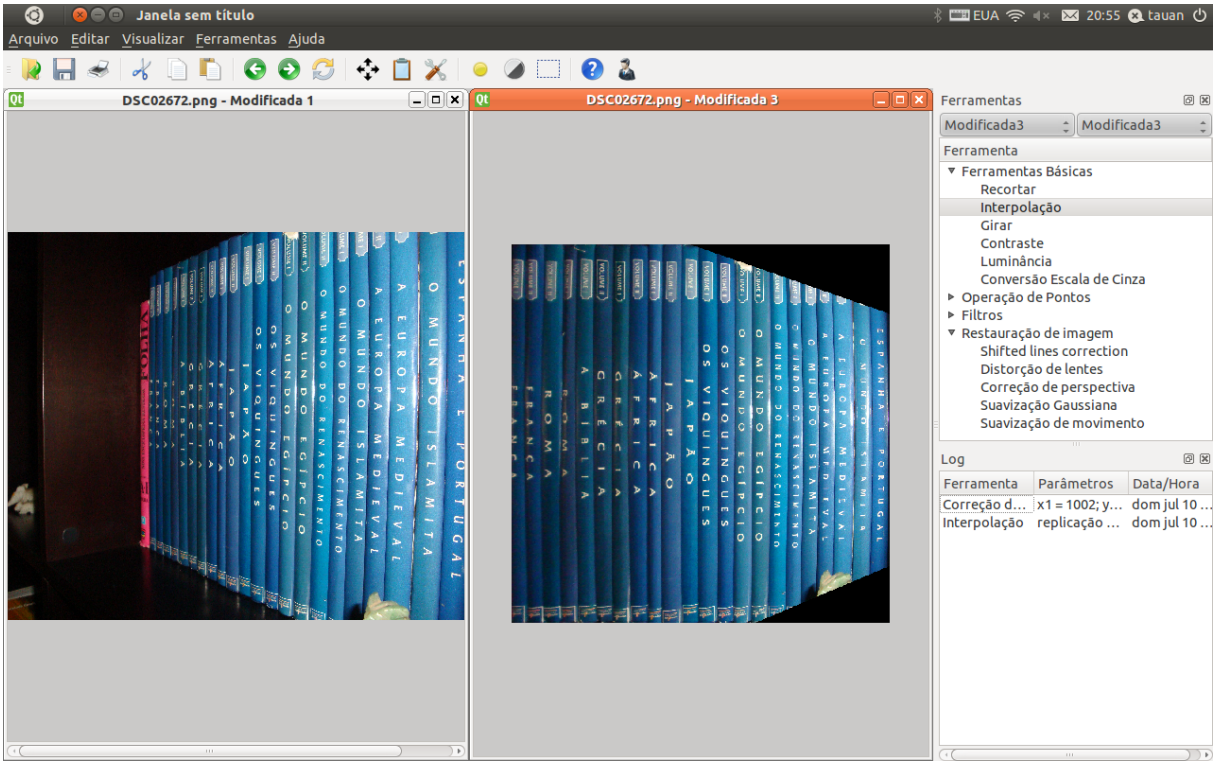

Figura 5.21. Ferramenta Correção de Perspectiva.

5.2.6. EQUALIZAÇÃO DE HISTOGRAMA

Equalizar o histograma significa obter a máxima variação do histograma de uma imagem, obtendo assim uma imagem com o melhor contraste. Assim não é necessária a

escolha de parâmetros. Como pode se notar na figura abaixo, onde é possível o reconhecimento da pessoa na figura.

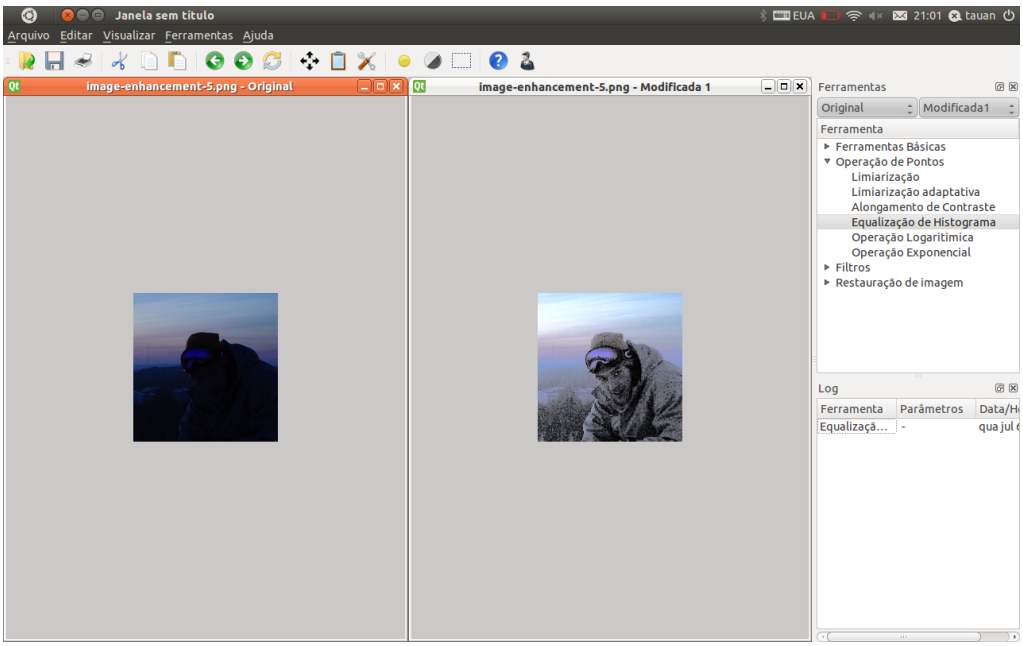

Figura 5.22. Ferramenta Equalização de Histograma.

5.2.7. INTERPOLAÇÃO

Na janela da ferramenta Interpolação há a opção de escolher qual o tipo da Interpolação desejada. Como mostrado na figura abaixo, a opção Vizinho mais próximo está selecionada e ao lado quantas vezes serão aplicadas na imagem. Vizinho mais próximo é uma interpolação que não se tem resultados bem refinados em imagens, geralmente ela é mais usada em textos.

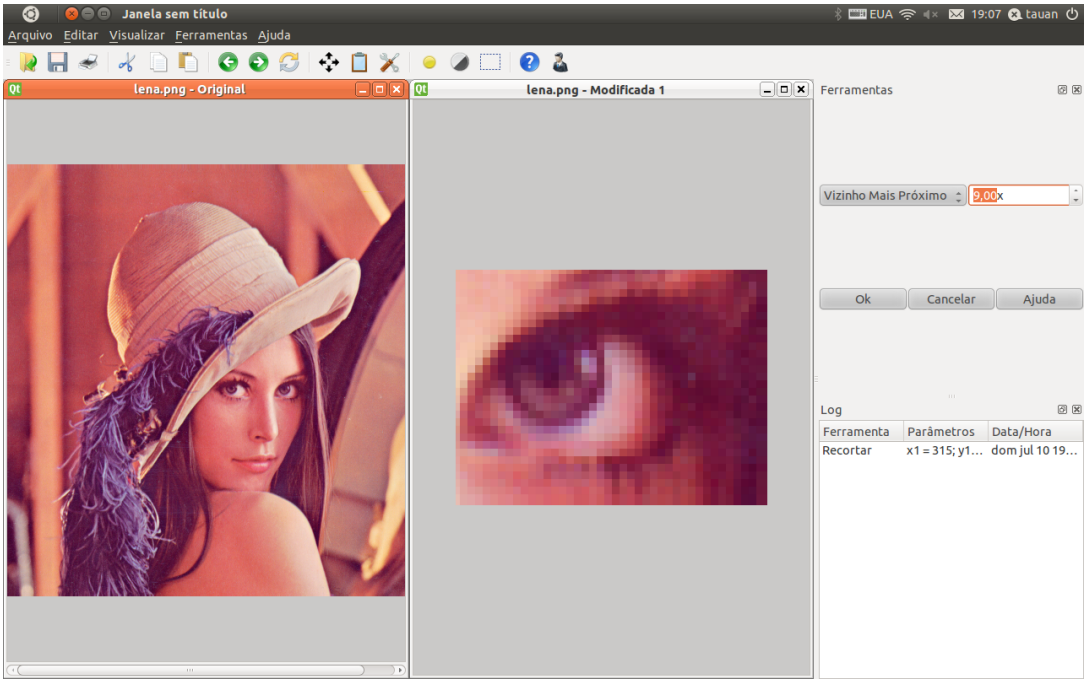

Figura 5.23. Ferramenta de Interpolação (Vizinho mais próximo).

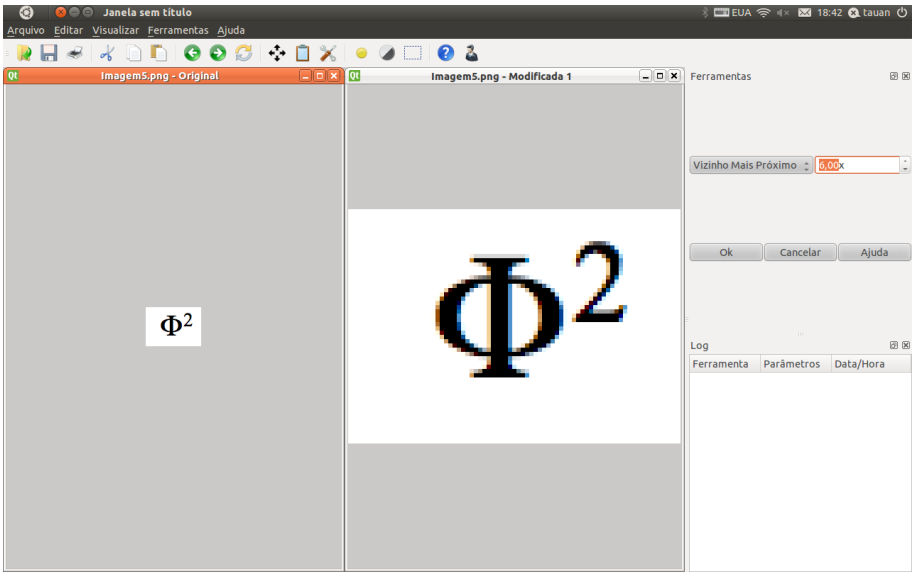

Figura 5.24. Ferramenta de Interpolação (Vizinho mais próximo).

A Interpolação Bicúbica é frequentemente usada para dimensionar as imagens e vídeos para exibição. Ela preserva detalhes melhor do que a Interpolação Bilinear.

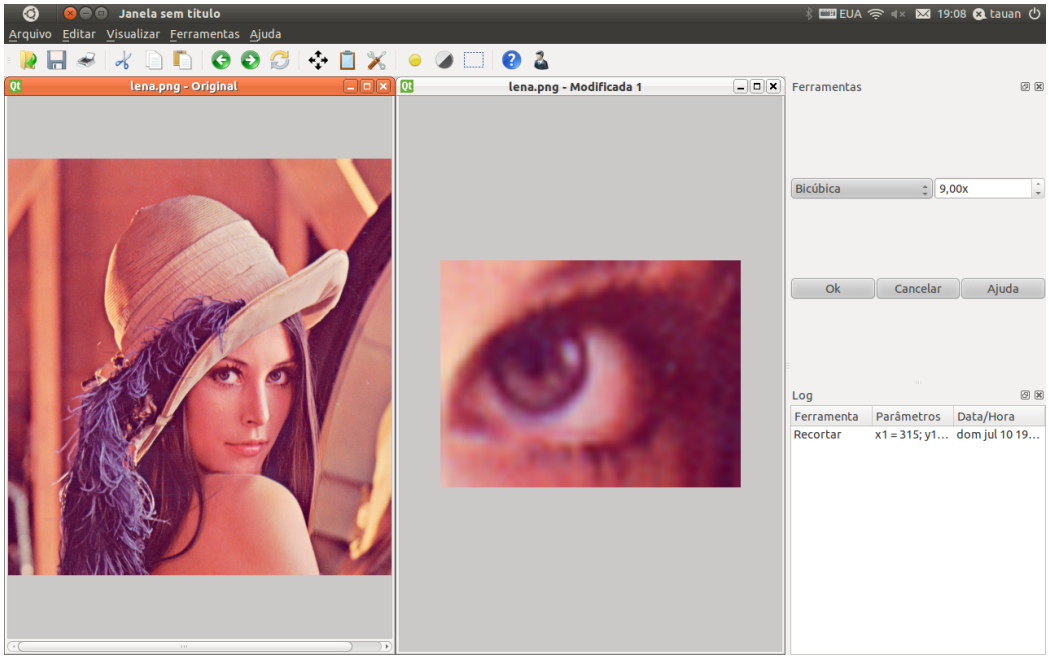

Figura 5.25. Ferramenta de Interpolação (Bicúbica).

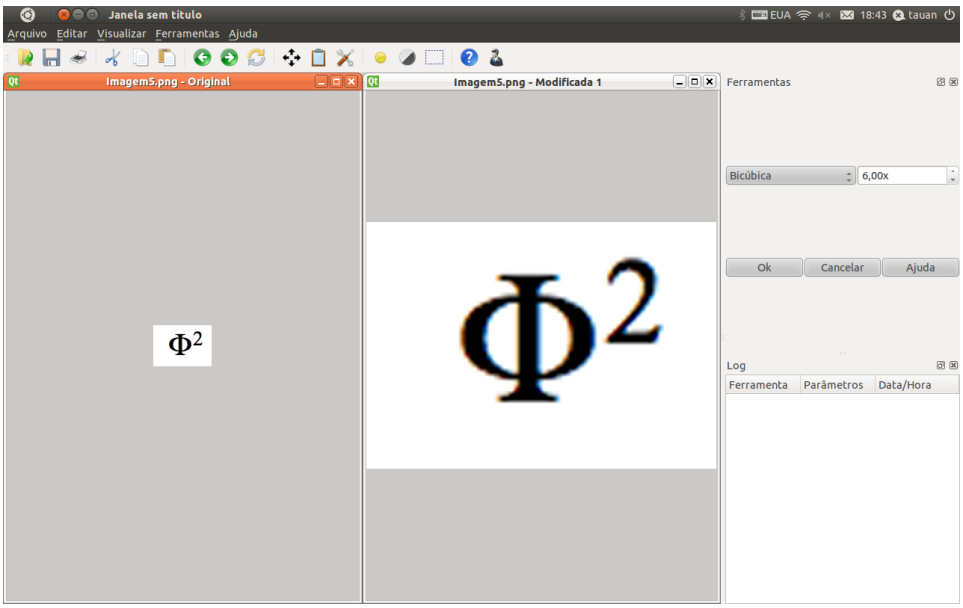

Figura 5.26. Ferramenta de Interpolação (Bicúbica).

A Interpolação Bilinear reduz algumas distorções visuais causada por um redimensionamento da imagem a um fator não integral de zoom. Interpolação Bilinear tende, no entanto, a produzir um maior número de artefatos de Interpolação (como aliasing, ofuscamento, halos e borda) do que a Interpolação Bicúbica.

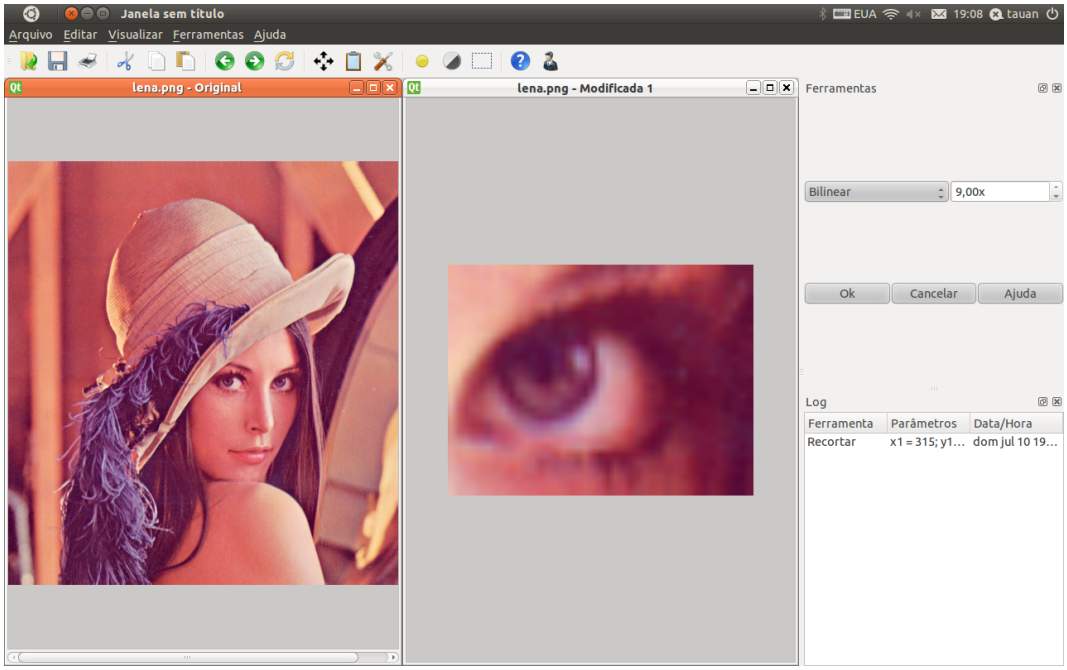

Figura 5.27. Ferramenta de Interpolação (Bilinear).

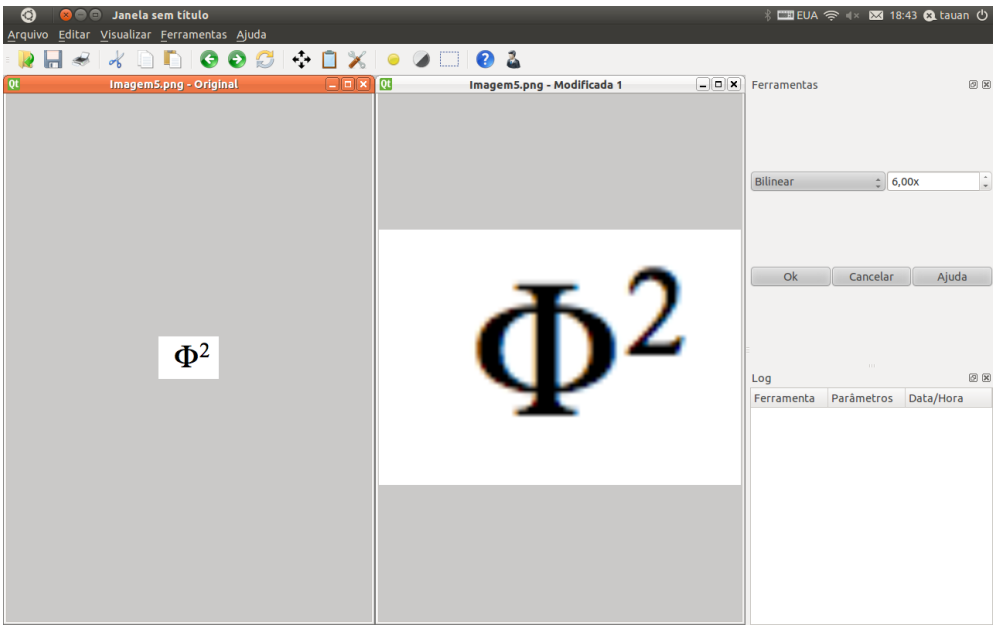

Figura 5.28. Ferramenta de Interpolação (Bilinear).

E por último a Interpolação Lanczos que obteve o resultado muito semelhante da Interpolação Bicúbica.

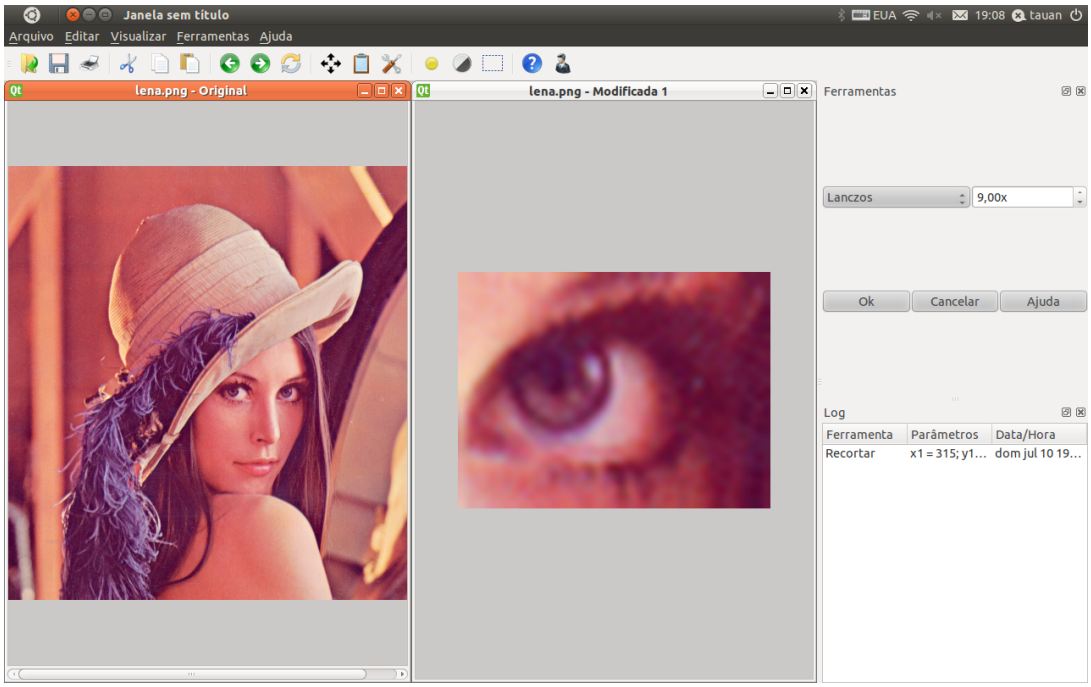

Figura 5.29. Ferramenta de Interpolação (Lanczos).

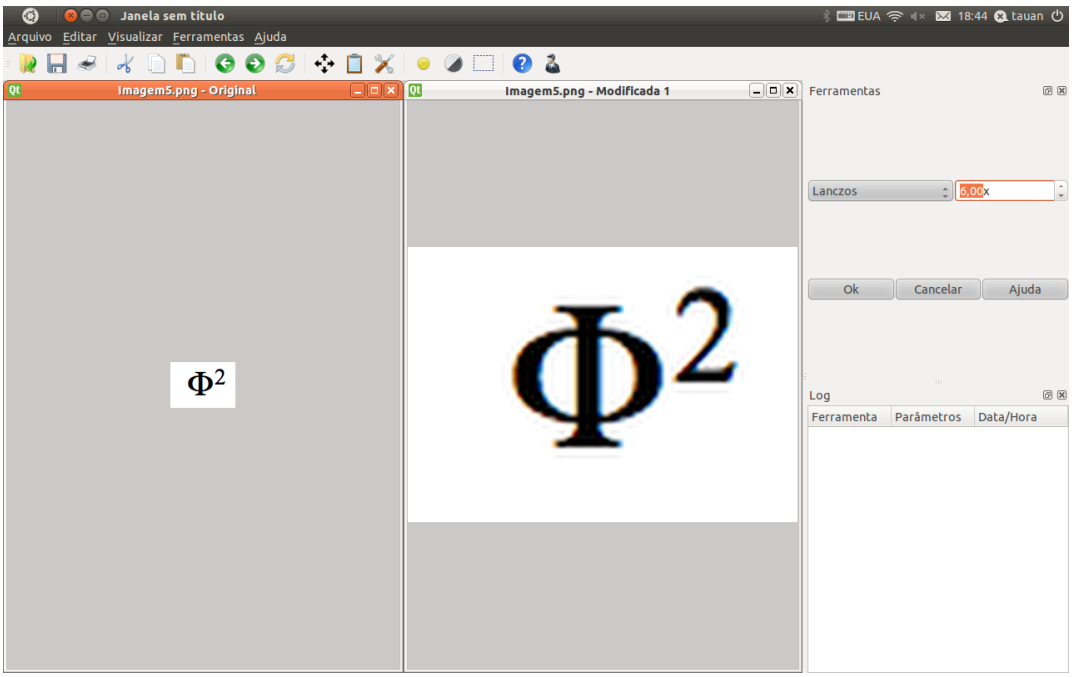

Figura 5.30. Ferramenta de Interpolação (Lanczos).

5.2.8. APLICAÇÕES PRÁTICAS

Englobando as ferramentas mostradas no capítulo 5.2 e a interface no capítulo 5.1 é possível realizar várias aplicações práticas.

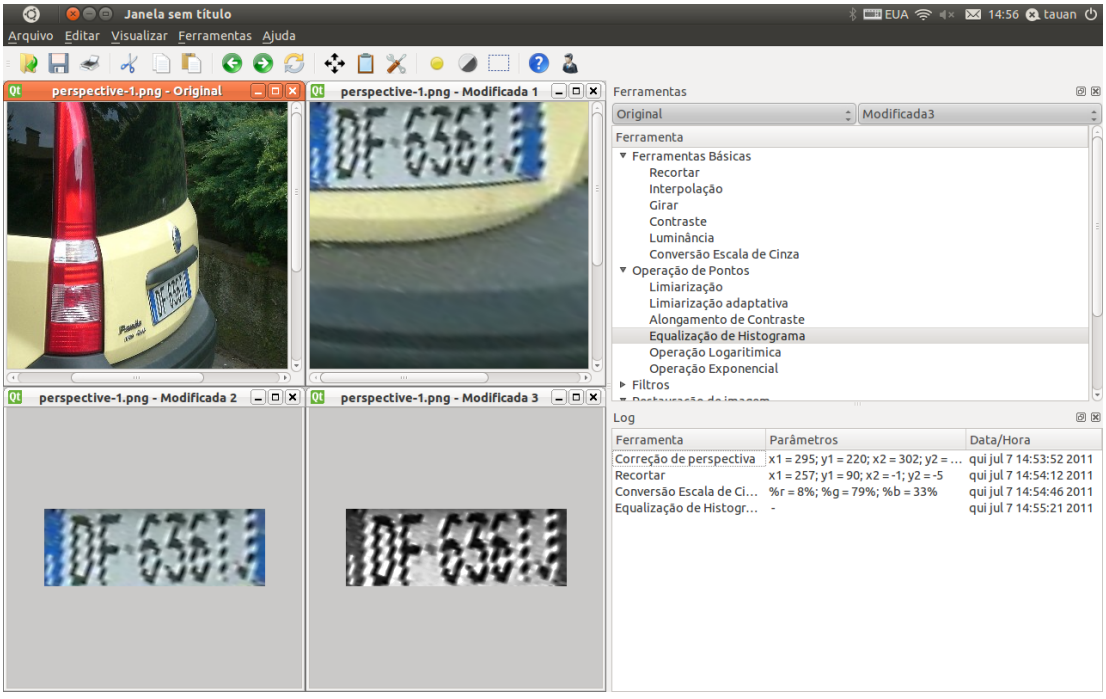

Figura 5.31. Aplicação 1.

Observado na figura acima, a utilização de várias ferramentas fez com que se obtivesse como resultado final a imagem reconhecível da placa de um carro, antes

dificilmente visível. Todas as ferramentas utilizadas foram acrescentadas no *log*, igualmente os parâmetros, data e o horário.

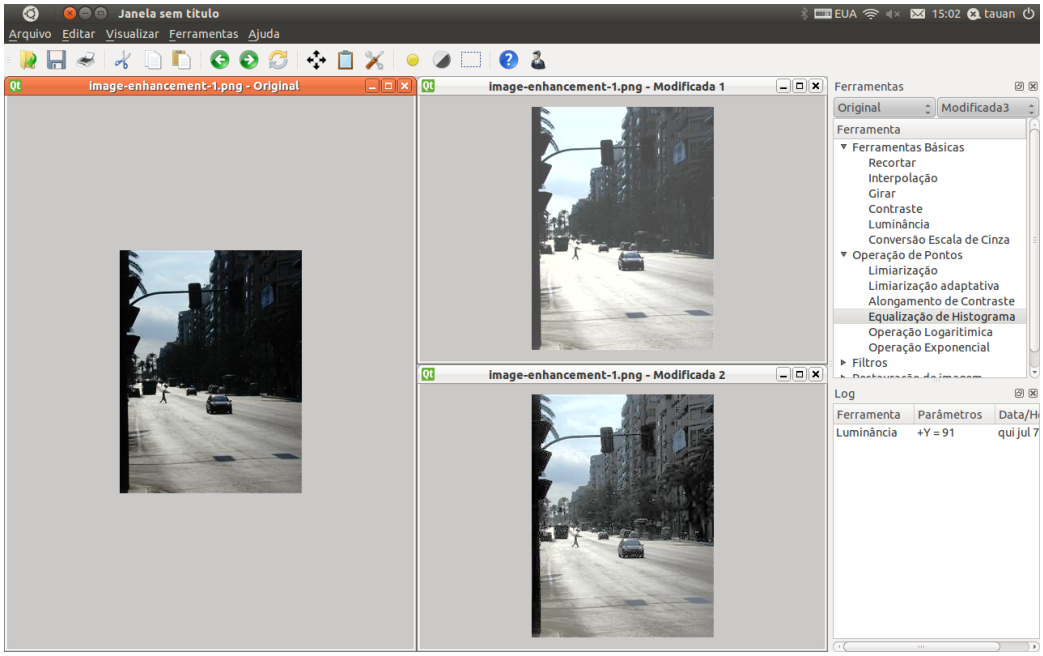

Figura 5.32. Aplicação 2.

Como mostrado na figura 5.32. Aplicação 2. , o software tem a facilidade de comparação entre duas ferramentas utilizadas na mesma imagem e o reconhecimento de qual ferramenta terá o melhor resultado. Na figura é observado que a luminância apesar de ter um bom resultado, ela deixa a desejar em alguns detalhes como os contornos.

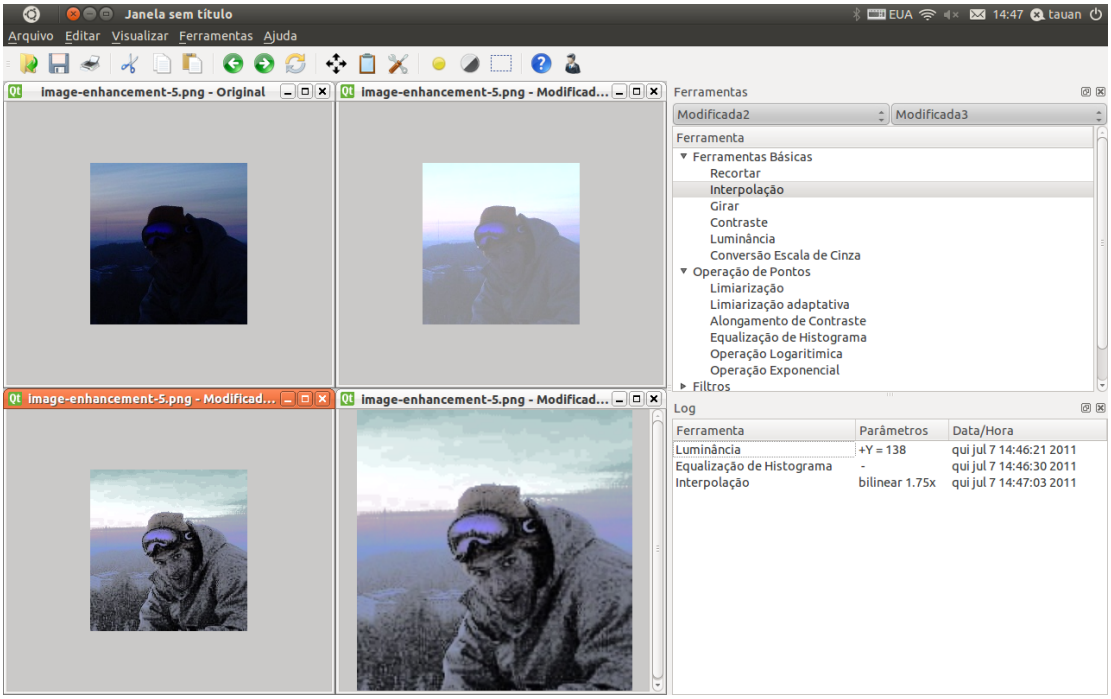

Figura 5.33. Aplicação 3.

Usando a mesma imagem das figura 5.29 e figura 5.14 e utilizando as duas ferramentas utilizadas nas figuras em sequência e em seguida uma interpolação bilinear para ampliar a imagem, teve um melhor resultado que as figura 5.29 e figura 5.14. Podendo realizar o conhecimento do indivíduo mais facilmente.

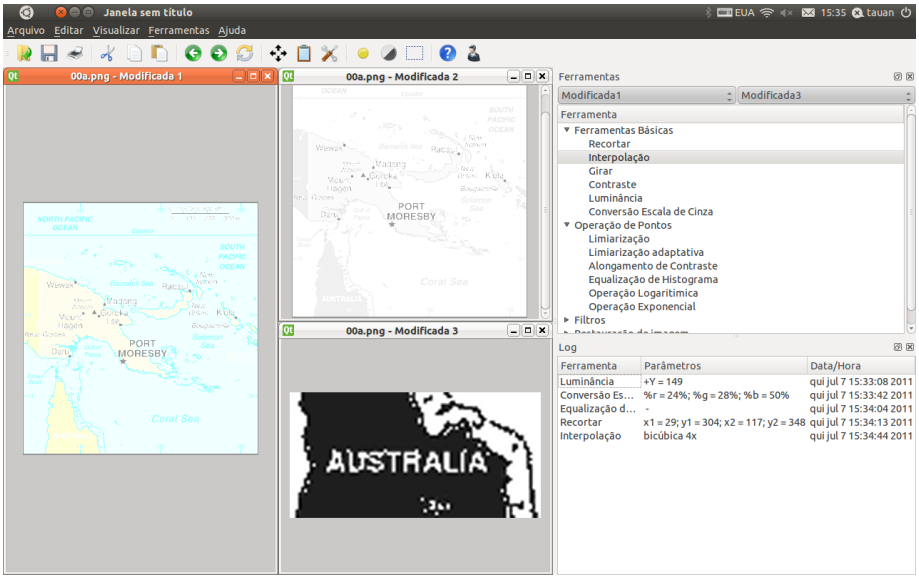

Figura 5.34. Aplicação 4.

Aplicando várias ferramentas na figura acima é possível tornar “AUSTRALIA” legível, seguindo a mesma idéia dos outros exemplos.

CONCLUSÃO

Software de processamento de imagens para investigação forense e suas ferramentas foram os temas abordados nesse trabalho, motivado, principalmente, pela ausência de um software no mercado brasileiro desse padrão.

A criação de uma interface com a tentativa de facilitar ao máximo a interação software/usuário permite fazer comparações entre várias imagens, aplicar várias ferramentas, localizar facilmente as ferramentas e o log. Apesar da falta de algumas ferramentas, foi observada a grande margem de aplicações que podem ser feitas com o software, principalmente na área de investigação forense, onde foi possível verificar algumas aplicações com resultados relevantes.

A criação desse software tem por objetivo principal a melhoria das investigações forenses, tanto em relação ao tempo quanto em qualidade, permitindo ao perito reduzir o tempo dispendido e ao aperfeiçoamento da qualidade do trabalho com imagens.

Um software que integre todas as ferramentas de análise forense de imagens com logs que permitam a reprodução fidedigna de todas as técnicas empregadas durante o exame pericial, bem como apresente uma interface prática e ágil na aplicação de suas ferramentas, possibilitará ao perito criminal realizar suas análises com a rapidez e reprodutibilidade necessárias a uma investigação técnica criminal.

Com esse trabalho foi possível desenvolver a arquitetura embrionária de um Sistema Integrado de Análise Forense de Imagens. Novos algoritmos que facilitem o trabalho pericial devem ser estudados e incorporados a esta plataforma com facilidade de uso que otimizem e documentem as análises.

BIBLIOGRAFIA

[1] D. Liebowitz, A. Criminisi, e A. Zisserman, "Creating Architectural Models from Images", apresentado no Comput. Graph. Forum, 1999, pp.39-50.

[2] R. Gonzales e R. Woods, “Digital Image Processing”, 2 edição, 2002

[3] Informação sobre o produto dTective disponível no sítio <http://www.avid.com/forensic>, acessado em 21/06/2011

[4] Informação sobre o produto Impress disponível no sítio <http://www.imix.nl/>, acessado em 21/06/2011

[5] Informação sobre o produto SignalScape disponível no sítio <http://www.signalscape.com/>, acessado em 21/06/2011

[6] Informação sobre o produto Video Analyst disponível no sítio <http://solutions.intergraph.com/hardware/vas/>, acessado em 21/06/2011

[7] Informação sobre o produto Video Investigator disponível no sítio <http://www.cognitech.com/>, acessado em 21/06/2011

[8] Informação sobre o produto Amped Five disponível no sítio <http://www.amped.it/>, acessado em 21/06/2011

[9] C. E. Duchon, "Lanczos Filtering in One and Two Dimensions". Journal of Applied Meteorology, 1979

[10] M. Jeriana, S. Paolinob, F. Cervellib,*, S. Carratoa,A. Matteib, L. Garofanob, “A forensic image processing environment for investigation of surveillance video, apresentado no <www.sciencedirect.com> , 2006

[11] Documentação de referência do OpenCV disponível no sítio <http://opencv.willowgarage.com/documentation/cpp/index.html>, acessado em 21/06/2011

[12] H.D. Cheng, M. Xue, X.J. Shi, “Contrast enhancement based on a novel homogeneity measurement”

[13] Informação sobre o produto QT disponível no sítio <http://qt.nokia.com/>, acessado em 21/06/2011

[14] R. Castagno, S. Marsi and G. Ramponi, “A Simple Algorithm for the Reduction of Blocking Artifacts in Images and its Implementation”

[15] R. Fisher, S. Perkins, A. Walker and E. Wolfart, “Image Processing Operator Worksheets”, <http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm>, acessado em 21/06/2011

ANEXO 1: TRANSFORMACAOLUMINANCIA.H

```
#ifndef TRANSFORMACAOLUMINANCIA_H
#define TRANSFORMACAOLUMINANCIA_H
#include <QString>
#include "Transformacao.h"
class TransformacaoLuminancia : public Transformacao {
public:
    virtual QString obterNome() const {
        return QString::fromUtf8("Lumin\303\242ncia");
    }
    virtual QString obterParametros() const {
        return QString("+Y = ") + QString::number(incrementoLuminancia);
    }
    virtual void aplicar(const cv::Mat &src, cv::Mat &dst) const;
    int incrementoLuminancia;
};
#endif // TRANSFORMACAOLUMINANCIA_H
```

ANEXO 2: TRANSFORMACAOLUMINANCIA.CPP

```
#include "TransformacaoLuminancia.h"
#include <vector>
void TransformacaoLuminancia::aplicar(const cv::Mat &src, cv::Mat &dst)
const {
    if (src.channels() == 3) {
        cv::cvtColor(src, dst, CV_BGR2YCrCb);
        std::vector< cv::Mat > planes;
        cv::split(dst, planes);
        cv::MatIterator_< unsigned char > it = planes[0].begin< unsigned
char >();
        cv::MatIterator_< unsigned char > it_end = planes[0].end< unsigned
char >();
        while (it != it_end) {
            *it = cv::saturate_cast< unsigned char >(*it +
incrementoLuminancia);
            ++it;
        }
        cv::merge(planes, dst);
        cv::cvtColor(dst, dst, CV_YCrCb2BGR);
    } else if (src.channels() == 1) {
        dst = src.clone();
        cv::MatIterator_< unsigned char > it = dst.begin< unsigned char
>();
        cv::MatIterator_< unsigned char > it_end = dst.end< unsigned char
>();
        while (it != it_end) {
            *it = cv::saturate_cast< unsigned char >(*it +
incrementoLuminancia);
            ++it;
        }
    }
}
```

ANEXO 3: FERRAMENTALUMINANCIA.H

```
#ifndef FERRAMENTALUMINANCIA_H
#define FERRAMENTALUMINANCIA_H
#include <QWidget>
#include "Ferramenta.h"
#include "TransformacaoLuminancia.h"
namespace Ui {
    class FerramentaLuminancia;
}
class FerramentaLuminancia : public Ferramenta {
    Q_OBJECT
public:
    FerramentaLuminancia();
    ~FerramentaLuminancia();
    virtual QString obterNome() const {
        return transformacao.obterNome();
    }
    virtual void iniciar();
private slots:
    void on_horizontalSlider_valueChanged(int value);
    void on_spinBox_valueChanged(int value);
    void on_ok_clicked();
    void on_cancel_clicked();
private:
    Ui::FerramentaLuminancia *ui;
    TransformacaoLuminancia transformacao;
    cv::Mat dstTemp;
};
#endif // FERRAMENTALUMINANCIA_H
```

ANEXO 4: FERRAMENTALUMINANCIA.CPP

```
#include "FerramentaLuminancia.h"
#include "ui_FerramentaLuminancia.h"
#include <QSharedPointer>
#include "Transformacao.h"
#include "qt_util.h"
FerramentaLuminancia::FerramentaLuminancia()
    : ui(new Ui::FerramentaLuminancia) {
    ui->setupUi(this);
}
FerramentaLuminancia::~FerramentaLuminancia() {
    delete ui;
}
void FerramentaLuminancia::iniciar() {
    dstTemp = subWindowSrc->obterImg().clone();
    qt_util::setValueSemSinais(ui->spinBox, 0);
    qt_util::setValueSemSinais(ui->horizontalSlider, 0);
    transformacao.incrementoLuminancia = 0;
    subWindowDst->exibirImagem(dstTemp);
}
void FerramentaLuminancia::on_horizontalSlider_valueChanged(int value) {
    ui->spinBox->setValue(value);
}
void FerramentaLuminancia::on_spinBox_valueChanged(int value) {
    ui->horizontalSlider->setValue(value);
    transformacao.incrementoLuminancia = value;
    transformacao.aplicar(subWindowSrc->obterImg(), dstTemp);
    subWindowDst->exibirImagem(dstTemp);
}
void FerramentaLuminancia::on_ok_clicked() {
    subWindowDst->definirImg(dstTemp);
    subWindowDst->transformacoes = subWindowSrc->transformacoes;
    subWindowDst->inserirTransformacao(QSharedPointer< Transformacao >(new
TransformacaoLuminancia(transformacao)));
    emit onEsconder();
}
void FerramentaLuminancia::on_cancel_clicked() {
    cancelar();
    emit onEsconder();
}
```