UNIVERSIDADE DO ESTADO DE MATO GROSSO
CAMPUS UNIVERSITÁRIO DO VALE DO TELES PIRES
DEPARTAMENTO DE LICENCIATURA EM COMPUTAÇÃO

Thiago Cesar Serrilho

# IDENTIFICAÇÃO DE CÓDIGO DE BARRAS EM DOCUMENTOS DE TRAMITAÇÃO INTERNA DA UNEMAT COM UTILIZAÇÃO DE *WEBCAM* PARA COLETA DE DADOS

COLIDER

2009

THIAGO CESAR SERRILHO

# IDENTIFICAÇÃO DE CÓDIGO DE BARRAS EM DOCUMENTOS DE TRAMITAÇÃO INTERNA DA UNEMAT COM UTILIZAÇÃO DE *WEBCAM* PARA COLETA DE DADOS

Monografia apresentada como exigência parcial para a conclusão da disciplina “Projeto de Licenciatura II” do Curso de Licenciatura em Computação, da Universidade de Mato Grosso - UNEMAT, Campus Universitário do Vale do Teles Pires, sob a Coordenação da Professora Mestre Egeslaine de Nez.

Orientador: Tales Nereu Bogoni

Colider

2009

Dedico este trabalho a minha namorada Sheyla Rodrigues da Silva, pois ao longo dos quase 3 anos em que estamos “juntos” sempre foi uma pessoa super carinhosa, amável e compreensiva, me apoiando e incentivando em tudo o que eu faço. Estamos distantes fisicamente, mas em pensamento estamos sempre juntos.

Agradeço primeiramente Deus por mais esta conquista em minha vida.

Aos meus pais, Carlos Cesar Serrilho e Elisabeth Mardegan Serrilho pelo carinho e dedicação, sempre me orientando e guiando pelos melhores caminhos.

Ao meu irmão Fernando pelo companheirismo.

A minha namorada Sheyla pelos momentos felizes que me proporciona, pela paciência, compreensão e apoio no desenvolvimento do trabalho.

Ao meu orientador Tales Nereu Bogoni, que esteve sempre pronto para tirar as duvidas que foram surgindo ao longo do desenvolvimento deste trabalho.

A professora Egeslaine de Nez pela ajuda referente às normas de formatações do trabalho.

A Karyne, pela ajuda oferecida na tradução do resumo.

A todos os autores que serviram de fonte de pesquisas e a todas as universidades que publicam os trabalhos de seus acadêmicos na internet, pois sem a colaboração destes, este trabalho não seria concluído.

A UNEMAT, aos professores e coordenação do Campus Universitário do Vale do Teles Pires pela oportunidade de realização de curso de alto nível de forma gratuita.

A todos que de certa forma contribuíram com o desenvolvimento deste trabalho.

Sinceros agradecimentos
Muito obrigado a todos.

Muitas das grandes realizações do mundo foram feitas por homens cansados e desanimados que, ainda assim, continuaram lutando.

**(Bernard Miller)**

## RESUMO

Atualmente existem inúmeros sistemas desenvolvidos com o intuito de fazer reconhecimento de padrões, seja ele em imagens ou em qualquer outro meio através do qual se possa extrair informações suficientes. Este trabalho descreve os passos necessários com o objetivo de realizar o reconhecimento e a decodificação de padrões de códigos de barras *EAN-13*. O trabalho descreve as principais características de um código de barras *EAN-13*, como as informações são representadas nesta simbologia, a forma que os números são codificados, os meios utilizados para assegurar a correta leitura de um código de barras e as principais ferramentas desenvolvidas com o objetivo de decodificar códigos de barras, partindo dos leitores do tipo caneta até se chegar aos leitores que utilizam câmeras como mecanismo de captura das informações. O texto também aborda os principais sistemas capazes de capturar imagens em formato digital como *scanners*, câmeras fotográficas digitais, filmadoras digitais e *webcams*, além do conceito de imagem digital e os principais problemas que podem ocorrer quando se utiliza câmeras como dispositivo de captura de imagens. O trabalho está focado no reconhecimento de padrões contidos em códigos de barras. No decorrer do texto foram descritos os conceitos e os principais métodos de reconhecimento de padrões como o método estático, sintático, redes neurais e conjuntos difusos. O texto também descreve as etapas que geralmente são encontradas em sistemas de reconhecimento de padrões como a captura da informação, pré-processamento, localização e decodificação dos padrões encontrados. Por fim o texto apresenta os métodos e as ferramentas utilizadas no desenvolvimento de um sistema que realiza o reconhecimento e a decodificação de padrões de código de barras *EAN-13* utilizando uma *webcam* para captura das imagens além dos testes realizados no sistema simulando diferentes problemas comuns de ocorrer quando se utiliza código de barras e a avaliação dos resultados.



**Palavras chave:** Reconhecimento de padrões, códigos de barras, *webcam*.

**ABSTRACT**

Nowadays, there are numberless advanced systems with the purpose to do reconnaissance of the pattern, whether in pictures or any other means by which to extract enough information. This work describes the steps necessary in order to achieve recognition and decoding standard barcodes EAN-13. The work describes the main features of a barcode EAN-13, how information is represented in symbols, the way that numbers are encoded, the means used to ensure correct reading of a barcode and the main tools developed in order to decode barcodes, starting with the readers of these pen until they reach the readers who use cameras as a mechanism for capturing information. The text also discusses about the major systems that can capture images in digital form such as scanners, digital cameras, digital camcorders and webcams, beyond the concept of digital image and the main problems that can occur when using cameras as a device to capture images. The work is focused on recognition of patterns contained in barcodes, during the text are described the concepts and main methods of pattern recognition as the static, syntactic, neural networks and fuzzy sets. The text also describes the steps that usually are found in systems of pattern recognition as the capture of information, preprocessing, location and decoding of patterns found. Finally the paper presents the methods and tools used to develop a system that performs recognition and decoding standard barcode EAN-13 using a webcam to capture images beyond of testing the system by simulating various problems commonly occurring when using bar code and evaluation of the results.

**Keywords:** Pattern recognition, barcodes, webcam.

## LISTA DE FIGURAS

## LISTA DE TABELAS

**SUMÁRIO**

# INTRODUÇÃO

Atualmente vivemos em uma sociedade onde as tecnologias estão presentes nas suas mais variadas formas. Porém, quando pensamos em tecnologia é comum fazermos associação com computares ou qualquer outro aparelho eletrônico que temos em casa ou no trabalho. Quando pensamos desta maneira não estamos errados, eles realmente são tecnologias, mas o que temos que pensar é que tecnologia não é só isso. Ela está presente em praticamente tudo que está a nossa volta, até mesmo em uma em uma folha de papel que utilizamos para escrever. Cabe a nos sabermos utilizar os recursos que a tecnologia nos oferece de modo que ela deixe mais funcional e eficiente o nosso dia-a-dia.

A Universidade do Estado de Mato Grosso possui inúmeros campi espalhados por diversas cidades do Estado de Mato Grosso. Todos esses campi têm seus documentos de tramitação interna e muitos destes documentos são constantemente enviados principalmente para a sede da Universidade, situada na cidade de Cáceres - MT. Durante esse processo de envio de documentos entre os campi eles podem se perder no caminho ou ficarem esquecidos em alguma repartição sem uma resposta por parte de quem o recebeu.

Pensando nisso surgiu a idéia de desenvolver um sistema capaz de fazer a identificação de padrões de códigos de barras para uma futura implantação

dentro da Universidade do Estado de Mato Grosso - Unemat, com o intuito de agilizar o processo de protocolização e acompanhamento de documentos internos da instituição além de proporcionar maior eficiência com baixo investimento, pois o sistema utiliza ferramentas de baixo custo como meio de captura de imagens.

Este processo até então é feito de forma manual, sem um controle eficiente, que poderia contribuir para o desaparecimento de documentos tanto enviados para outras localidades quanto os que forem recebidos pela instituição.

O sistema em funcionamento deverá fazer a localização e o reconhecimento de um padrão de código de barras, além de fazer a decodificação do mesmo, ou seja, extrair os caracteres que se encontram codificados em forma de barras no código.

# CAPÍTULO 1: HISTÓRICO E DETALHAMENTO DOS CÓDIGOS DE BARRAS

Este capítulo irá explicar o que é um código de barras e quais são as principais características que fazem dele um dos mais utilizados sistema de codificação de informações em todo o mundo. Aborda também um breve histórico dos códigos de barras dando ênfase ao padrão *EAN-13* que foi utilizado neste trabalho bem como sua estrutura, interpretação, regras para formação dos módulos e cálculo do dígito verificador.

## 1.1. Código de barras

Segundo Soares (2001) existem diversos tipos de código de barras que são destinados para áreas diversas. Cada tipo possui seu conjunto próprio de regras para que seja possível a decodificação de seus caracteres. Geralmente os padrões se diferenciam quanto ao seu aspecto visual, tipos de dados que suportam armazenar, bem como sua capacidade máxima de armazenamento, fazendo com

que um determinado padrão de código de barras possa não ser viável dependendo da finalidade da sua utilização.

> A tecnologia de código de barras foi desenvolvida visando a entrada de dados automatizados e a abolição de erros humanos de digitação. Esta tecnologia está destinada a quase todo tipo de aplicação onde se deseja automatizar a identificação de produtos, mercadorias, pessoas, documentos, etc. Tem sido extremamente útil na indústria, no comércio, em bancos e na identificação de pessoal, através de crachás, em portarias, etc. Portanto, não existe a necessidade de pessoas realizarem esta tarefa, pois um sistema poderá fazer de forma rápida e segura a mesma tarefa que seria feita por uma ou várias pessoas (MELLO, 2004).

## 1.2. Histórico

Segundo Florence (2009), no dia 26 de junho de 1974 um cliente do supermercado da cidade de *Troy*, estado de Ohio nos Estados Unidos, fez a primeira compra de um produto que possuía código de barras, tratava-se de um pacote de chicletes. Com isso deu-se início a uma nova era do mercado varejista.

De lá para cá surgiram diversos padrões de códigos de barras. Um deles foi o padrão *EAN.* Segundo Soares (2001) este padrão de código de barras começou a ser desenvolvido em 1974 quando 12 países se juntaram com o objetivo de desenvolver um novo sistema de codificação que seria utilizado tanto em produtos comerciais quanto industriais, porém, o padrão só começou a ser utilizado em 1976.

Segundo Costa (2009b), em junho de 1984 foi realizada uma reunião com empresários e executivos da indústria e do comércio que estavam interessados no desenvolvimento do processo de automação comercial. Com isso a Associação

Brasileira de Automação Comercial (ABAC) realizou o seu primeiro congresso a nível nacional. Nesse congresso chegou-se à conclusão de que era primordial a implantação do Código Nacional de Produtos[1] no Brasil. Após o congresso foi apresentada uma recomendação ao governo que posteriormente a transformou em Lei através do Decreto 90.595. O Decreto 90.595 de 29 de novembro de 1984, de acordo com o artigo 1º, cria o Sistema de Codificação Nacional de Produtos, ficando definido o padrão internacional *EAN* para todo o território nacional. Já em dezembro de 1984 foi publicada a portaria de nº. 143, do Ministério da Indústria e Comércio, que conferia à ABAC a responsabilidade de orientar e administrar a implantação do Código Nacional de Produtos no país.

Ainda segundo Costa (2009b), no ano de 1994 com o objetivo de fortalecer a imagem da entidade em todos os campos de atuação, a Associação Brasileira de Automação Comercial alterou sua sigla de ABAC para EAN BRASIL, com o intuito de tornar a associação reconhecida a nível nacional e internacional como representante legal responsável pela utilização dos sistemas *EAN* no Brasil. Atualmente a Associação Brasileira de Automação (GS1 Brasil) atua no sentido de estabelecer as normas técnicas necessárias, promover a cooperação entre parceiros comerciais, assegurar apoio aos empresários, divulgar novas tecnologias e, principalmente, incentivar a modernização.

### 1.3. Interpretação do padrão *EAN-13*

[1] Código nacional de produtos: O mesmo que código de barras (COSTA, 2009b).

Segundo Soares (2001) o código *EAN* é utilizado para identificar produtos em diversas partes do mundo e para isso foi criada uma identificação através dos 2 ou 3 primeiros dígitos do código de barras, que ficou conhecido como “bandeira”, onde cada país que utiliza esse padrão de código de barras possui um órgão responsável pelo cadastramento de cada produto de uma determinada empresa ou fabricante.

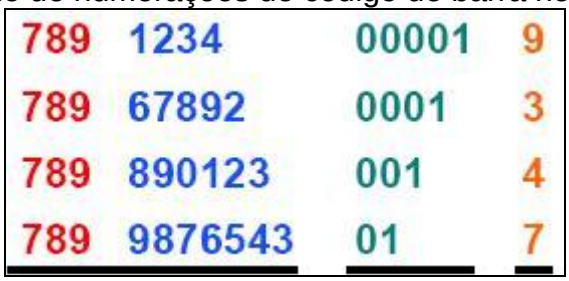
> O sistema *EAN* é um conjunto de padrões, que possibilita a identificação dos produtos, unidades logísticas e localizações. Ele facilita os processos de comércio eletrônico. Foi desenvolvido para fornecer soluções que garantam identificação exclusiva e sem ambigüidades. É um padrão internacional rígido onde cada produto terá seu código exclusivo, aplicável no mundo inteiro, sem repetição, o que possibilita a integração e a troca de informações entre os vários elos da cadeia produtiva, desde o fabricante até o consumidor final (SILVA e PAPANI, 2008).

Esse padrão é composto por 13 dígitos que podem ser divididos em quatro partes conforme a Figura 1, onde os 3 primeiros dígitos (representados em vermelho) são reservados para identificação da organização responsável por controlar e licenciar a numeração do código de barras no país. No Brasil, a empresa responsável pelo licenciamento dos códigos de barra é a GS1 Brasil[2] sendo seu código o número 789. Portanto, um código de barras com os três primeiros dígitos iguais a 789 indica que o produto está registrado no Brasil. Os próximos dígitos podem variar, podendo ter de 4 a 7 dígitos e representam a identificação da indústria dona da marca do produto, eles estão representados na Figura 1 pela cor azul. Os dígitos na cor verde representam a identificação de um determinado produto pela

[2] Disponível em: <http://www.gs1brasil.com.br>. Acesso em: 10 jul.2009.

indústria que o produziu e o último dígito (cor laranja) é chamado de dígito verificador que auxilia na segurança da leitura (RIBINIK, 2006a).

Figura 1. Exemplo de numerações de código de barra no padrão *EAN-13*.




Fonte: RIBINIK, 2006a.

É bom ressaltar que o código de barras para ser utilizado internamente dentro de uma instituição não precisa necessariamente seguir estas regras, elas são geralmente utilizadas quando uma empresa exporta seus produtos para outros países e necessitam de uma numeração única.

### 1.4. Como o código de barras *EAN-13* é formado

Segundo Soares (2001), o código de barras *EAN-13* é composto por 13 caracteres numéricos, dos quais apenas 12 estão representados no código na forma de módulos. Ainda segundo Soares (2001), módulos são a menor unidade da medida de largura de uma barra ou espaçamento e podem assumir os valores 0 (zero) ou 1 (um), onde os módulos com valores iguais a 1 (um) representam as

barras de cor preta, e aqueles com valores iguais a 0 (zero) são interpretados como barras brancas.

Segundo Ribinik (2009b), o código de barras “é a representação gráfica em barras claras e escuras das combinações binárias utilizadas pelo computador. Decodificadas por leitura óptica que informam os números arábicos ou letras que constituem o código de barras, conforme a simbologia”.

Os dígitos presentes no código de barras são uma combinação de módulos pretos (valor 1) e módulos brancos (valor 0), sendo que cada dígito do código de barras é composto por 7 módulos. No total, um código de barras padrão *EAN-13* é composto por 95 módulos, incluindo os módulos referentes às barras de início e fim do código de barras e também os módulos referentes às barras de divisão que existem entre os dígitos do lado esquerdo e os dígitos do lado direito do código de barras.

Por padrão, as barras que marcam o início e o fim do código de barras são formadas por 3 módulos cada uma e assumem os valores 101. As barras que fazem a divisão entre os dígitos do lado esquerdo e os dígitos do lado direito são formadas por 5 módulos, cujos valores são 01010. Para um melhor entendimento da estrutura do código de barras *EAN-13* é necessário entender primeiro como os módulos são ordenados para que possam representar um número propriamente dito.

Segundo Soares (2001), para gerar as barras do lado esquerdo do código de barras padrão *EAN-13* é utilizada uma tabela auxiliar de paridade, que irá definir a alternância entre duas outras tabelas, sendo elas a Tabela A (paridade impar) e a Tabela B (paridade par). A alternância entre estas duas tabelas é baseada no 1º ou no 13º dígito do código de barras, dependendo do sentido de leitura, ou seja, se o código for lido da direita para a esquerda o dígito de referencia

será o 13º, já se o sentido de leitura do código for da esquerda para a direita o dígito de referência será o 1º.

Para exemplificar o processo, o sentido de leitura utilizado será da esquerda para a direta, ficando assim o primeiro dígito do código como referência conforme representa a Tabela 3. A Tabela 1 corresponde à tabela auxiliar de paridades onde o dígito de referência define a alternância entre as tabelas A e B de padrões necessários para formar os módulos dos 6 dígitos do lado esquerdo do código.

Tabela 1. Tabela auxiliar de paridades.

| **Dígito de referência** | **Seqüência das tabelas utilizadas para formar os 6 dígitos do lado esquerdo do código** | | | | | |
|---|---|---|---|---|---|---|
| | **1º padrão** | **2º padrão** | **3º padrão** | **4º padrão** | **5º padrão** | **6º padrão** |
| **0** | A | A | A | A | A | A |
| **1** | A | A | B | A | B | B |
| **2** | A | A | B | B | A | B |
| **3** | A | A | B | B | B | A |
| **4** | A | B | A | A | B | B |
| **5** | A | B | B | A | A | B |
| **6** | A | B | B | B | A | A |
| **7** | A | B | A | B | A | B |
| **8** | A | B | A | B | B | A |
| **9** | A | B | B | A | B | A |

Fonte: SOARES, 2001.

Desta forma, se o dígito de referencia, por exemplo, for o número 7, deverá ser utilizada a seqüência de padrões presentes nas tabelas ABABAB para formar os 6 dígitos que estão representados por barras no lado esquerdo do código.

A Tabela 2 é formada pelas Tabelas A e B de padrões, onde na Tabela A estão os módulos com paridade impar, pois a soma dos módulos referente a um

dígito resulta em um número impar, enquanto que na Tabela B estão todos os módulos com paridade par, onde a soma dos módulos referentes a um dígito resulta em um número par. Além destas duas tabelas existe uma terceira tabela de padrões que será chamada de Tabela C e será utilizada para gerar os módulos correspondentes aos 6 dígitos do lado direito do código de barras.

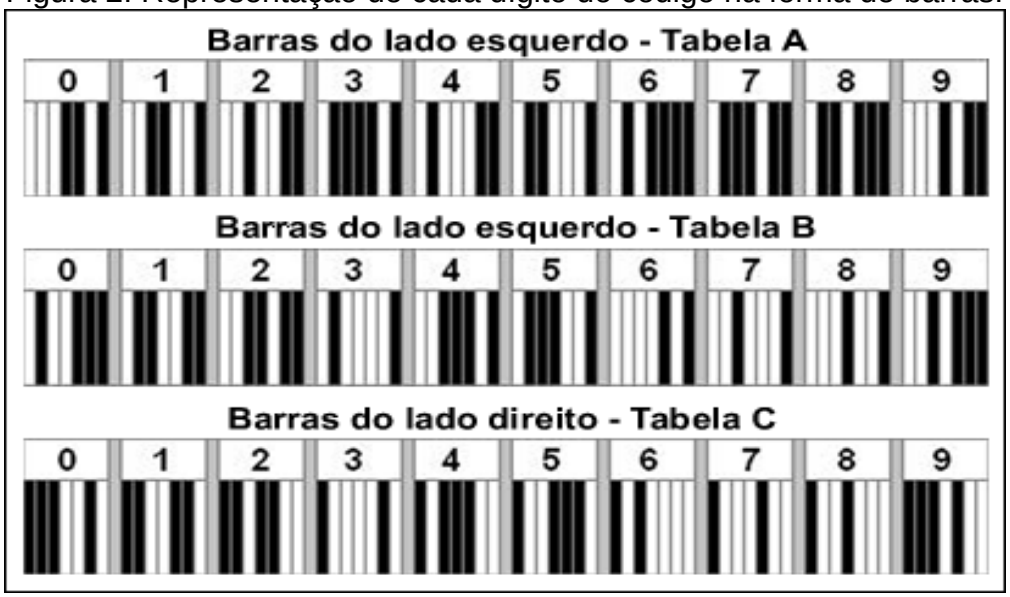
Tabela 2. Módulos necessários para formar cada dígito do código de barras.

| | **Dígitos do lado esquerdo** | | **Dígitos do lado direito** |
|---|---|---|---|
| **Dígito** | **Tabela A** | **Tabela B** | **Tabela C** |
| **0** | 0001101 | 0100111 | 1110010 |
| **1** | 0011001 | 0110011 | 1100110 |
| **2** | 0010011 | 0011011 | 1101100 |
| **3** | 0111101 | 0100001 | 1000010 |
| **4** | 0100011 | 0011101 | 1011100 |
| **5** | 0110001 | 0111001 | 1001110 |
| **6** | 0101111 | 0000101 | 1010000 |
| **7** | 0111011 | 0010001 | 1000100 |
| **8** | 0110111 | 0001001 | 1001000 |
| **9** | 0001011 | 0010111 | 1110100 |

Fonte: SOARES, 2001.

A Tabela 2 representa os módulos de cada dígito na forma de seqüência binárias, onde os zeros representam as barras brancas do código e os uns representam as barras pretas. Já a Figura 2 faz a representação de cada dígito do código na forma de barras.

Figura 2. Representação de cada dígito do código na forma de barras.




Fonte: BOGOMOLNY, 2009.

A Figura 3 ilustra o exemplo de um código de barras *EAN-13* que será utilizado para demonstrar o processo de formação dos módulos do lado esquerdo e do lado direito do código de barras.

Figura 3. Exemplo de código de barras *EAN-13*.


Fonte: FORD, 2002.

Observando as Tabelas 1 e 2, os módulos referente aos dígitos do lado esquerdo do código de barras (representado na Figura 3) ficariam distribuídos da

seguinte maneira conforme demonstra a Tabela 4. Lembrando que a seqüência de alternância entre a Tabela A (paridade impar) e a Tabela B (paridade par) é definida de acordo com o 1º do código de barras, que no caso do exemplo da Figura 3 é o dígito 1.

Tabela 3. Ordem dos dígitos do código de barras representado na Figura 3.

| 1º | 2º | 3º | 4º | 5º | 6º | 7º | 8º | 9º | 10º | 11º | 12º | 13º |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **0** | **1** | **2** | **8** |
| | Lado esquerdo | | | | | | Lado direito | | | | | |

Fonte: SOARES, 2001.

Tabela 4. Formação dos módulos de cada dígito do lado esquerdo do código.

| Primeiro dígito: **1** | | | Seqüência das tabelas: **AABABB** | | |
|---|---|---|---|---|---|
| **Dígito** | **2** | **3** | **4** | **5** | **6** | **7** |
| **Padrão** | 0010011 | 0111101 | 0011101 | 0110001 | 0000101 | 0010001 |
| **Tabela** | **A** | **A** | **B** | **A** | **B** | **B** |

Fonte: SOARES, 2001.

Para os dígitos do lado direito do código não existe alternância entre as tabelas de paridades A e B. Os seis dígitos do lado direito seguem a Tabela C que é fixa. Ainda tomando a Figura 3 como exemplo, será demonstrado como são formados os módulos de cada dígito do lado direito, representados na Tabela 5. Lembrando que o 13º dígito é chamado de dígito verificador e é calculado de acordo com os outros 12 dígitos do código de barras.

Tabela 5. Formação dos módulos de cada dígito do lado direito do código.

| **Dígito** | **8** | **9** | **0** | **1** | **2** | **8** |
|---|---|---|---|---|---|---|
| **Padrão** | 1001000 | 1110100 | 1110010 | 110 0110 | 1101100 | 1001000 |
| **Tabela** | **C** | **C** | **C** | **C** | **C** | **C** |

Fonte: SOARES, 2001.

É bom ressaltar que o primeiro dígito do código de barras padrão *EAN-13* não está representado em forma de barras, mas será utilizado juntamente com os outros 11 dígitos para efetuar o cálculo do décimo terceiro dígito do código de barras, também conhecido como dígito verificador. O processo necessário para a realização do cálculo do dígito verificador será detalhado a seguir.

## 1.5. Cálculo do dígito verificador

Segundo Soares (2001), o dígito verificador é útil para verificar se realmente o código de barras foi decodificado de maneira correta. Para realizar o cálculo do dígito verificador deve-se seguir alguns passos que serão detalhados a seguir, tomando como exemplo a Figura 3. Lembrando que nos exemplos o sentido de leitura está sendo da esquerda para a direita.

**Passo 1:** Somar todos os dígitos que se encontram em posições pares.

Tabela 6. Soma dos dígitos em posições pares.

| **Posição** | 2º | | 4º | | 6º | | 8º | | 10º | | 12º |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dígito** | 2 | + | 4 | + | 6 | + | 8 | + | 0 | + | 2 |
| **Total** | **22** | | | | | | | | | | |

Fonte: SOARES, 2001.

**Passo 2:** Multiplicar a soma do passo anterior por 3.

Tabela 7. Soma dos dígitos em posições pares multiplicado por 3.

| **Posição** | 2º | | 4º | | 6º | | 8º | | 10º | | 12º |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dígito** | 2 | + | 4 | + | 6 | + | 8 | + | 0 | + | 2 |
| **Total** | **22 x 3 = 66** | | | | | | | | | | |

Fonte: SOARES, 2001.

**Passo 3:** Somar os dígitos que se encontram em posições impares.

Tabela 8. Soma dos dígitos em posições impares.

| **Posição** | 1 | | 3 | | 5 | | 7 | | 9 | | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Dígito** | 1 | + | 3 | + | 5 | + | 7 | + | 9 | + | 1 |
| **Total** | **26** | | | | | | | | | | |

Fonte: SOARES, 2001.

**Passo 4:** Somar os valores encontrados nos passos 2 e 3.

Tabela 9. Soma total.

| | |
|---|---|
| **Resultado do Passo 2** | 66 |
| **Resultado do Passo 3** | 26 |
| **Total** | **92** |

Fonte: SOARES, 2001.

**Passo 5:** O menor número que somado ao total do Passo 4, de modo que o total seja um número múltiplo de 10 será o dígito verificador.

**92 + 8 = 100**

Portanto o dígito verificador neste caso é o número 8. Quando o total do passo 4 já resultar em um múltiplo de 10, o dígito verificador assumirá o valor 0 (zero).

## CAPÍTULO 2: FORMAS DE AQUISIÇÃO DE INFORMAÇÃO

Este capítulo aborda uma breve introdução sobre o conceito de imagem digital. Também aborta sobre os principais métodos e dispositivos de leitura e decodificação de código de barras existentes atualmente, descrevendo as principais características dos equipamentos, a tecnologia utilizada por eles, suas vantagens e desvantagens. Em seguida serão citadas as principais ferramentas destinadas à captura de imagens e vídeo em formato digital, descrevendo as principais diferenças de cada ferramenta. Por fim o capítulo aborda os principais problemas que podem ocorrer durante a captura de imagens de códigos de barras através da *webcam.*

### 2.1. Conceito de Imagem digital

Segundo Silva (2004), uma imagem digital é composta por uma combinação de matrizes bidimensionais sobrepostas onde cada ponto desta matriz[3]

[3] Matriz: Disposição bidimensional de linhas e colunas usada na organização e comparação de dados (MATIAS e RODRIGUES, 2009).

corresponde a um *pixel*[4] da imagem e pode assumir valores que variam de 0 a 255. Cada uma das matrizes representa uma camada de cor que determinam o modo de coloração da imagem.

> Os modelos mais populares são o RGB (red, green, blue) para monitores coloridos, e o CMY (cyan, magenta, yellow) para impressoras coloridas; são modelos orientados para o hardware. Outros modelos existentes como HSI (hue, saturation, intensity) ou (matiz, saturação, intensidade) e o HSV (matiz, saturação e valor), por exemplo, são modelos orientados para a manipulação de cores (MIANO *apud* CARVALHO, MENEGUETE JUNIOR e SILVA, 2003).

Uma imagem no padrão *RGB*, por exemplo, é composta de 3 camadas de cores, sendo uma para a cor vermelha (*Red*), uma para a cor verde (*Green*) e uma para a cor azul (*Blue*). Estas três camadas de cores quando são combinadas formam as demais cores.

> Uma imagem colorida é uma imagem multibanda, onde a cor em cada ponto (x,y) é definida através de três grandezas: luminância, matiz e saturação. A luminância está associada com o brilho da luz, a matiz com o comprimento de onda dominante e a saturação com o grau de pureza (ou intensidade) da matiz. A maioria das cores visíveis pelo olho humano pode ser representada como uma combinação de três cores primárias: vermelho (R), verde (G) e azul (B). Assim, uma representação comum para uma imagem colorida utiliza três bandas R, G, e B com profundidade 1 byte por pixel (FALCÃO e LEITE, 2003).

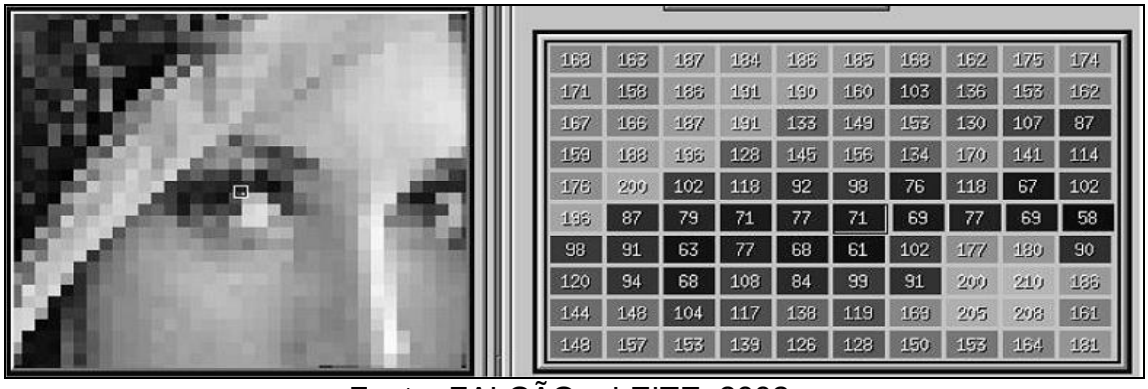
Em sistemas destinados ao reconhecimento de padrões em imagens, geralmente não se trabalha com imagens coloridas a não ser que sejam sistemas

[4] *Pixel*: Acrônimo de *Picture Element*. É o menor elemento de uma imagem digital ou digitalizada (SILVA, 2006).

que irão depender de algum processamento baseado em alguma cor específica como, por exemplo, detecção de pele.

É usual utilizar imagens em tons de cinza, pois facilitam as etapas seguintes como a binarização da imagem e diminuem o custo de processamento, uma vez que a imagem é formada por uma única matriz de *pixels*, onde a intensidade de cada *pixel* pode variar de 0 a 255, sendo que quanto maior for o valor da intensidade do *pixel* mais claro será o ponto na imagem. A Figura 4 ilustra uma imagem em tons de cinza e ao lado pode ser vista a matriz com os valores de intensidade dos níveis de cinza em uma área de 10 x 10 *pixels* situada no olho da figura.

Figura 4. Exemplo de imagem monocromática e a matriz de intensidade dos níveis de cinza.




Fonte: FALCÃO e LEITE, 2003.

## 2.2. Aquisição de dados

Atualmente vivemos em um mundo onde a informática está presente em quase todos os ambientes. Os avanços nesta área da tecnologia proporcionam

maior rapidez e agilidade na realização de tarefas do dia-a-dia, como digitação de textos, fazer compras sem sair de casa, comunicação com pessoas que estão distantes, inclusive com a possibilidade de visualizar a pessoa através de câmeras ligadas ao computador. Estas mesmas câmeras que utilizamos para uma conversa na internet podem ser utilizadas como uma ferramenta de baixo custo para realizar tarefas que não são convencionais para ela.

Com uma simples *webcam* é possível dar "visão" ao computador. Utilizando dispositivos de captura de imagens e algoritmos capazes de tratar as imagens obtidas, podemos identificar objetos de interesse nas imagens. Algumas tarefas consideradas repetitivas e cansativas podem ser feitas com muito mais agilidade e rapidez se forem feitas por uma máquina. Tarefas como, por exemplo, digitar números de códigos de barras.

Os humanos conseguem observar e identificar os objetos através da visão. A visão humana é impressionante ao se considerar a rapidez na aquisição, processamento e identificação de objetos. Através da visão consegue-se perceber o mundo ao redor, possibilitando a tomada de decisões e execução de atividades (CONTE e PEZZIN, 2005).

No mundo da Informática não é muito diferente, câmeras acopladas ao computador passam a atuar como a visão humana e recebem o nome de visão computacional, que pode ser definida como "uma área da Ciência da Computação que tem por objetivo modelar o mundo real ou reconhecer objetos, transformando-os em imagens digitais." (MILAN *et. al. apud* PINTO e REINBRECHT, 2007).

## 2.3. Dispositivos de leitura de código de barras

Segundo Fernandes (2001), “um dispositivo de leitura de código de barras, transforma os dados codificados em barras em sinais elétricos de duração proporcionais as barras”.

> Código de barras é a representação gráfica, em barras claras e escuras, das combinações binárias utilizadas pelo computador. Através de um scanner, as combinações são decodificadas por meio de leitura óptica. Desta forma, o scanner detecta os números binários representados pelas barras, que são equivalentes ao número que aparece logo abaixo delas (SILVA e PAPANI, 2008).

É comum encontrar em estabelecimentos comerciais, algum tipo de aparelho que faça a leitura dos códigos de barra dos produtos. O uso destas leitoras de código de barras agilizam muitas das atividades que envolvam os códigos de barras. Segundo Castro (2007) um leitor de códigos de barra geralmente é composto por uma fonte de luz, uma lente e um condutor ótico que traduz impulsos óticos em elétricos.

> O leitor ótico usa um foto-sensor (dispositivo que converte energia luminosa em energia elétrica) para converter o Código de Barras em um sinal elétrico que será transformado em bits à medida que percorre o código, calculando a relação entre as larguras de barras e espaços. Dessa forma, ocorre uma tradução dos diferentes padrões impressos em caracteres inteligíveis a um sistema computadorizado (SOARES, 2001).

Atualmente existem no mercado diversos tipos de leitoras de códigos de barras, cabe ao usuário escolher a que melhor se adapta á suas necessidades. Na seqüência serão descritos alguns modelos de leitores de códigos de barras disponíveis no mercado e suas principais características.

### 2.3.1. Leitoras do tipo caneta

Segundo Castro (2007), leitores do tipo caneta são constituídos por uma fonte de luz e um fotodiodo[5] que são colocados na ponta de uma caneta. O papel do fotodiodo é medir a intensidade da luz refletida, gerando uma onda que é utilizada para fazer a medição da largura das barras e dos espaços presentes no código de barras. As áreas escuras (barras) absorvem a luz enquanto que as áreas claras (espaços) refletem a luz.

> A caneta ótica é um dispositivo leitor de código de barra que tem em sua ponta um emissor de luz normalmente vermelho e um sensor para receber a reflexão dessa luz (ou não). Assim se a caneta ótica for colocada em uma superfície branca, a luz emitida vai refletir na superfície e o sensor vai captar essa reflexão e gerará um sinal elétrico que corresponde a superfície branca. De maneira oposta, se a caneta for posicionada em cima da superfície escura, normalmente preta, a luz emitida da caneta será absorvida e o sensor não recebera nenhum reflexo. Então o sensor irá gerar um sinal elétrico correspondente a cor preta. Se a caneta for passada sobre uma etiqueta de código de barra com velocidade constante, a caneta irá gerar uma seqüência de sinais elétricos conforme a mesma vai passando pelas barras (largas e finas) pretas e espaços em branco. (FERNANDES, 2001).

[5] Fotodiodo: O fotodiodo é um diodo de junção construído de forma especial, de modo a possibilitar a utilização da luz como fator determinante no controle da corrente elétrica (SOUZA e PEREIRA, 2009?).

Segundo Fernandes (2001), através da luz que foi refletida pelos módulos que compõem os espaços do código de barras ou pela sua ausência nas barras, o leitor consegue interpretar o código. A interpretação é feita através de um conversor[6] analógico/digital que irá transformar os sinais elétricos analógicos[7] produzidos pela luz recebida em um sinal digital[8] utilizando um sensor fotoelétrico. Deste modo, o sensor ira gerar uma seqüência de pulsos que poderão ser 0 ou 1. O número 0 (zero) é gerado pela reflexão da luz no código e representam as barras brancas (espaços) enquanto que o número 1 (um) é gerado pela ausência de reflexão da luz e representa as barras escuras do código, ou seja, cada módulo (barra clara ou escura) é formado por um número que poderá ser zero ou um.

Deste modo, cada dígito do código de barras é formado por uma seqüência binária. Por exemplo, a seqüência barra (1), espaço (0), espaço (0), barra (1), barra (1), barra (1), espaço (0), para uma leitora gera a seqüência binária 1001110 que irá representar o dígito 5 na Tabela C do padrão *EAN-13.*

Segundo Silva e Papani (2008), as combinações binárias que representam as barras claras e escuras do código de barras são baseadas em critérios diferentes para cada simbologia[9]. Cada simbologia possui um conjunto de padrões próprio que são organizados de uma determinada forma, podendo codificar caracteres numéricos e alfanuméricos.

---

[6] Conversor: Dispositivo eletrônico utilizado em câmaras digitais e *scanners* para quantificar as cargas elétricas registradas pelo CCD (PARANHOS, 2009).

[7] Analógico: Sinal elétrico ou forma de onda, cuja amplitude e freqüência variam continuamente (PELEGRINA, 2008a).

[8] Digital: Todo dispositivo que gera, armazena e transmite dados codificados pelo sistema binário. Os dados são numericamente representados pelos algarismos um e zero (PELEGRINA, 2008b).

[9] Simbologia é a forma como são padronizadas as regras que estabelecem quantidade e larguras das barras e espaçamentos a fim de expressar uma palavra-código (SOARES, 2001).

Figura 5. Exemplo de leitor de código de barras do tipo caneta.

Fonte: Disponível em: <http://www.kebelc.com.br/leitor_de_codigo_barras_caneta_cw100.htm>. Acesso em: 20 set. 2009.

### 2.3.2. Leitoras do tipo *laser scanner*

De acordo com Freire e Grilo (2008), nos equipamentos do tipo *scanner* o feixe de luz oscilam constantemente permitindo a leitura dos códigos em pequenos intervalos de tempo. Em todas estas situações o operador tem necessidade de apontar o feixe de luz para o código de barras para que seja possível a sua leitura. Os leitores do tipo manual com tecnologia *laser*[10] podem fazer a leitura sem contato direto entre a etiqueta de código de barras e o *scanner*, podendo decodificar etiquetas planas, curvas e em superfícies irregulares.

> Os leitores a *laser* trabalham da mesma maneira que os leitores de caneta, exceto que usam um feixe de *laser* como fonte de luz e tipicamente usam um espelho ou prisma giratório para fazer a varredura do feixe de *laser* para a frente e para trás ao longo do código de barra. Um fotodiodo é usado medir a intensidade da luz refletida de volta do código de barra. Em ambos os tipos de leitores, a luz emitida pelo leitor se dá numa freqüência específica e fotodiodo é desenhado para detectar somente esta freqüência (CASTRO, 2007).

[10] Laser: Sigla de *Light Amplification by Stimulated Emission of Radiaton*, que significa "ampliação de luz por meio da emissão estimulada de radiações" (CLAUDIO, FACHINI e RAMOS, 1996).

Figura 6. Exemplo de leitor de código de barras do tipo laser scanner.

Fonte: Disponível em: <http://www.equipashop.net/index.asp?Produto=Leitor-de-Codigo-de-Barras-Laser-LS2208&Email=Tradepar>. Acesso em: 20 set. 2009.

### 2.3.3. Leitoras do tipo *CCD*

Segundo Castro (2007), os leitores do tipo *CCD* (*Charge Coupled Device*) trabalham utilizando uma matriz com centenas de minúsculos sensores de luz alinhados na forma de uma linha na cabeça do leitor, onde cada um dos sensores mede a intensidade de luz recebida, gerando um padrão idêntico ao do código de barras que está sendo lido. Possuem algumas vantagens por serem baratos, leves e por consumirem pouca energia e desvantagem com relação à limitação da distância entre o leitor e o código que se deseja decodificar.

Figura 7. Exemplo de leitor de código de barras do tipo CCD.

Fonte: Disponível em: <http://www.tecnoimports.com.br/index.asp?secao=19&categoria=90&subcategoria=0&id=144>. Acesso em: 20 set. 2009.

### 2.3.4. Leitoras baseadas em câmeras

Segundo Castro (2007), os leitores baseados em câmeras são os mais modernos. Trata-se de um leitor que utiliza uma pequena câmera de vídeo que captura a imagem do código de barras. Após a captura o leitor passa a processar a imagem com o objetivo de decodificar as barras existentes no código. Os fatores que podem interferir no processo de decodificação são o baixo contraste da imagem, falhas de impressão do código, sujeira na superfície do código, defeitos nas barras ocasionados pelo efeito de espalhamento de tinta.

“Estes aparelhos não são muito populares por causa do seu preço elevado, mas prevê-se que substituirão os *scanners laser* dos supermercados num futuro próximo” (ALCOBIA, 2007).

Figura 8. Exemplo de leitor de código de barras baseado em câmera.

Fonte: Disponível em: <http://www.kioskbrasil.com.br/dbimagens/fotos/b0a08128e370e598fe7a7ce6643897de.jpeg>. Acesso em: 20 set. 2009.

## 2.4. Dispositivos responsáveis pela captura de imagens

Existem inúmeros dispositivos responsáveis por capturar imagens em formato digital, alguns com custo elevado e outros a preços bem acessíveis para a maioria dos usuários. Para Tavares (*et. al.* 2007), a captura de imagens é um processo que utiliza dispositivos ou equipamentos para converter imagens em dados digitalizados. Trata-se do processo da representação digital de determinada imagem que poderá ser armazenada ou manipulada com uso de *softwares* ou códigos específicos.

> As imagens são produzidas por uma variedade de dispositivos físicos, tais como câmeras e vídeo câmeras, equipamentos de radiografia, microscópios eletrônicos, magnéticos e de força atômica, radares, equipamento de ultra-som, entre vários outros. A produção e utilização de imagens podem ter diversos objetivos, que vão do puro entretenimento até aplicações militares,

> médicas ou tecnológicas. O objetivo da análise de imagens, seja por um observador humano ou por uma máquina, é extrair informações úteis e relevantes para cada aplicação desejada (ESQUEF, 2002).

Após a imagem ser digitalizada por algum dispositivo ela poderá ser armazenada, por exemplo, no disco rígido de um computador ou algum outro dispositivo capaz de armazenar imagens em formato digital. A partir daí, “a imagem armazenada pode ser processada para cumprir os mais variados objetivos: compressão, melhoramento, reconhecimento de padrões, etc.” (FALCÃO e LEITE, 2003).

### 2.4.1. *Scanners*

De acordo com Scuri (1999) os *scanners* são semelhantes às máquinas foto copiadoras e estão basicamente subdivididos em três tipos: manuais, de mesa ou de cilindro.

> No caso de *scanners* manuais a varredura é feita deslizando o aparelho sobre a imagem a ser capturada. No de mesa, em processo análogo ao da *Xerox*, a varredura é realizada automaticamente pelo aparelho que desloca a fonte de luz e os foto-detetores. No de cilindro tanto a fonte de luz quanto a imagem se movem para fornecer maior qualidade ao processo. Neste caso a fonte de luz se move dentro do cilindro enquanto a imagem é fixada na sua superfície que gira durante o processo (SCURI, 1999).

Ainda segundo Scuri (1999), independente do tipo do *scanner* ele será composto por uma fonte de luz em forma de uma linha que percorre a imagem impressa medindo a quantidade de luz refletida ou transmitida em cada ponto. Em seguida a imagem é convertida em sinal elétrico através de um conjunto de fotodetectores[11]. Logo após o sinal elétrico é finalmente digitalizado e enviado ao computador.

"[...] *scanners* provem as imagens de mais alta resolução, mas em compensação são mais caros. Podem capturar diversos tipos de superfície, as mais importantes são papel e filmes fotográficos" (SCURI, 1999).

### 2.4.2. Câmeras digitais

Segundo Cappelaro (*et. al.* 2004), já se pensava em capturar imagens sem a utilização de filmes[12] desde 1908, quando Archibald Campbell Swinton propôs uma forma de se capturar imagens de forma eletrônica. Porém, naquela época os avanços nessa área da tecnologia não foram suficientes para que o projeto pudesse ser realizado. O projeto que Archibald Campbell Swinton havia desenvolvido só se tornou realidade no final da Segunda Guerra Mundial. A tecnologia de vídeo foi à primeira idéia para se substituir as câmeras fotográficas analógicas tradicionais. No início dos anos, 80 a *Sony* lançou uma câmera fotográfica que não utilizava filmes, as fotos eram armazenadas em disquetes de 2 polegadas semelhantes aos

[11] Fotodetectores: Os fotodetectores produzem uma corrente elétrica proporcional à intensidade da luz por eles absorvida (MINAS, 2005-2006).
[12] Filme: Película transparente e fotossensível na qual se registra a imagem por meio de processo fotográfico (FUNGHI, 2007).

utilizados em computadores. O sinal de vídeo era gravado no disquete em formato analógico, necessitando de acessórios para fazerem a conversão analógico/digital para que as imagens fossem utilizadas em computadores.

> Em seu nível mais básico, uma câmera digital, assim como uma câmera convencional, possui uma série de lentes que focaliza a luz para criar a imagem de uma cena. Mas em vez de focalizar essa luz sobre um pedaço de filme, ela o faz sobre um dispositivo semicondutor que grava a luz eletronicamente. Um computador então decompõe essas informações eletrônicas em dados digitais (MARCONI, 2008).

Segundo Nicioli (2009), a principal diferença que existente entre as fotos de câmeras comuns e as fotos de câmeras digitais está no tipo de armazenamento. Em uma foto tirada com uma câmera comum o armazenamento e feito em um filme que altera sua química em função da exposição à luz, já no caso de fotos obtidas com câmeras digitais o *CCD* converte a intensidade de luz que incidem sobre ele em valores digitais armazenáveis na forma de *bits* e *bytes*[13]. Leite (2009) cita que “o sensor de imagem é o “filme” em uma máquina fotográfica digital, é o dispositivo que de fato captura “o quadro””.

As câmeras fotográficas digitais podem conter geralmente dois tipos de sensores, os do tipo *CCD* e os do tipo *CMOS*. Segundo Leite (2009), “o sensor de imagem empregada pela maioria das máquinas fotográficas digitais é um dispositivo chamado de CCD”.

[13] *Byte*: Unidade que permite medir a quantidade de informação e é, normalmente, o conjunto de oito *bits* (CARDOSO *et.al.* 2009).

> O CCD é uma coleção de diodos sensíveis á luz, minúsculos, que convertem fótons (luz) em elétrons (custo elétrico). Estes diodos são chamados photosites. A luz incide sobre o CCD repleto de photosites, recebida por estes e convertida em sinais elétricos, mas estes sinais elétricos que são “construídos” no CCD não são sinais digitais que estão prontos para ser usados por seu computador. Para digitalizar as informações, os sinais elétricos devem ser passados por um conversor analógico/digital (ADC). Esta tarefa é controlada por um microprocessador instalado dentro da câmera digital. O processo pelo qual o CCD converte imagens em imagens eletrônica é chamado conversão fotoelétrica. [...] Os sensores CMOS são muito mais baratos de fabricar do que os sensores CCD. Eles não necessitam de uma linha de produção totalmente dedicada, assim o preço diminui porque os custos fixos da produção são distribuídos em um muito maior número de dispositivos utilizados por diferentes aparelhos eletrônicos e não apenas nos sensores. Sensor de CMOS geralmente tem qualidade menor, assim como a resolução e sensibilidade. Porém ele permite vida longa da bateria, pois consomem menos energia do que o CCD (LEITE, 2009).

### 2.4.3. Câmeras de vídeo digital

Segundo Ayres (2007), as filmadoras digitais possuem um sistema que faz a conversão analógica para digital e em seguida grava essas informações em formato de *bytes* e não em formato de trilhas contínuas de padrões magnéticos, sendo esta a principal vantagem das câmeras digitais em relação às câmeras tradicionais analógicas, pois os filmes gerados por elas podem ser transferidos como arquivos eletrônicos compostos por zeros e uns e sem perda de qualidade, o que não acontecia quando os vídeos eram armazenados em fitas magnéticas.

> As filmadoras digitais têm em comum um sistema que transforma a informação analógica em digital e grava essas informações em *bytes* ao invés de trilhas contínuas de padrões magnéticos. Essa é uma vantagem importante das digitais em relação às analógicas, pois podem transferir imagens como arquivos eletrônicos compostos de zeros e uns, sem perder a qualidade como acontecia com as fitas de vídeo tradicionais (AYRES, 2007).

O funcionamento do *CCD* de uma câmera de vídeo é muito similar a uma câmera fotográfica digital. A diferença é que nas filmadoras as imagens são gravadas como pequenas fotos tiradas em seqüência em espaço de tempo minúsculo para depois serem agrupadas para dar a sensação de movimento (AYRES, 2007).

### 2.4.4. *Webcams*

Segundo Oliveira (2009) uma *webcam* é uma câmera de vídeo que foi projetada para computadores que acessam a Internet. Existem diversos modelos de *webcams*, sendo a resolução[14] da imagem e o número de *FPS*[15] suas principais características. A qualidade de uma *webcam* está relacionada com a capacidade de capturar imagens com resoluções altas, pois quanto maior a resolução da *webcam* melhor será a qualidade da imagem capturada. Stemmer (*et. al*. 2005) cita que:

> Desenvolvidas inicialmente para o ambiente doméstico e de escritório, mas já ganham espaço até mesmo em aplicações industriais. Tratam-se de câmeras coloridas digitais, com varredura progressiva e boas taxas de transferência de dados, já apresentando até mesmo alta resolução de imagens (megapixel). São em geral de pequeno tamanho, peso e baixo custo. Possuem um conjunto óptico próprio e fixo, o que dificulta seu uso para aplicações de metrologia. São atualmente bem empregadas em vídeo-conferências, sistemas de vigilância e monitoramento de ambientes, auxílio a deficientes físicos, dentre outros (STEMMER *et al.* 2005).

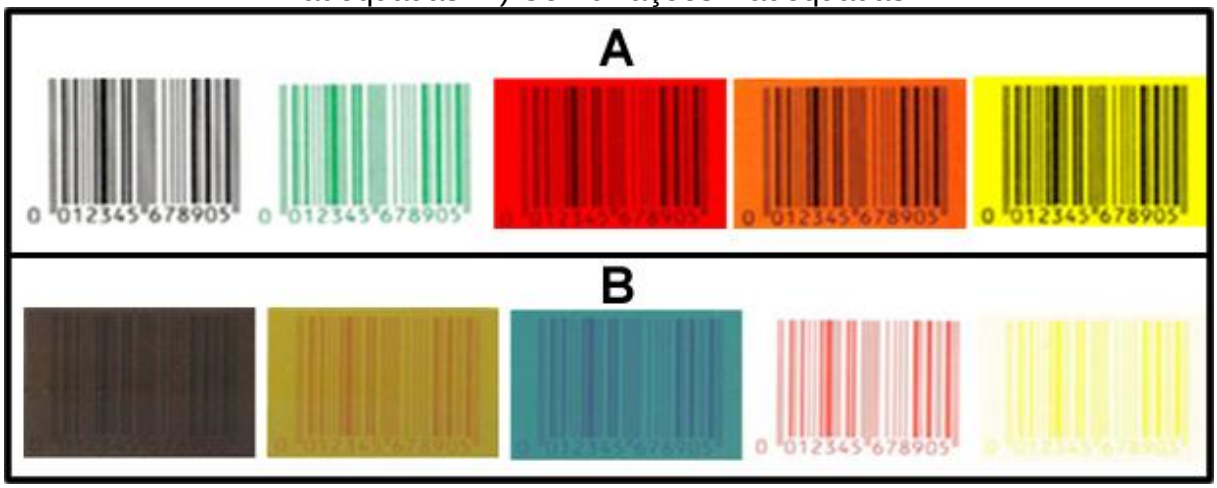
[14] Resolução: Quantidade de pixels que uma imagem digital tem por polegadas (medida mais difundida no mercado). Quanto mais alta a resolução de uma imagem, mais qualidade ela terá para impressão em papel (PARANHOS, 2009).
[15] *FPS*: Quadros por segundo. Expressão relacionada ao *frame-rate*, que indica o número de quadros por segundo em um jogo, vídeo, ou qualquer outro tipo de imagem em movimento (REIS, 2005).

## 2.5. Problemas durante a captura de imagens

Inúmeros fatores podem interferir durante o processo de captura da imagem e podem fazer com que o código de barras não seja decodificado ou que ele seja decodificado de forma incorreta. Fatores como a cor do código de barras, códigos danificados, a distancia entre o código de barras e a *webcam* e se tratando de leitura por uma *webcam* também deve ser levado em conta o fator luminosidade.

Segundo Ribinik (2006a), existem combinações de cores que são adequadas para os códigos de barras e também as combinações inadequadas. A Figura 9 ilustra quais são as possibilidades adequadas de combinação e as combinações que são consideradas inadequadas para códigos de barra.

Figura 9. Combinações de cores para códigos de barras. A) Combinações adequadas. B) Combinações inadequadas.




Fonte: RIBINIK, 2006a.

Lembrando que estas combinações de cores foram baseadas em leitores óticos de código de barras e, portanto, algumas destas combinações consideradas apropriadas para códigos de barras poderão não funcionar adequadamente quando se for utilizar a *webcam* como meio de captura da imagem.

Defeitos nos códigos de barras causados por falhas de impressão ou pelo fato do papel onde o código estiver afixado estar amassado ou manchado por algum produto poderão gerar problemas graves, pois passarão informações incorretas ao sistema que por sua vez poderá não ser capaz de identificar e decodificar o código de barras.

A luminosidade é outro ponto importante, pois ela pode interferir na captura da imagem e posteriormente na decodificação do código. Para Freire (2008), “o código de barras tem que estar visível para ser lido, a sua leitura só é realizada no sentido do código de barras e de perto”. Portanto todos esses fatores devem ser levados em consideração para proporcionar a captura de uma imagem de qualidade que facilitará a localização e a decodificação do código de barras.

# CAPÍTULO 3: CONCEITOS E MÉTODOS DE RECONHECIMENTO DE PADRÕES

Este capítulo ira abordar sobre o que é o reconhecimento de padrões com um breve histórico sobre o assunto, citando as principais dificuldades que existiam na época de seu surgimento e outras que ainda continuam a existir bem como as vantagens de se utilizar sistemas destinados ao reconhecimento de padrões. E em seguida serão descritos alguns métodos de reconhecimento de padrões. Será abordado também sobre as etapas de um sistema de reconhecimento de padrões em imagens digitais que vão desde a captura da uma imagem até se chegar ao reconhecimento de um padrão propriamente dito.

## 3.1. Reconhecimento de padrões

Os estudos envolvendo reconhecimento de padrões se iniciaram antes da década de 60, porém nesse período os estudos ficaram restritos principalmente a pesquisas teóricas e estatísticas. Com o aumento do número de computadores houve também um aumento pela procura de sistemas que abordassem o

reconhecimento de padrões. Atualmente o reconhecimento de padrões é encontrado em grande parte dos sistemas inteligentes que são desenvolvidos com o objetivo de tomar algum tipo de decisão (THEODORIDIS e KOUTROUMBAS *apud* NASCIMENTO, 2007).

> Essas características podem ser obtidas em sistemas de analise de imagens projetados e utilizados em ambientes operacionais limitados. Ou seja, sistemas específicos, desenvolvidos para reconhecimentos de padrões de uma classe restrita de imagens. No entanto, o desenvolvimento de um sistema para reconhecimento de qualquer padrão de objetos em imagens, que se assemelhe a capacidade humana para realizar esta tarefa, ainda não foi eficientemente idealizado (NASCIMENTO, 2007).

Portanto, um sistema computadorizado até o presente momento ainda não se iguala à capacidade que o ser humano possui de identificar objetos presentes não só em imagens, mas em qualquer coisa que possa ser visualizado por seus olhos. Desta forma, “[...] uma das principais metas de análise de imagens automaticamente é dotar um computador com a capacidade similar dos seres humanos de reconhecer e compreender uma imagem a partir de um padrão visto anteriormente” (GONZALEZ e WOODS, 2005).

Segundo Souza (1999), os sistemas de reconhecimento de padrões possuem grande abrangência na área da ciência, pois através deles podem ser realizados estudos em diversos campos de pesquisas como biologia, psicologia, medicina, marketing, finanças, meteorologia, sensoriamento remoto, processamento de imagens, entre outros. Castro e Prado (2007) definem reconhecimento de padrões como:

> Um procedimento em que se busca a identificação de certas estruturas nos dados de entrada em comparação a estruturas conhecidas e sua posterior classificação dentro de categorias, de modo que o grau de associação seja maior entre estruturas de mesma categoria e menor entre as categorias de estruturas diferentes.

O reconhecimento de padrões pode ser classificado como supervisionado e não supervisionado. Segundo Góes e Siqueira (2005), na “[...] classificação supervisionada as classes são definidas pelo projetista do sistema enquanto que na não supervisionada, as classes são aprendidas de acordo com a similaridade dos padrões”.

## 3.2. Métodos de reconhecimento de padrões

Existem diversos métodos que foram desenvolvidos para serem utilizados em sistemas de reconhecimento de padrões, cada um com uma funcionalidade diferente, destinados a reconhecer padrões de diversas formas. Nas seções seguintes serão descritos 4 métodos de reconhecimento de padrões mais utilizados quando se pretende desenvolver sistemas que possuam tal capacidade.

### 3.2.1. Reconhecimento de padrões estatístico

> A forma mais geral e natural de formular soluções para o reconhecimento de padrões é o RP estatístico, através do qual é reconhecida a natureza estatística tanto da informação que se quer representar quanto dos resultados que devem ser expressados (BISCHOP *apud* RAITTZ, 1997).

Segundo Campos (2001), o nome deste método se justifica devido ao fato da classificação de um determinado padrão ser realizada utilizando-se estimativas de distribuições probabilísticas, onde o reconhecedor de padrões é avaliado por um conjunto de testes composto preferencialmente por padrões de todas as classes, desde que estas classes não façam parte do conjunto de treinamento.

> Os algoritmos de RP baseados em estatística subdividem o problema de classificação em duas tarefas distintas: a extração de características e a comparação casamento destas características com as de modelos perfeitos, livres de ruídos, representativos dos seus respectivos padrões. Estas tarefas são realizadas por dois módulos denominados de extrator de características e classificador (DEVIJVER e KITTLER *apud* LUDWIG JUNIOR, 2004).

### 3.2.2. Reconhecimento de padrões sintático

Segundo Raittz (1997), a base do reconhecimento de padrões sintático está no fato da inter-relação ou a interconexão das características produzirem informações importantes e que muitas das vezes essas informações não são constituídas apenas pela existência ou ausência de valores numéricos de conjuntos de características. Para utilizar o método de reconhecimento de padrões sintático

inicialmente deve-se extrair as informações estruturais consideradas importantes, para só então realizar a analise de similaridade entre os padrões.

> Tipicamente, o RP sintático formula uma descrição hierárquica de padrões complexos, construída a partir de sub-padrões mais simples, sendo que no nível mais baixo, se encontram os elementos mais simples, extraídos dos dados de entrada que são chamados de primitivas (RAITTZ, 1997).

Castro e Prado (2007) citam que este método é particularmente eficiente em padrões que não podem ser descritos através de medidas numéricas como é convencional ou por serem muito complexos a ponto de suas características não poderem ser mencionadas, transformando-se em características globais.

### 3.2.3. Reconhecimento de padrões por redes neuronais

Segundo Fausett (*apud* GUIMARÃES e CHAVES NETO, 2006), as redes neurais surgiram na década 50 com a necessidade de se compreender o funcionamento do cérebro humano e reproduzir algumas de suas características como a grande capacidade de conexão entre os neurônios e também a tolerância ás falhas.

Segundo Guimarães e Chaves Neto (2006), uma rede neural é formada por diversos elementos de processamentos chamados de neurônios. Esses elementos ficam dispostos em camadas e a eles são associados pesos. O processo necessário para fazer o cálculo do peso é chamado de treinamento ou

aprendizagem da rede. O treinamento pode ser de dois tipos: supervisionado e não supervisionado.

> O treinamento pode ser supervisionado, quando cada vetor do conjunto de entradas é associado a uma resposta e o objetivo é determinar a resposta correta para todos os vetores de entrada, ou não supervisionado, quando apenas o conjunto de entrada é fornecido e devem-se extrair propriedades de acordo com determinadas representações internas (GUIMARÃES e CHAVES NETO, 2006).

Durante o processo de treinamento de uma rede neural são feitas as correções necessárias dos pesos nas conexões, com o objetivo de se estabelecer relações entre as características e classes que consigam fazer da melhor forma possível a identificação entre padrões de classes distintas (RAITTZ, 1997).

> Os neurônios são conectados entre si mediante padrões que definem a arquitetura, ou topologia, da rede. Cada neurônio possui um estado, chamado ativação, que é uma função das entradas que ele recebe. Esta ativação é responsável pelo envio de um sinal para outros neurônios e pode ser binária ou contínua, dependendo da função de ativação utilizada. As funções de ativação mais utilizadas são a Função Passo, ou Degrau Unitário, a Função Linear e a Função Sigmóide (GUIMARÃES e CHAVES NETO, 2006).

Castro (2000) menciona que existem dois modos de treinamento para redes neurais: o Modo Padrão e o Modo de *Batch*. No Modo Padrão os pesos são corrigidos toda vez que a rede receber um novo vetor do conjunto de treinamento e estas correções são feitas apenas no vetor que for apresentado. Desta forma ocorrem N alterações em cada ciclo, onde N representa o número de conjuntos de treino existentes. No Modo de *Batch* é realizada uma única correção em cada ciclo,

pois todos os conjuntos de treinamento são apresentados á rede onde é feito o cálculo do erro médio e com base nesse no resultado deste erro são feitas as correções necessárias dos pesos.

### 3.2.4. Reconhecimento de padrões usando conjuntos difusos

Segundo Raittz (1997), existem muitas maneiras de se aplicar a teoria dos conjuntos difusos em reconhecimento de padrões, mas duas são tidas como formas clássicas de reconhecimento de padrão difuso sendo elas os métodos de reconhecimento através de listas de pertinência o os métodos difusos sintáticos.

> No método clássico da lista de pertinência, cada classe é caracterizada por um conjunto de padrões que é armazenado no sistema de reconhecimento de padrões. Um padrão desconhecido a ser classificado é comparado com os padrões armazenados um a um. O padrão é classificado como um membro de uma classe se ele combinar com um dos padrões pertencentes a esta classe (Raittz, 1997).

> No método sintático difuso, um padrão é representado por uma cadeia de sub-padrões concatenados chamados de primitivas. Estas primitivas são vistas como o alfabeto de uma linguagem formal. Um padrão então é uma sentença gerada por alguma gramática. Todos os padrões, cujas sentenças são geradas pela mesma gramática, pertencem a uma mesma classe. Um padrão desconhecido então, é classificado como pertencente a uma classe particular, se ele puder ser gerado pela gramática correspondente a ela. (KLIR *apud* Raittz, 1997)

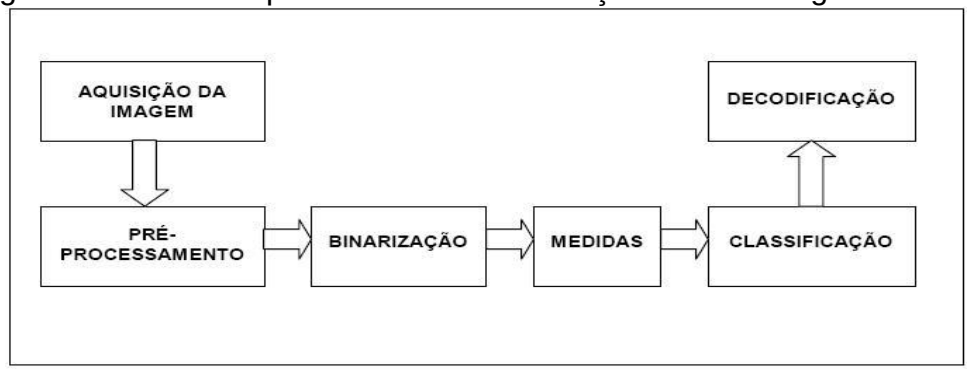
A teoria dos conjuntos difusos permite, de uma forma adequada, modelar situações em que ocorram imprecisões. De acordo com essa teoria, um conjunto não tem limites nitidamente definidos. Um elemento pode pertencer

parcialmente a um conjunto ou pertencer a mais de um conjunto ao mesmo tempo (CASTRO, 2000).

## 3.3. Etapas de um sistema de reconhecimento de padrões

Para atingir o objetivo desejado de reconhecimento de padrões de código de barras do qual trata este trabalho, inicialmente devem ser executados alguns passos, que vão desde a captura da imagem por um dispositivo de captura de imagens até o reconhecimento de um padrão de código de barras propriamente dito. A Figura 10 ilustra as fases de um processo de reconhecimento de padrões de códigos de barra em imagens digitais. Os passos necessários serão melhor detalhados nas próximas sessões.

Figura 10. Fases do processo de decodificação de um código de barras.




Fonte: SOARES, 2001.

A primeira etapa de um sistema de reconhecimento de padrões foi citada no capítulo anterior e corresponde a captura da imagem em um formato digital por algum dispositivo qualquer, capaz de realizar tal tarefa. No caso deste trabalho a imagem será capturada por uma *webcam*. A próxima etapa corresponde ao pré-processamento da imagem capturada.

### 3.3.1. Pré-processamento da imagem

Na etapa correspondente ao pré-processamento são feitas alterações na imagem com o objetivo de realçar características importantes e diminuir o processamento através da remoção de informações desnecessárias. Nesta etapa a imagem geralmente é transformada em tons de cinza e em seguida é segmentada através da binarização da imagem e posteriormente passa pela detecção de bordas e outros ajustes necessários.

> Os principais processos de tratamento de imagens utilizados no reconhecimento de padrões após a aquisição da imagem são a conversão da imagem, caso colorida, em tons de cinza, a binarização, a detecção do contorno, e a segmentação (GONZALES e WOODS *apud* PEREIRA, *et. al,* 2009).

As seções seguintes descrevem algumas etapas de pré-processamento geralmente utilizadas em sistema de reconhecimento de padrões, tendo como entrada uma imagem em tons de cinza.

#### 3.3.1.1. Binarização da imagem

Para o processo de reconhecimento de padrões do qual trata este trabalho não há necessidade de mantermos uma imagem colorida ou em tons de cinza, pois dificultarão o trabalho de localização e identificação do padrão de código de barras que se deseja. Para facilitar o processo, a imagem passará por outro processo chamado binarização.

> Para interpretar uma imagem, as variações nos valores de intensidade devem ser analisadas. Os números de níveis de quantificação usados mais comumente para representar intensidades de imagens são de 256 diferentes níveis de cinza, o que implica um maior esforço computacional e de armazenamento. Estas limitações encorajaram o uso de sistemas de visão binária [...] (FELICIANO, SOUZA e LETA, 2005).

Segundo Rezende (*et. al.* 2007), “O processo de binarização consiste em transformar uma imagem de tons de cinza para somente preto e branco, para daí poder tratar as cores como 0’s e 1’s”. Este procedimento irá facilitar o processo de reconhecimento de padrões e também tornará mais eficiente a detecção de bordas na imagem.

> A binarização se destaca entre os diversos métodos de segmentação de imagens devido ser relativamente simples e de fácil implementação computacional, através de algoritmos rápidos. Por este motivo é o método

> mais utilizado em sistemas de visão automatizados que atuam em aplicações médicas e industriais (ESQUEF, 2002).

Para realizar esta operação "o algoritmo de binarização calcula um valor de limiar da imagem e compara esse valor com os valores dos *pixels* da imagem, transformando em branco os valores maiores que o limiar e, em preto, os valores menores" (NUNES e PRADO, 2002). Desta forma a imagem passará a ser composta apenas por dois *bits*[16] onde a cor branca irá assumir o valor 0 (zero) e a cor preta o valor 1 (um). O resultado deste processo é apresentado na Figura 11, onde à esquerda (A) tem-se a imagem original em tons de cinza e à direita (B) o resultado obtido após a imagem passar pelo processo de binarização.

Figura 11. Resultados obtidos com a binarização.


Fonte: Disponível em: <http://www.drv.ind.br/site/images/stories/visao_maq4.gif>. Acesso em: 20 set. 2009.

Segundo Silva (2004), “a limiarização de imagens é uma das técnicas mais importantes de segmentação, quando direcionada a aplicações de tempo real”. Ainda segundo Silva (2004), a principal dificuldade neste processo é a escolha

[16] *Bit*: Forma abreviada de *binary digit*, corresponde à menor unidade de informação que pode ser manipulada por um computador. Em um *BIT* podemos armazenar um dígito binário - 0 ou 1 no sistema binário de numeração (MEDEIROS, 2000).

correta de um limiar de intensidade que separe da melhor forma possível o fundo do objeto da imagem de forma que ela continue o mais representativa possível.

> Deve ser ressaltado que não existe um modelo formal para a segmentação de imagens. A segmentação é um processo empírico e adaptativo, procurando sempre se adequar às características particulares de cada tipo de imagem e aos objetivos que se pretende alcançar (ESQUEF, 2002).

"De um modo geral, as técnicas de segmentação utilizam duas abordagens principais: a similaridade entre os *pixels* e a descontinuidade entre eles. Sem dúvida, a técnica baseada em similaridade mais utilizada é a chamada binarização" (ESQUEF, 2002).

#### 3.3.1.2. Detecção de bordas na imagem

Para os humanos é fácil observar uma imagem digital e detectar as suas bordas, mas para um sistema computadorizado realizar esta mesma operação não é tão simples assim.

> Uma borda em uma imagem digital pode ser definida como sendo o limite entre duas regiões que possuem níveis de cinza relativamente distintos. Este tipo de abordagem para segmentação pressupõe que as regiões, ou "objetos" da imagem, sejam suficientemente homogêneos em luminância. Isto garante que a transição entre estas regiões possam ser determinadas com base apenas nas descontinuidades entre os níveis de cinza (ESQUEF, 2002).

Conforme Esquef (2002), as bordas de uma imagem digital podem ser detectadas pela mudança brusca das tonalidades de cinza. Existem diversos algoritmos que foram desenvolvidos com o objetivo de realizar a detecção de bordas em imagens digitais, entre eles *Roberts, Sobel e Canny*. Segundo Silva e Alves (2008), estes algoritmos também são conhecidos como filtros[17] por derivadas, pois é o método mais comum de diferenciação utilizado em processamento de imagem.

> Dependendo do fim a que se destina, a detecção de bordas pode ser tida como um fim ou como um pré-processamento para passos subseqüentes. De qualquer forma, para que sejam obtidos os resultados desejados, é necessário que a estratégia de detecção de bordas seja eficiente e confiável (VALE e DAL POZ, 2002).

#### 3.3.1.2.1. *Roberts*

De acordo com Seara (1998), Roberts “é o mais antigo e simples algoritmos de detecção de bordas”.

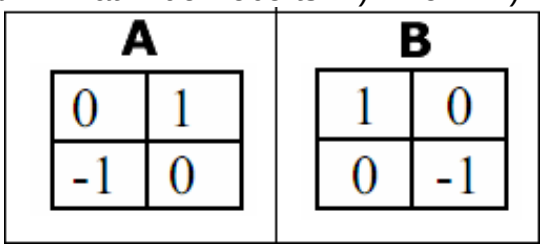
Segundo Bourchardt e Pezzin (2005), o algoritmo de *Roberts* se destaca por ser o mais simples de implementar e por não utilizar muito processamento, pois utiliza uma matriz 2x2, o que também o torna muito sensível a ruídos[18]. A Figura 12 exemplifica a matriz de *Roberts* para os dois eixos da imagem.

---

[17] Filtro: São algoritmos os quais podem ser aplicados as imagens, visando obter determinados efeitos (PARANHOS, 2009).

[18] Ruído: Um ponto preto isolado que possui ao seu redor somente pontos brancos é considerado ruído (BOUCHARDT e PEZZIN, 2005).

Figura 12. Matriz de Roberts. A) Eixo X. B) Eixo Y.




Fonte: PEDROSA e BARCELOS, 2008.

De acordo com Silva e Alves (2008), “a soma de todos os coeficientes da máscara é igual à zero, como também, a soma das diagonais. Já em regiões em que os *pixels* estão situados entre duas fronteiras o resultado do operador é diferente de zero”. Quando isto ocorre o algoritmo de *Roberts* define o ponto como sendo uma borda.

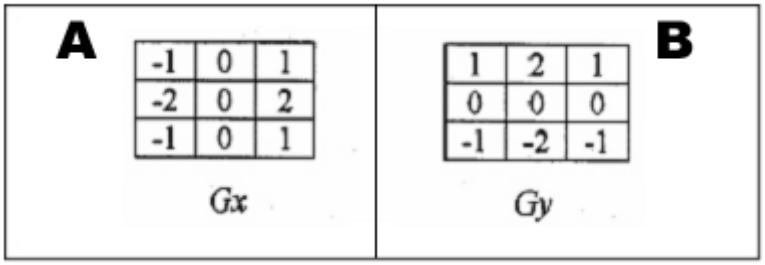
“Uma desvantagem do operador de *Roberts* é a sua anisotropia, ou seja, sua assimetria. Dependendo da direção, certas bordas são mais realçadas que outras, mesmo tendo igual magnitude” (PEDROSA e BARCELOS, 2008).

### 3.3.1.2.2. *Sobel*

Segundo Bourchardt e Pezzin (2005), o algoritmo de *Sobel* é semelhante ao de *Roberts*, a principal diferença é que *Sobel* utiliza uma matriz 3x3, o que o torna mais eficiente em imagens com muitos ruídos, porém necessita de um poder maior de processamento.

Para Silva e Alves (2008), “o detector de bordas de *Sobel* tem como objetivo destacar bordas de uma imagem no sentido horizontal e vertical. Está baseado na aproximação de *Sobel* para a derivada, retornando a borda onde o gradiente é máximo”. Ainda segundo Silva e Alves (2008) “a desvantagem deste método é que o deslocamento da máscara é realizado ou no eixo horizontal (eixo X) ou no eixo vertical (eixo Y), ou seja, só detecta bordas nas duas direções. A Figura 13 ilustra as matrizes de *Sobel* para detecção de bordas no eixo X e Y.

Figura 13. Matriz de Sobel. A) Detecção de bordas no eixo X. B) Detecção de bordas no eixo Y.




Fonte: GONZALES *apud* ERCOLIN FILHO, 2007.

### 3.3.1.2.3. *Canny*

Segundo Silva e Alves (2008), o algoritmo de *Canny* foi desenvolvido por *J. Canny* no ano de 1986 com o objetivo de fazer a detecção de bordas baseado na utilização de um operador gaussiano, sendo apontado por alguns autores como a técnica que apresenta resultados mais consistentes na determinação de contornos de regiões.

> O operador de *Canny* funciona como um processo multi-estágio. Primeiramente, a imagem é suavizada usando convolução *Gaussiana*. Então, uma derivada primeira 2D simples é aplicada à imagem suavizada para destacar regiões da imagem com derivadas primeiras espaciais altas. Arestas dão origem a picos na imagem da magnitude do gradiente. O algoritmo então rastreia ao longo do topo desses picos e seta para zero todos os *pixels* que não estão no topo dos picos, de forma a gerar linhas finas na saída, um processo conhecido como supressão de não-máximos (LIMA, *et. al.* 2008).

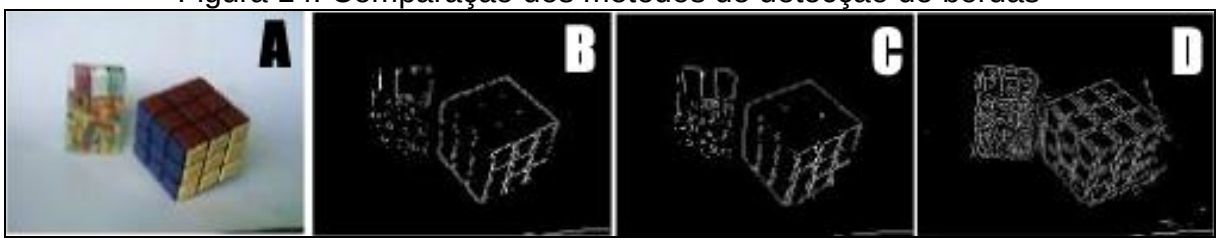
*Canny* utiliza um método conhecido como limiarização por histerese onde os *pixels* que pertencem a uma borda são definidos utilizando dois limiares *TH* e *TL*, sendo um maior e um menor. Primeiramente todos os *pixels* da imagem que possuem um nível de cinza maior que *TH* (limiar maior) são marcados como sendo um *pixel* de borda. Em seguida todos os *pixels* que estiverem conectados aos *pixels* que foram marcados como bordas e que possuam níveis de cinza maiores que *TL* (limiar menor) também passam a ser marcados como *pixels* de borda. Com este método é possível evitar o rompimento das linhas de bordas ocasionadas pela variação dos níveis de cinza (ARTERO e TOMMASELLI, 2009).

Para Bourchardt e Pezzin (2005), o algoritmo de *Canny* tenta diminuir os ruídos da imagem e objetiva localizar todas as bordas existentes na imagem, fazendo a aproximação da borda detectada com a borda original da imagem diminuindo ao máximo o número de *pixels* agrupados para a mesma borda.

> O Operador de *Canny* trabalha com base em 3 critérios ótimos para localização de contornos: mínima probabilidade de detecção de múltiplas bordas; boa localização, isto é, mínima possibilidade de erro na detecção dos pontos pertencentes à borda verdadeira; e detecção de uma única borda, ou seja, se houver duas respostas uma delas é considerada falsa (SILVA e ALVES, 2008).

Existem outros algoritmos que realizam a detecção de bordas em imagens digitais, aqui foram citados apenas os mais conhecidos. A Figura 14 dá um exemplo de cada um dos métodos de detecção de bordas aqui citados aplicados sobre uma imagem digital colorida (A) e o resultado alcançado após ser realizada a detecção de bordas pelos algoritmos de *Roberts* (B), *Sobel* (C) e *Canny* (D).

Figura 14. Comparação dos métodos de detecção de bordas




Fonte: CIRULLO FILHO e LAFUENTE, 2002.

## 3.4. Biblioteca *OpenCV*

Existem diversas bibliotecas que foram desenvolvidas para facilitar a criação de sistemas de reconhecimento de padrões. Neste trabalho será utilizada a biblioteca *OpenCV (Intel Open Source Computer Vision Library)*, por ser considerada a mais eficiente em aplicações de tempo real.

> Reconhecimento de padrões em tempo real é uma tarefa crucial em várias áreas, tais como Realidade Aumentada, Robótica e Interação Homem Computador. A biblioteca *OpenCV* é uma solução open source para visão

> computacional que provê várias funcionalidades adequadas para esse tipo de aplicação (LIMA, *et. al.* 2008).

Segundo Pereira (*et. al.* 2009), a biblioteca *OpenCV* foi desenvolvida pela *Intel* e lançada no ano de 1999 para ser utilizada em sistemas de grande porte como interfaces 3D e aplicações em tempo real. Ainda segundo Pereira (*et. al*., 2009), “A biblioteca *OpenCV* provê uma série de facilidades quanto ao uso de funções algébricas, operações com matrizes, além de acesso facilitado a dispositivos de captura de imagens como *webcams* por exemplo”.

Segundo Lima (*et. al.* 2008), a biblioteca pode ser encontrada para *download* na internet e seu código é aberto para modificações e melhorias desde que sejam respeitadas as cláusulas da licença. Segundo Pereira (*et. al*. 2009), a biblioteca é “livre para uso comercial e de pesquisa sob a licença *BSD*, uma licença de código aberto considerada pouco restritiva quando comparada a outras licenças”.

Segundo Alves (2005), no *OpenCV* as imagens são representadas pela estrutura *IPLIMage*. Através desta estrutura é possível ter acesso a diversos atributos da imagem como largura, altura, máscara de cores, números de canais, entre outros. A biblioteca também possui estruturas dinâmicas para desenhar retângulos, triângulos, círculos, matrizes, grafos, etc.

> São disponibilizadas funções básicas para adicionar, subtrair, multiplicar, dividir, encontrar valor máximo e mínimo, comparar e aplicar operações lógicas entre imagens e matrizes. Há também funções específicas de matrizes e vetores como matriz transposta e inversa, produto escalar e vetorial, autovalores e autovetores de matrizes simétricas e resolver sistemas de equações lineares (ALVES, 2005).

A biblioteca *OpenCV* disponibiliza uma interface gráfica bem simples, permitindo a criação de janelas, *track bars*, carregamento e exibição de imagens e vídeos. Permite o carregamento de imagens nos formatos que são considerados os mais utilizados como *BMP*, *JPG*, *PNG* e *TIF* e arquivos de vídeo no formato *AVI*, podendo ainda salvar as imagens nos formatos citados e também vídeos capturados a partir de uma *webcam* em formato *AVI* (ALVES, 2005).

# CAPÍTULO 4: REVISÃO DE TRABALHOS QUE UTILIZAM *WEBCAM* PARA COLETA DE CÓDIGO DE BARRAS

Neste capítulo serão citados alguns trabalhos desenvolvidos com a finalidade de reconhecer e decodificar algum tipo de padrão de código de barras não necessariamente sendo o padrão *EAN-13*. Todos os sistemas descritos têm em comum a utilização de câmeras como meio de captura das informações.

## 4.1. Trabalhos relacionados

Este não se trata de um trabalho inovador na área de reconhecimento de padrões, pois já existem no meio comercial e também no meio acadêmico sistemas que realizam a mesma tarefa de reconhecer um padrão de código de barras e fazer a decodificação do mesmo utilizando uma *webcam* ou câmeras de celulares como meio de captura de imagens. Serão citados nas próximas sessões quatro sistemas, todos destinados à decodificação de algum tipo de código utilizando câmeras como meio de captura de informações.

## 4.2. *bcWebcam*

O software chamado *bcWebCam*[19] foi desenvolvido pela empresa *QualitySoft*[20] e em sua pagina na internet está disponível uma versão demo do sistema que faz a decodificação de diversos padrões de código de barras, entre eles o *EAN-13*, utilizando uma *webcam* para capturar a imagem do código de barras e exibindo os seus dígitos após a decodificação das barras.

Figura 15. *Print screen* da *interface* do sistema *bcWebCam.*


Fonte: Captura realizada pelo autor do trabalho durante testes realizados dedicados especificamente à pesquisa.

Em testes realizados com o *software*, os fatores que mais pesaram foram o foco da *webcam*, a distancia e a luminosidade do ambiente. Nele os códigos

[19] Disponível em: <http://www.bcwebcam.de/en/index.htm>. Acesso em: 15 ago. 2009.
[20] Disponível em: <http://www.qualitysoft.de/>. Acesso em: 15 ago. 2009.

de barras só puderam ser lidos se o foco da *webcam* fosse ajustado para uma distância aproximada de 10 cm do código de barras. Durante o dia em um local com iluminação natural o *software* funcionou sem maiores problemas, porém, durante testes realizados no período noturno no mesmo local com iluminação artificial indireta, não foi possível à leitura do código de barras em alguns casos.

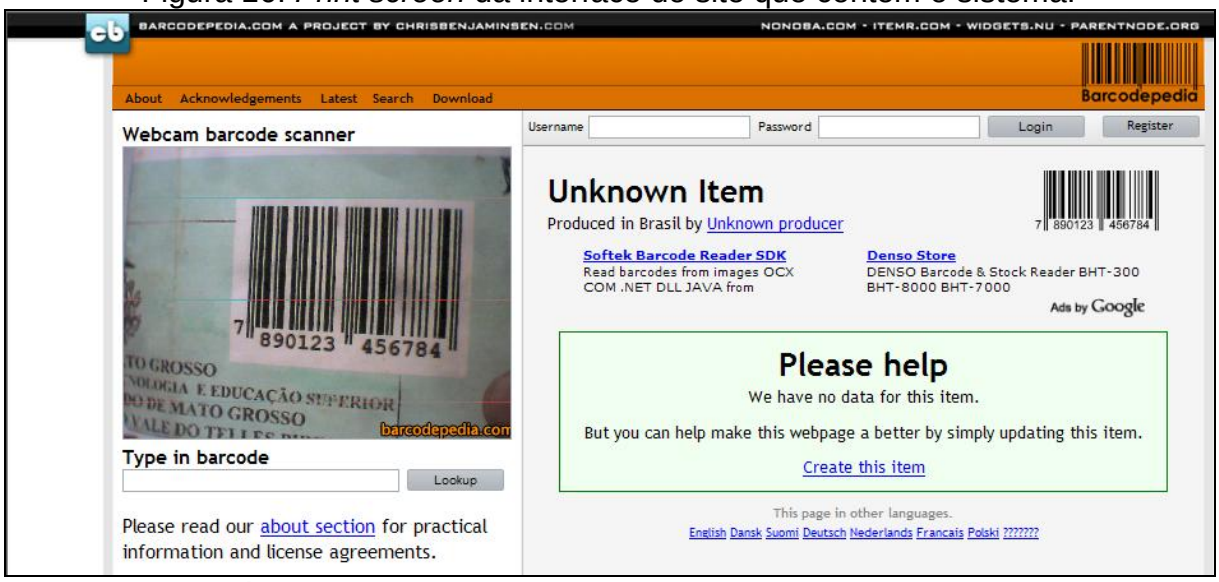
### 4.3. *BarcodePedia*

O sistema que leva o nome de *BarcodePedia*[21] realiza o mesmo processo de decodificação de códigos de barras. O diferencial deste sistema é que ele está incorporado em uma página na Internet, no site da *BarcodePedia*. O sistema foi criado por *Chris Benjaminsen* e trata-se de um aplicativo desenvolvido utilizando a tecnologia *Flash*.

O sistema além de fazer a decodificação de códigos de barras ainda realiza pesquisas em uma base de dados que retorna informação referente ao produto ao qual o código de barras está relacionado. Esse banco de dados é constantemente atualizado pelos próprios usuários que se cadastram gratuitamente no site. Isto significa que nem todos os códigos de barras que forem lidos pelo sistema irão retornar alguma informação associada ao código. O sistema está limitado à decodificação de códigos de barras nos padrões *EAN-13* e *UPC-A*.

[21] Disponível em: <http://en.barcodepedia.com/>. Acesso em. 15 ago. 2009.

Figura 16. *Print screen* da interface do site que contem o sistema.




Fonte: Disponível em: <http://en.barcodepedia.com/>. Acesso em: 25 set. 2009.

## 4.4. *Zebra Barcode Reader*

O sistema de códigos de códigos de *Zebra Barcode Reader*[22] foi desenvolvido por *Jeff Brown* para rodar em sistemas operacionais *Linux*. O interessante deste sistema é que ele é *Open Source*[23], ou seja, é um *software* livre.

> *Zebra Barcode Reader* é uma solução de código aberto que roda em *Linux* capaz de decodificar um código de barras presente no vídeo ao vivo (*WebCam* ou placa de captura), como também processar arquivos de imagem. Este projeto suporta os tipos mais populares de código de barras (FARIA, 2009).

[22] Disponível em: <http://vivaolinux.com.br/artigo/Zebra-Barcode-Reader-Lendo-código-de-barras-com-a-sua-Webcam-no-Linux>. Acesso em: 16 ago. 2009.

[23] *Open source*: Significa código fonte aberto. Programas open source divulgam seus códigos fonte para que qualquer pessoa com conhecimento de programação possa modificá-lo, ampliá-lo e melhorá-lo (COSTA, 2009a).

Segundo Faria (2009), uma imagem gerada por uma *webcam* ou por um *scanner* é enviada para ser processada pelo sistema que por sua vez utiliza técnicas eficazes para fazer a localização e decodificação do código de barras. O processo é semelhante aos leitores tradicionais de código de barras utilizado em supermercados. Está disponível no *You-Tube* um vídeo (disponível em: <http://www.youtube.com/watch?v=5HLVRT4dO6I>. Acesso em: 25 set. 2009) chamado *Utilizando uma webcam para leitura de códigos de barras* que demonstra o funcionamento do sistema.

## 4.5. *i-nigma*

O *i-nigma* é um *software* criado para ser utilizado em celulares com o objetivo de ler *QR Codes* através da câmera do celular. O sistema foi desenvolvido no Japão se tornou um dos mais populares *softwares* do mundo em leitura de *QR Codes*, com cerca de 60 milhões de usuários, sendo distribuído gratuitamente (ALVES, 2009).

> O *QR Code* é um código 2D que, quando capturado por uma câmera de celular através de um *software* apropriado, leva o usuário para um site na internet. Os *QR Codes* são bem conhecidos na Ásia, onde são usados em anúncios impressos e em embalagens de produtos (ALVES, 2009).

Inicialmente o usuário deverá instalar o *software i-nigma* em seu celular, com o sistema instalado basta o usuário capturar uma imagem contendo um *QR Code* através da câmera do celular para que o sistema decodifique o código e extraia as informações nele contidas (TREVISAN, 2009).

> Toda vez que o usuário utiliza o *i-nigma* para ler um *QR Code*, o aplicativo envia essa informação para um servidor ligado ao *software*. Em um banco de dados são guardadas as informações sobre o histórico de acessos de cada usuário e de cada *QR Code* (ALVES, 2009).

Em um *QR Code* podem ser armazenados diversos tipos de informações como endereços, páginas de Internet, *e-mail*, *sms*, preços de produtos, entre outros. A quantidade de informações que pode ser armazenada em um código é variável, podendo chegar a 2.953 *bytes* (TREVISAN, 2009). Está disponível no *You-Tube* um vídeo (disponível em: <http://www.youtube.com/v/ZifQsp5-RFg>. Acesso em: 20 set. 2009) chamado *Aceder à net com a câmara do telemóvel* que demonstra o funcionamento deste sistema.

# CAPÍTULO 5: DETALHAMENTO DO PADRÃO UTILIZADO, EQUIPAMENTOS DE COLETAS DE DADOS E FORMA DE INTERPRETAÇÃO DOS RESULTADOS

Neste capítulo serão descritos os métodos e as ferramentas utilizadas para desenvolver um sistema de reconhecimento de padrões de código de barras *EAN-13* e a decodificação dos dígitos codificados na forma de barras no código.

## 5.1. Equipamentos e ferramentas utilizadas

O sistema de reconhecimento de padrões de código de barras foi desenvolvido em linguagem *C++* com a utilização do compilador *DEV C++*[24] na sua versão 4.9.9.2, rodando no Sistema Operacional *Windows XP (Service Pack 3)* em um computador com processador *Intel Pentium Dual Core* de 2.0 *GHz* com 1 Giga de memória *RAM* e placa de vídeo *GeForce 8500 GT* com 512 *MB*. Como ferramenta auxiliar para o desenvolvimento do sistema foi utilizada a biblioteca *OpenCv 1.1*[25] desenvolvida pela *Intel* e recomendada para aplicações de tempo real.

[24] Disponível em: <http://www.bloodshed.net/devcpp.html>. Acesso em: 18 ago. 2009.
[25] Disponível em: <http://sourceforge.net/projects/opencvlibrary/>. Acesso em: 18 ago. 2009.

Tanto o complicador quando a biblioteca utilizada são disponibilizados gratuitamente para *download* na Internet.

O alvo de pesquisa deste trabalho é o reconhecimento e a decodificação do padrão de código de barras conhecido como *European Article Numbering* (*EAN*). Esse padrão tem duas variações, sendo eles o *EAN-8*[26] e o *EAN-13*. As principais diferenças entre eles estão na sua aparência e na quantidade de caracteres que eles suportam. Neste trabalho será implantado o padrão *EAN-13*, que atualmente é o utilizado no território nacional e um dos mais utilizados em todo o mundo.

Como dispositivo de captura de imagens foi utilizada uma *webcam* da marca Clone[27] com velocidade máxima de captura de imagens de 30 *FPS*, ligada ao computador através de conexão *USB*[28] 1.1. A *webcam* possui ajuste de foco manual e não dispõe de fonte de iluminação própria. Captura imagens coloridas no padrão *RGB* com resolução máxima de 320 x 240 *pixels* e utiliza sensores do tipo *CMOS*.

Figura 17. *Webcam* utilizada na captura das imagens.

Fonte: Figura capturada pelo autor do trabalho, 2009.

[26] *EAN-8*: Utilizado para identificar unidade de consumo, quando a embalagem não tem espaço físico para marcar o *EAN-13* (NEVES, *et. al.* 2004).
[27] Disponível em: <http://www.clone.com.br/db/index02.asp>. Acesso em: 10 mai. 2009.
[28] *USB* - *Universal Serial Bus*: Usada pela maioria dos modelos de câmeras digitais para transferir as imagens para o computador via cabo (PARANHOS, 2009).

## 5.2. Desenvolvimento lógico do sistema

Com base nas etapas de um sistema de reconhecimento de padrões ilustradas na Figura 10, a primeira etapa corresponde a captura da imagem que será encaminhada para processamento por parte do sistema desenvolvido. Uma *webcam* ligada a ao computador será a responsável por capturar a imagem em tempo real e transformá-la em um formato digital para que posteriormente possa ser tratada pelo sistema.

## 5.3. Captura da imagem

Para iniciar a *webcam* e capturar os *frames* foram utilizadas funções da biblioteca *OpenCv* previamente destinadas para este fim. A linha de código a seguir cria um objeto para captura chamado CAPTURA_CAM e a função seguinte informa ao sistema que a fonte de captura será uma câmera previamente instalada no computador.

```
CvCapture* CAPTURA_CAM = cvCaptureFromCAM(CV_CAP_ANY);
```

Com o objeto de captura da imagem criado será possível dar início a captura dos *frames*. A função responsável por esta tarefa é a seguinte:

```
IMG_ENT = cvQueryFrame(CAPTURA_CAM);
```

A função cvQueryFrame faz a captura de um *frame* e armazena esse *frame* na variável IMG_ENT, que guardará a imagem em seu estado original, sem nenhuma alteração por parte do sistema. Em seguida utiliza-se outra função para fazer um clone da imagem de entrada conforme demonstra a linha de código a seguir:

```
IMG_SAI = cvCloneImage(IMG_ENT);
```

A função *cvCloneImage* admite um único parâmetro que corresponde á imagem que deverá ser clonada, no caso IMG_ENT. O clone de IMG_ENT será armazenado em uma nova variável do tipo imagem denominada IMG_SAI e todo o processamento daqui para frente será feito em IMG_SAI, preservando a imagem original.

### 5.3.1. Pré-processamento da imagem capturada

Com a imagem capturada, será dado início à segunda etapa do sistema que corresponde ao pré-processamento, onde serão feitos ajustes na imagem para melhorar sua qualidade e eliminar informações desnecessárias a fim de tornar o processamento mais rápido, facilitar o processo de localização do padrão de código de barras e posteriormente a decodificação do mesmo.

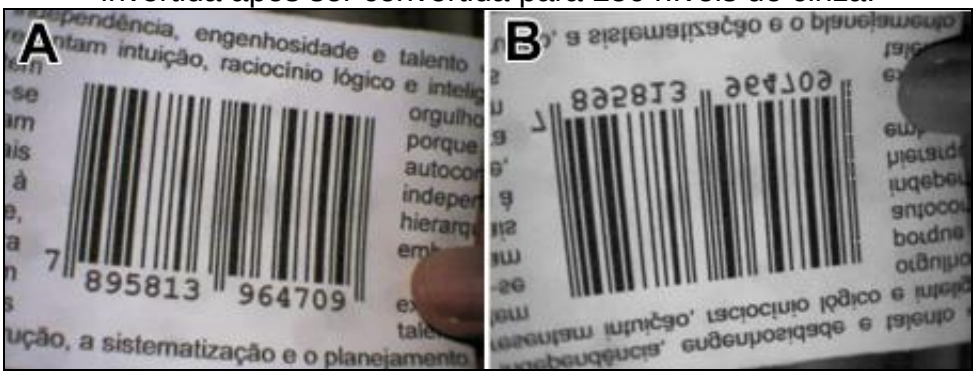
Conforme foi mencionado, a *webcam* utilizada captura uma imagem colorida no padrão *RGB*, com resolução de 320 x 240 *pixels* e, portanto, possui 3 matrizes com 240 linhas e 320 colunas cada uma, onde são armazenados os valores das cores vermelho, verde e azul de cada *pixel*, o que equivale a 3 matrizes contendo 76.800 registros cada uma, totalizando 230.400 registros.

Como neste sistema não há necessidade de utilizar imagens em cores, a primeira parte do pré-processamento será converter a imagem coloria em uma imagem em tons de cinza, formada por uma única matriz de *pixel* com 240 linhas e 320 colunas, totalizando 76.800 registros, o que representa uma considerável diminuição de informações desnecessárias. A biblioteca *OpenCv* já possui funções prontas para realizar esse tipo de conversão conforme pode ser visto a seguir:

```
cvCvtColor( IMG_ENT, IMG_SAI, CV_RGB2GRAY )
```

A função *cvCvtColor* é compota por 3 campos, no primeiro deve ser informado a imagem de origem, que no caso é IMG_ENT no padrão *RGB*, no segundo campo será o local onde será armazenada a imagem em tons de cinza, no caso IMG_SAI, e o último campo corresponde ao tipo de conversão de cores que deverá ser feito na imagem, que no caso será *RGB2GRAY* (*RGB* para *Gray)*. Porém, ao fazer a conversão da cor, a imagem de saída (IMG_SAI) passa a ficar de cabeça para baixo conforme demonstra a Figura 18.

Figura 18. Imagens obtidas a partir da *webcam*. A) Imagem original. B) Imagem invertida após ser convertida para 256 níveis de cinza.




Fonte: Captura de tela realizada pelo autor do trabalho, 2009.

O problema que pode ser visto na Figura 18 ocorre pelo seguinte motivo:

> No OpenCV as imagens de três canais de cor (RGB) são carregadas na memória com as componentes de cor de cada pixel na ordem invertida, ou seja, na ordem BGR. Também é convencionado que coordenada (0,0) seja o pixel situado no canto inferior esquerdo da imagem. Em várias aplicações essas convenções mudam, e o ponto (0,0) pode ser admitido no canto superior esquerdo da imagem. A função cvConvertImage permite tanto modificar a ordem das componentes de cor (RGB/BGR) quanto o referencial (0,0) da imagem (FERREIRA e MORAES, 2007).

Para corrigir este problema descrito acima utilizou-se a função *cvConvertImage* conforme demonstra a linha de código a seguir.

```
cvConvertImage(IMG_ENT, IMG_SAI, CV_CVTIMG_FLIP);
```

A função *cvConvertImage* admite 3 campos. No primeiro deve ser informado a imagem colorida em eu estado original (IMG_ENT), no segundo campo deve ser informado a imagem que deverá ser modificada, no caso será a própria imagem que foi convertida em níveis de cinza (IMG_SAI) e no terceiro campo é informado o tipo de conversão que deverá ser feito em IMG_SAI. São duas as possibilidades: *CV_CVTIMG_SWAP_RB* e *CV_CVTIMG_FLIP*. A primeira opção é utilizada quando queremos corrigir a ordem da seqüência de cores de *BGR* para *RGB*, porém, como não iremos trabalhar com imagens em cores, este parâmetro não será utilizado. A segunda opção inverte o verticalmente o referencial da imagem e, desta forma, é feito um giro na imagem fazendo com que ela retorne para o modo correto.

Terminada a primeira etapa do pré-processamento é dado início á segunda etapa, desta vez para transformar a imagem que se encontra em tons de cinza para uma nova imagem com apenas 2 tons de cinza, formando uma imagem binária, ou seja, uma imagem constituída por apenas duas cores, preto e branco.

Na *OpenCv* existem algumas funções destinadas à binarização de imagens como, por exemplo, as funções *CvAdaptativeThreshold* e *cvThreshold*. No caso da primeira função não é necessário informar um valor de limiar, pois o cálculo é feito automaticamente. Porém optou por não utilizar as funções e sim criar uma seqüência de repetição encadeada, sendo uma para o eixo X e outra para o eixo Y da imagem, onde ao mesmo tempo em que a imagem for sendo binarizada será formada uma matriz binária chamada MAT_BIN, que será utilizada nas etapas seguintes para localizar seqüências de padrões do código de barras. Para ter acesso ao valor dos níveis intensidade dos *pixels* foi utilizada a função abaixo:

```
COR = cvGet2D(IMG_SAI,X,Y);
```

A função cvGet2D é composta por 3 campos onde no primeiro campo é informada a imagem da qual se deseja extrair o valor de intensidade dos *pixels*, os outros dois parâmetros serão as coordenadas X e Y da imagem. Também foi criada uma variável do tipo *Cvscalar* a qual se deu o nome de COR, que servirá para armazenar o valor do *pixel,* que posteriormente será comparado com o valor do limiar que decidirá se o pixel na posição X,Y terá sua intensidade alterada para 255 ou para 0 .

A escolha correta de um limiar é sempre uma tarefa complicada, pois depende muito da iluminação do ambiente que modifica constantemente. Mesmo se utilizando uma fonte de iluminação estável podem ocorrer problemas, pois um limiar pode funcionar adequadamente para binarizar um código de barras contido em uma

imagem capturada pela *webcam*, já em outro pode não funcionar corretamente. Para tentar diminuir esses problemas foi criado um intervalo para o limiar, variando de 80 a 110. O limiar se inicia em 80 e a cada volta do *looping* o limiar é acrescido até chegar em 110, onde a partir daí ele começa a decrescer até se chegar a 80 novamente. Desta forma o código pode não ser binarizado satisfatoriamente com o valor de limiar inicial, mas posteriormente poderá ser satisfatório com algum dos valores do intervalo.

O limiar será o ponto de divisão das cores, será feita uma comparação do valor armazenado na variável COR com o valor da variável LIMIAR, onde o valor de um *pixel* que tiver o valor de intensidade igual ou acima do limiar terá o seu valor de intensidade modificado para 255 (cor branca) e o valor do *pixel* que estiver abaixo do limiar terá seu valor de intensidade modificados para 0 (cor preta). A Figura 19 ilustra uma imagem de um código de barras binarizada pelo sistema.

Figura 19. Resultados do pré processamento. A) I imagem original. B) Imagem com 256 níveis de cinza. C) Imagem binarizada.


Fonte: Capturas de tela realizadas pelo autor do trabalho, 2009.

Esta mesma lógica será utilizada para formar a matriz binária, pois quando for igual ou acima do limiar será armazenado o número zero na posição X,Y

da matriz MAT_BIN e quando estiver abaixo será armazenado o valor 1. Esta comparação será feita individualmente em todos os *pixels* existentes na imagem, ficando armazenado na matriz MAT_BIN a representação binária de toda a imagem que poderá ser utilizada para encontrar as seqüências binárias de padrões existentes no código de barras como as barras de início e fim que definirão o local do código de barras e posteriormente as seqüências que representam os dígitos do código de barras.

### 5.3.2. Localização do código de barras

Após a imagem ser binarizada e a matriz binária ser montada já é possível fazer a localização do código de barras existente na imagem. A localização será feita através dos padrões de início e fim do código. Conforme foi mencionado no Capítulo 1, por padrão as barras que delimitam o início e o fim de um código de barras *EAN-13* são formadas pela seqüência binária 101. Porém somente esta seqüência não é suficiente para localizar as barras, pois este mesmo padrão pode ser encontrado facilmente em outras partes do código e também no restante da imagem. Como todos os módulos de cada dígito do lado esquerdo se iniciam com o número 0 e todos os módulos dos dígitos do lado direito terminam com o número 0, o padrão de início para o sistema poderiam ser 1010 e o de fim 0101, mas mesmo assim este padrão é facilmente encontrado em outras partes do código. Então optou-se por juntar esta seqüência com as margens de silêncio (espaços em branco encontrados tanto do lado esquerdo quando do lado direito do código) existentes por

padrão nos códigos *EAN-13,* ficando então o padrão de início (lado esquerdo) a seqüência 00001010 e o padrão de fim (lado direito) a seqüência 01010000.

Para localizar as barras de início e fim foi criada uma nova seqüência de repetição encadeada para cada eixo da imagem (X e Y), de modo que a imagem toda seja percorrida *pixel* a *pixel*. A busca pelos padrões foi feita através de comparações dos valores dos *pixels* no eixo horizontal da imagem da seguinte forma:

```
If (MAT_BIN[X][Y]==0 and MAT_BIN[X][Y+1]==0 and
MAT_BIN[X][Y+2]==0 and MAT_BIN[X][Y+3]==0 and
MAT_BIN[X][Y+4]==1 and MAT_BIN[X][Y+5]==0 and
MAT_BIN[X][Y+6]==1 and MAT_BIN[X][Y+7]==0)
```

A seqüência de código acima corresponde a busca pelo padrão de início do código de barras (00001010). Para encontrar as barras de fim, o processo é o mesmo, mudando apenas os valores de comparação. Essa busca é feita em todos os *pixels* da imagem a fim de fazer a localização exata do código de barras, tendo sua altura e largura.

O ponto inicial do código será marcado através da linha e da coluna em que for encontrada a primeira seqüência de padrões referente às barras de início do código (00001010) e serão armazenadas nas variáveis PX1 (linha inicial) e PY1 (coluna inicial). O processo para encontrar o ponto final do código é semelhante, a diferença é que a linha e a coluna serão marcadas somente quando for encontrada última seqüência de padrões referentes às barras de fim do código (01010000) e serão armazenadas nas variáveis PX2 (linha final) e PY2 (coluna final). Com estes

quatro pontos encontrados é possível ter a localização do código de barras. Estes valores serão utilizados na etapa seguinte para isolar, do restante da imagem, somente a área de interesse.

### 5.3.3. Definição da região de interesse.

Quando o código de barras for encontrado através dos padrões de início e fim citados na sessão anterior, o restante da imagem deve ser ignorado e o processamento deverá ser feito somente na área de interesse, ou seja, no local onde o código de barras está localizado. Para realizar este processo foi utilizada a função *cvSetImageRoi*.

> Em várias aplicações é desejável processar somente uma determinada área, uma região de interesse (Region Of Interest - ROI) da imagem. O OpenCV permite que um ROI seja definido facilmente. Quando uma imagem é criada, na estrutura de dados IplImage existe um campo “roi” que é inicialmente definido como nulo (NULL). Ao utilizarmos qualquer função de processamento de imagem do OpenCV, a função verifica se existe um ROI definido na estrutura de dados da imagem, e caso exista um ROI o processamento é feito somente dentro da região definida pelo ROI (FERREIRA e MORAES, 2007).

Esta função recebe dois parâmetros, o primeiro é uma variável do tipo imagem onde a região de interesse será ativada e o segundo parâmetro é uma estrutura do tipo retângulo que admite 4 valores, que servirão para marcar a área de interesse. Os valores serão os seguintes: linha inicial (PX1), coluna inicial (PY1),

altura (PX2) e largura (PY2). Todos encontrados durante a busca pelos padrões de início e fim do código de barras citados na seção anterior.

Quando o *ROI* for ativado na imagem, será formado um retângulo ao redor do código de barras localizado, onde todo o processamento será concentrado. Qualquer outro ponto da imagem que estiver fora deste retângulo será automaticamente descartado pelo sistema até que a *ROI* seja desabilitada através da função *cvResetImageROI* da seguinte maneira:

```
cvResetImageROI(IMG_SAI);
```

A função *cvResetImageROI* admite um único parâmetro que corresponde a imagem na qual a *ROI* foi ativada e, com isso, o sistema passará a "enxergar" toda a imagem novamente.

### 5.3.4. Separação e decodificação das seqüências binárias de padrões

Com o padrão de código de barras localizado e devidamente destacado do restante da imagem, inicia-se a etapa para decodificação dos dígitos codificados na forma de barras no código. Para isso foi criado um arquivo do tipo texto chamado ARQ_EAN_AUX, que armazena a seqüência binária de toda a área que foi marcada como sendo um código de barras.

Na Tabela 10 é exibido parte do conteúdo de um arquivo, contendo 10 linhas e 20 colunas, geradas a partir de um código de barras contendo defeitos, tanto nas barras quanto nos espaços. A tabela é formada por seqüências de números zeros e uns caracterizando uma tabela binária, onde cada número zero da tabela corresponde a cor branca (barras claras) do código e cada número 1 corresponde a cor preta (barras escuras) do código.

Tabela 10. Exemplo de parte de uma seqüência binária armazenada em arquivo.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |

Fonte: CONTE e PEZZIN, 2005.

Observando o exemplo é possível verificar que, nas áreas que não contém defeitos, as colunas começam e terminam com a mesma seqüência binária e em áreas onde existem falhas no código a seqüência é desigual. Este arquivo será utilizado para fazer o somatório das colunas e estipular através deste somatório, o que deve ser realmente considerado como barra clara ou barra escura. Isto é necessário, pois a imagem capturada do código geralmente não é de boa qualidade, pode possuir falhas geradas durante a binarização, ou ainda o código pode estar um

pouco danificado, contendo falhas ou manchas que poderiam passar informações erradas ocasionando uma decodificação incorreta das barras.

O método utilizado para definir o que será considerado barra escura ou barra clara é baseado no método definido por Conte e Pezzin (2009), que diz o seguinte:

> Quando o somatório da 1ª coluna, por exemplo, for zero é porque não existe linha, todas as partes mostraram isso representando através do número “0”, se for 10, é por que existe linha, todas as 10 partes mostram isso, representando através do número “1”. No entanto se o somatório resultar em 9, também é uma linha, possui alguma falha, mas é linha, pois a maioria das partes indicou isso através do número “1”. Se o resultado for 2 então não é linha, tem um pouco de sujeira mas não é linha, pois a maioria das partes resultou em “0” (CONTE e PEZZIN, 2005).

Utilizando a Tabela 11 como exemplo, se o somatório de uma coluna for maior do que a metade da quantidade de linhas existentes no arquivo então será uma barra escura e se estiver abaixo da metade da quantidade de linhas será uma barra clara. Optou-se por utilizar este método, pois a altura do código pode variar devido a defeitos no código e, portanto, desta forma independente de quantas linhas o arquivo contiver, será possível estipular quais colunas serão consideradas barras e quais não serão.

Este processo irá gerar um novo arquivo de texto chamado ARQ_EAN, desta vez contendo uma única linha que será utilizada para fazer a decodificação e revelar os dígitos do código de barras. É interessante ressaltar que este processo não irá tornar o sistema capaz de solucionar qualquer problema existente no código de barras, mas o tornará mais eficiente, pois ao invés de se utilizar uma única linha para decodificar todos os padrões se utiliza a média de várias linhas.

Tabela 11. Exemplo do somatório das colunas de um código de barras *EAN-13*.

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| **SOMATÓRIO DAS COLUNAS** | | | | | | | | | | | | | | | | | | | |
| **2** | **10** | **0** | **0** | **10** | **10** | **0** | **0** | **9** | **3** | **0** | **10** | **8** | **3** | **0** | **5** | **0** | **9** | **8** | **0** |

Fonte: CONTE e PEZZIN, 2005.

O passo seguinte faz a leitura do arquivo criado (ARQ_EAN) e armazena o seu conteúdo em uma variável chamada STRING_EAN. Em seguida essa *string* é dividida em várias sub-strings, separando os módulos do código de barras em variáveis diferentes, que serão comparadas com todos os padrões de dígitos presentes no código *EAN-13.* Na comparação serão descartados os padrões que não representam dígitos como as margens de silêncio, os padrões de início e fim e os padrões de guarda central.

Cada dígito do padrão *EAN-13* é formado por 7 módulos, portando foi criado uma variável chamada DÍGITO, para armazenar 12 seqüências binárias com 7 dígitos cada uma. A seqüência de código a seguir cria as sub-strings e armazena cada uma em uma posição da variável DIGITO.

```
DIGITO[1]   = STRING_EAN.substr(3, 7);
DIGITO[2]   = STRING_EAN.substr(10, 7);
DIGITO[3]   = STRING_EAN.substr(17, 7);
DIGITO[4]   = STRING_EAN.substr(24, 7);
```

```
DIGITO[5]   = STRING_EAN.substr(31, 7);
DIGITO[6]   = STRING_EAN.substr(38, 7);
DIGITO[7]   = STRING_EAN.substr(50, 7);
DIGITO[8]   = STRING_EAN.substr(57, 7);
DIGITO[9]   = STRING_EAN.substr(64, 7);
DIGITO[10] = STRING_EAN.substr(71, 7);
DIGITO[11] = STRING_EAN.substr(78, 7);
DIGITO[12] = STRING_EAN.substr(85, 7);
```

Analisando a seqüência de código acima é possível observar que a primeira *substring* está sendo formada a partir da posição 3, isto ocorre porque a seqüência referente às barras de início que contém 3 módulos que foram ignorados pelo sistema. O mesmo ocorre entre os índices 6 e 7 da variável DÍGITO, onde ao invés da *substring* do índice 7 iniciar na posição 45, ela passou a ser iniciada na posição 50 devido à seqüência corresponde as barras de guarda central do código que possuem 5 módulos e que também foram ignoradas pelo sistema.

Após separar todas as seqüências é feita uma comparação com todos os padrões presentes nas Tabelas A, B e C de padrões, e à medida que as seqüências forem sendo decodificadas é formada uma matriz chamada COD_EAN_FINAL com 13 posições onde serão armazenados os números que forem sendo decodificados. Também foi criada uma segunda matriz chamada PARIDADE, que armazenará a seqüência de paridade dos 6 primeiros dígitos do lado esquerdo do código presentes na Tabela A (paridade impar) e na Tabela B (paridade par) que definirão o primeiro dígito do código.

No sistema, a seqüência das paridades será representada pelos números 0 e 1, onde o zero representa a paridade impar (Tabela A) e o um a paridade par (Tabela B). Esta seqüência binária de paridades será comparada com os padrões da tabela auxiliar de paridades (ver Tabela 1) para revelar o primeiro

dígito do código de barras. A titulo de exemplo será utilizado o código representado na Figura 20, sendo o sentido de leitura da esquerda para direita.

Figura 20. Exemplo de código de barras *EAN-13*.


Fonte: FORD, 2002

Tabela 12. Paridade dos 6 primeiros dígitos do lado esquerdo.

| Posição | Dígito | Módulos | Tabela |
|---|---|---|---|
| 2 | 5 | 0110001 | A |
| 3 | 0 | 0100111 | B |
| 4 | 1 | 0011001 | A |
| 5 | 0 | 0100111 | B |
| 6 | 3 | 0111101 | A |
| 7 | 1 | 0110011 | B |

Fonte: Desenvolvida pelo autor do trabalho como incremento da pesquisa, 2009.

Conforme pode ser visto na Tabela 12, as paridades dos 6 primeiros dígitos do lado esquerdo do código de barras formam a seqüência ABABAB, que corresponde ao dígito 7 da tabela auxiliar de paridades (Tabela 1). Este número será armazenado na posição 0 da variável COD_EAN_FINAL e será utilizado juntamente com os demais dígitos para calcular o dígito verificador que definirá se realmente o código decodificado é valido ou não.

### 5.3.5. Cálculo do dígito verificador

A partir de agora todos os dígitos do código de barras já foram decodificados, mas ainda deve ser verificado se o código de barras foi decodificado corretamente ou não, pois alguns dígitos podem ser decodificados de forma incorreta. O que garante que um código de barras foi decodificado corretamente é o dígito verificador, que é gerado através de cálculos com os 12 primeiros dígitos do código. Quando o cálculo for feito, o resultado deverá ser comparado com o ultimo dígito do código (que é o dígito verificador). Se o resultado do cálculo coincidir com o último dígito armazenado na posição 12 da variável COD_EAN_FINAL, o código será considerado válido, do contrário será considerado inválido.

O processo necessário para efetuar o cálculo do dígito verificador foi implementado no sistema da forma como foi descrito na Seção 1.5 do Capítulo 1, seguindo exatamente os mesmos passos.

Uma variável chamada CONDICAO será a responsável por reiniciar o sistema quando o código de barras for validado. A variável é iniciada com o valor 0 (zero) que corresponde a um código de barras inválido. Quando através do cálculo do dígito verificador o resultado apontar para um código válido será atribuído o valor 1 (um) a variável. Quando esta condição for atingida o sistema deverá ser reiniciado.

## 5.4. Funcionamento do sistema

Quando o sistema é iniciado ele entra em um *looping* infinito paralelo, onde não é realizada nenhuma operação a não ser a exibição da imagem que está sendo capturada pela *webcam*. Neste ponto o sistema fica aguardando que o usuário pressione uma tecla para dar início a localização e decodificação do código de barras. Como opção foi escolhida a tecla correspondente a lera “i” ou “l”.

Quando a tecla for pressionada, o sistema entra em um segundo *looping* infinito onde será feito o pré-processamento da imagem, a localização do código de barras, a separação e decodificação das seqüências de padrões, o cálculo do dígito verificador e a validação ou invalidação do código. Portanto, quando a tecla for pressionada o usuário deverá colocar o código de barras em frente a *webcam* a uma distância aproximada de 10 cm da forma mais alinhada possível para uma decodificação mais eficiente por parte do sistema.

Nesta etapa, o sistema ficará procurando pelos padrões de correspondentes as barras de inicio e fim do código de barras a fim de localizar toda a área onde ele se encontra. À medida que as barras forem sendo encontradas pelo sistema elas serão marcadas com a cor branca e a área do código será envolvida por um retângulo branco. Desta forma o usuário saberá que o código está posto na distância correta em relação a *webcam*.

A cada volta do *loopin* os arquivos gerados durante o processo serão eliminados e as matrizes esvaziadas. O critério de parada será a validação de um código de barras e, quando isto ocorrer, o sistema será reiniciado, as variáveis de imagens serão esvaziadas, a memória será esvaziada e o sistema retornará para o

*loopin* paralelo onde novamente ficará aguardando uma tecla ser pressionada para recomeçar a operação. A Figura 21 ilustra a interface do sistema onde a esquerda tem-se a imagem de entrada e a direita a imagem após ser processada pelo sistema.

O código fonte do sistema desenvolvido foi disponibilizado para download no site de compartilhamento de arquivos *4Shared* (disponível em: <http://www.4shared.com/file/141872190/c5d9f695/DECODIFICADOR_DE_CDIGO_DE_BARRAS_EAN-13__V_012c_BETA_.html>. Acesso em: 19 out. 2009).

Figura 21. Interface do sistema.


Fonte: Captura de tela realizada pelo autor do trabalho, 2009.

# CAPÍTULO 6: TESTES REALIZADOS E AVALIAÇÃO DOS RESULTADOS

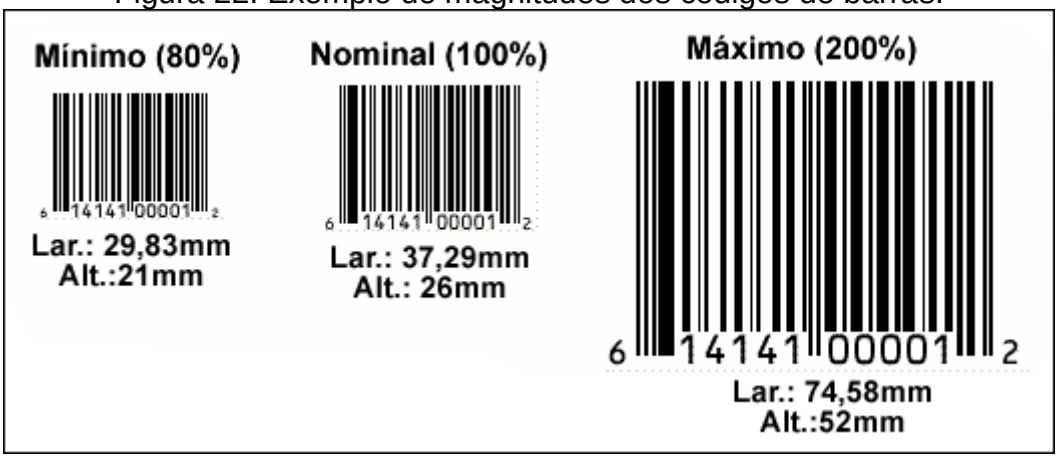
Este capítulo irá descrever os métodos utilizados durante os testes realizados no sistema bem como as principais dificuldades encontradas, as limitações, os problemas encontrados, as possíveis causas e as possíveis soluções para os problemas. Todos os testes descritos neste capítulo foram realizados no mesmo computador onde o sistema foi desenvolvido e também foi utilizada a mesma *webcam* citados no capítulo anterior.

## 6.1. Testes preliminares

Nos testes preliminares foram testados dois tamanhos de códigos de barra diferentes, sendo um em tamanho nominal e outro em tamanho máximo. O objetivo era analisar qual dos dois tamanhos se sairia melhor. Nos testes preliminares os códigos que mostraram uma facilidade maior de decodificação foram os códigos em tamanho máximo, pois se mostraram mais toleráveis a pequenas variações de distância do que os códigos em tamanho nominal. A Figura 22 dá um

exemplo da magnitude[29] dos códigos de barras, porém, estão sendo ilustrados somente para análise de magnitude, pois não se tratam de códigos de barras padrão *EAN-13* e sim *UPC*[30] que contém apenas 12 dígitos.

Figura 22. Exemplo de magnitudes dos códigos de barras.




Fonte: RIBINIK, 2009c.

## 6.2. Definição dos métodos utilizados nos testes.

Para realização dos testes foram impressos 6 códigos de barras com numerações diferentes, com os quais serão simuladas situações que podem ocorrer quando se utiliza códigos de barras. Nos testes, foram simulados os seguintes problemas: Manchas na superfície do código, código amassado, código riscado e código impresso com qualidade ruim. Além destes, foi um impresso um código normal, sem defeitos que também será utilizado nos testes do sistema. Todos os

[29] Tamanho.
[30] *Universal Product Code* (SOARES, 2001).

códigos de barras foram impressos em uma impressora a jato de tinta na melhor configuração possível, com exceção do código que simulará má qualidade de impressão, que foi impresso em modo rascunho rápido.

## 6.3. Resultados obtidos nos testes

Nas próximas seções serão descritos os resultados obtidos nos testes realizados com cada um dos códigos de barras impressos, serão descritas as principais dificuldades encontradas em cada um dos problemas que serão simulados e também os resultados obtidos nos testes realizados com os códigos de barras que não possuíam defeitos.

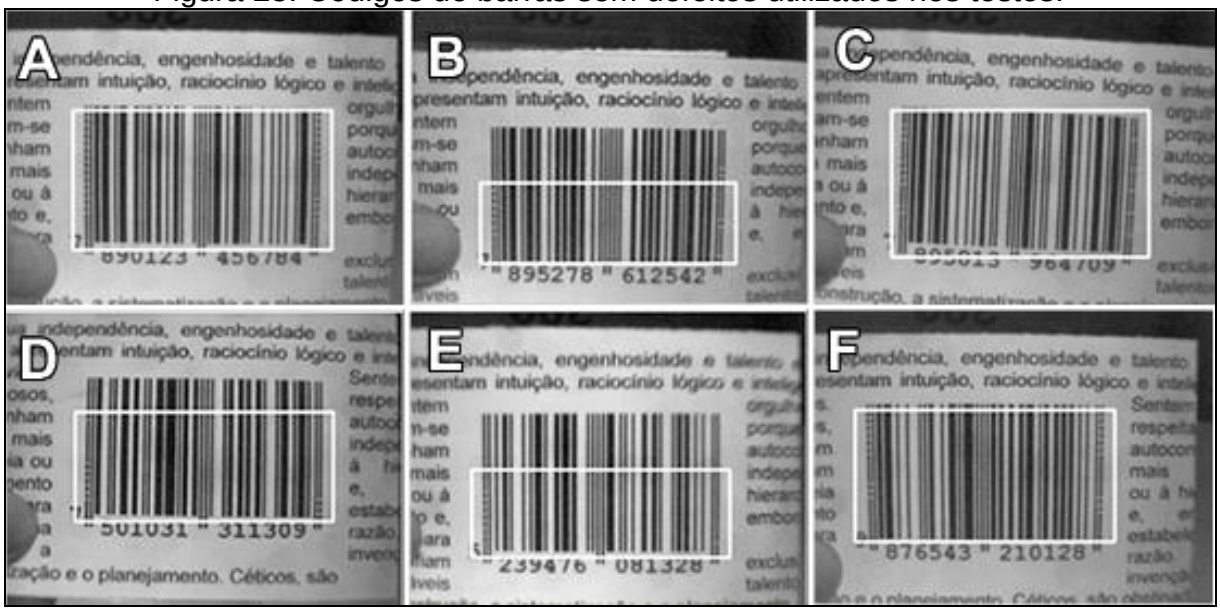
Em todos os testes, o tempo gasto para decodificação de um código leva em consideração que no momento em que o sistema for iniciado o código de barras ainda deverá ser posto frente à *webcam* na posição correta para que o sistema possa fazer a decodificação do código.

### 6.3.1. Resultados dos testes com códigos sem defeitos

Os códigos de barras considerados como sem defeitos foram os códigos impressos na melhor qualidade de impressão que a impressora disponibilizava. Estes códigos não contêm nenhum defeito na sua estrutura como

manchas, riscos ou amassados. A Figura 23 ilustra os códigos de barras sem defeitos utilizados nos testes do sistema.

Figura 23. Códigos de barras sem defeitos utilizados nos testes.




Fonte: Capturas de tela realizadas pelo autor do trabalho, 2009.

Pode-se perceber na Figura 23 que, em alguns casos, mesmo o código estando um pouco desalinhado é possível a sua decodificação. Porém, quando mais desalinhado o código de barras estiver, menores serão as chances de uma decodificação eficiente.

Nos testes realizados com códigos de barras em bom estado, mesmo com pequenas variações de luminosidade, todos puderam ser decodificados com mesmo valor de limiar (limiar = 100). O tempo mínimo necessário para decodificação de um código foi de 3 segundos e o tempo máximo foi de 6 segundos. O tempo médio para decodificação de um código foi de 4,50 segundos com desvio padrão de 1,37 segundos.

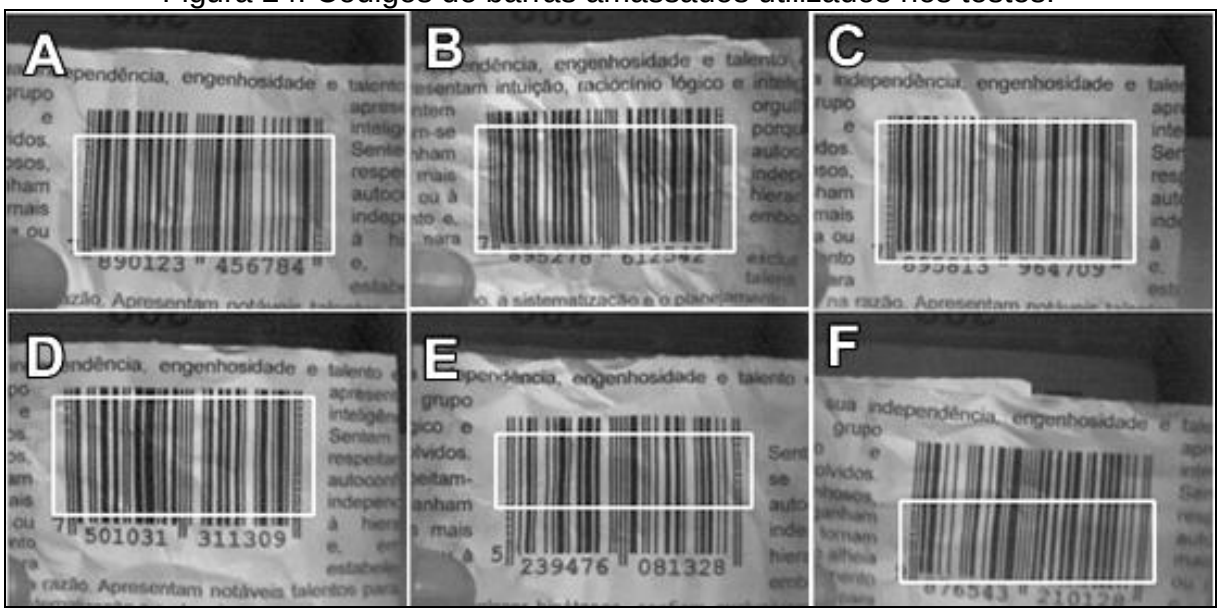
Foi postado no site *You-Tube* um vídeo (disponível em: <http://www.youtube.com/watch?v=v8__o4DZq94>. Acesso em: 05 out. 2009) chamado *Testes realizados com códigos de barras normais* que demonstra os testes realizados.

### 6.3.2. Resultados dos testes com códigos amassados

Neste teste foram simulados defeitos de códigos de barras amassados. Primeiramente os 6 códigos foram impressos em boa qualidade e, em seguida, todos eles foram amassados ficando com a forma de uma bola de papel. Quando o código de barras é amassado surgem problemas como barras tortas, sombras, diminuição da espessura das barras e espaços e desnivelamento da superfície do código. A Figura 24 ilustra os códigos de barras amassados utilizados nos testes, onde ficam evidentes os efeitos causados quando se amassa um código de barras. É possível notar que as barras passam a ficar onduladas devido ao desnivelamento da superfície do código, causando sombras em alguns locais, o que dificulta o pré-processamento da imagem, podendo tornar o código impossível de ser decodificado.

Nos testes realizados com os códigos amassados não houve a necessidade de realizar alterações no valor do limiar e em todos os casos os códigos foram decodificados com o mesmo valor de limiar (limiar = 100). O tempo mínimo necessário para decodificar um código amassado foi de 4 segundos e o tempo máximo foi de 9 segundos. O tempo médio para decodificação de um código foi de 5,17 segundos com desvio padrão de 2,04 segundos.

Figura 24. Códigos de barras amassados utilizados nos testes.




Fonte: Capturas de tela realizadas pelo autor do trabalho, 2009

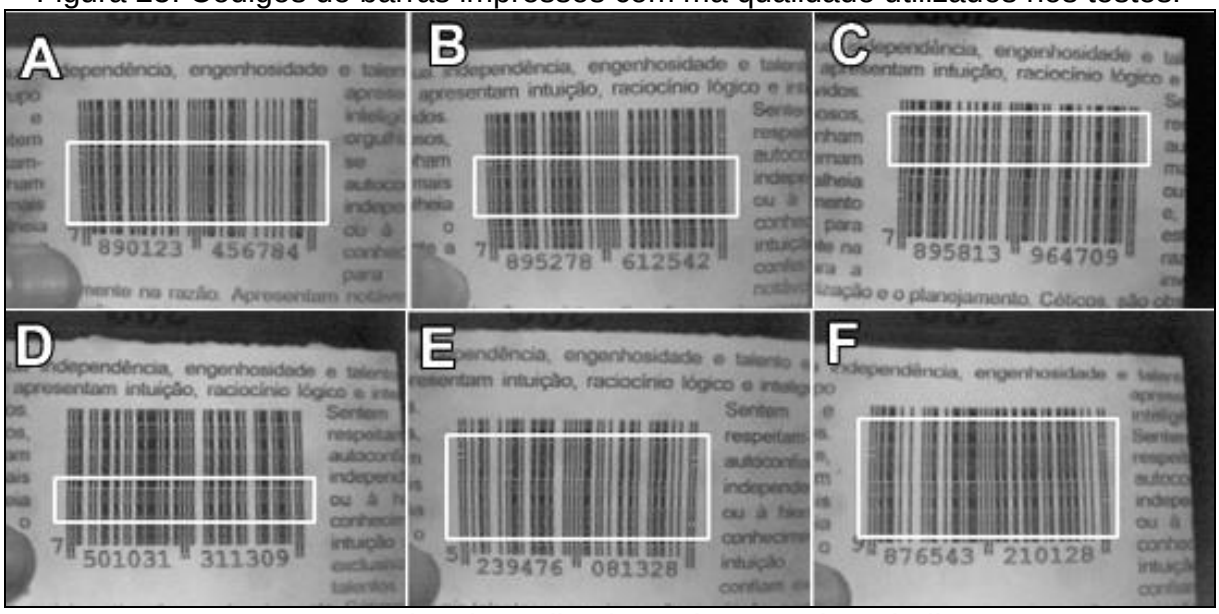
Foi postado no site *You-Tube* um vídeo (disponível em: <http://www.youtube.com/watch?v=D1vEuVpNZ-U>. Acesso em: 05 out. 2009) chamado *Testes realizados com códigos de barras amassados* que demonstra os testes realizados.

### 6.3.3. Resultados dos testes realizados com códigos apresentando má qualidade de impressão

Os códigos utilizados neste teste foram impressos na pior qualidade de impressão que a impressora disponibilizava (modo rascunho rápido) e foram utilizados para simular os efeitos causados por códigos impressos com má

qualidade. É possível analisar através da Figura 25, que os códigos de barras contém falhas de impressão nas barras, formando trechos mais claros que outros No geral as barras do código ficam relativamente cinza ao invés da cor preta. Outro defeito notado foi o borramento da tinta, fazendo com que as barras tivessem sua largura ligeiramente aumentada.

Figura 25. Códigos de barras impressos com má qualidade utilizados nos testes.




Fonte: Capturas de tela realizadas pelo autor do trabalho, 2009

Para que o sistema fosse capaz de decodificar os códigos de barras impressos com má qualidade houve a necessidade de aumentar o valor do limiar padrão de 100 para 108. Essa necessidade se deu porque as barras do código possuíam uma cor mais clara do que os códigos que foram impressos com boa qualidade. Nos testes realizados com os códigos impressos com má qualidade, o tempo mínimo necessário para a decodificação de um código foi de 3 segundos e o

tempo máximo foi de 17 segundos. O tempo médio para decodificação de um código foi de 8,50 segundos com desvio padrão de 5,28 segundos.

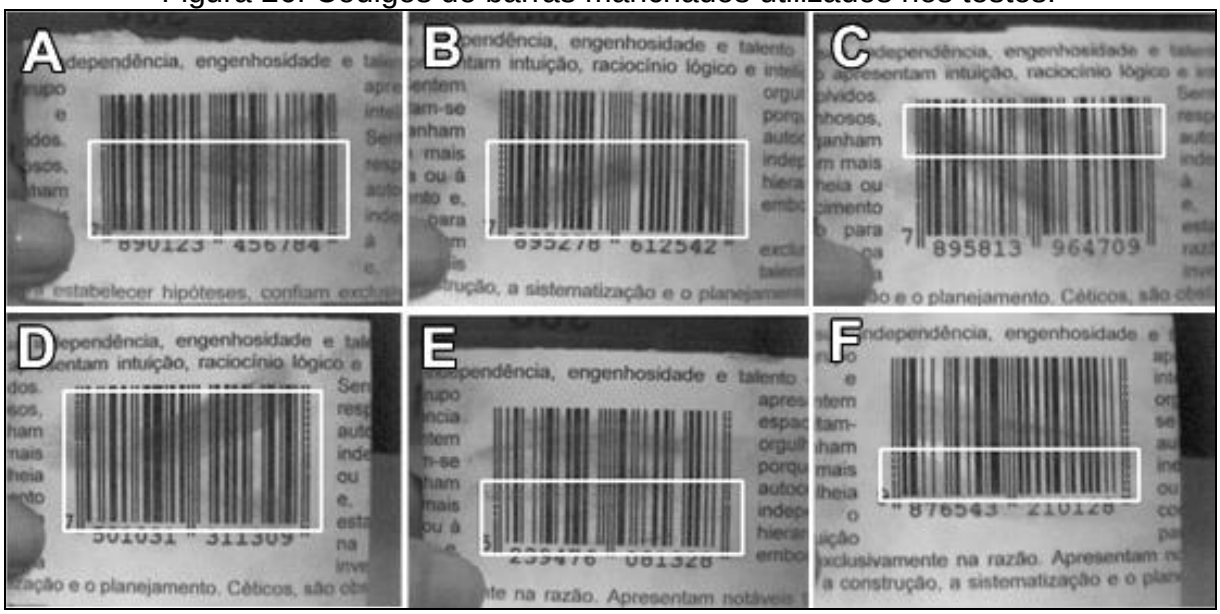
Foi postado no site *You-Tube* um vídeo (disponível em: <http://www.youtube.com/watch?v=b-jrOIWXTBU>. Acesso em: 05 out. 2009) chamado *Testes realizados com códigos de barras apresentando má qualidade* que demonstra os testes realizados.

### 6.3.4. Resultados dos testes realizados com códigos manchados

Este teste simula defeitos nos códigos de barras causados por manchas. Primeiramente os códigos de barras foram impressos em boa qualidade e em seguida, enquanto a tinta ainda não estava completamente seca, foi pressionando o dedo sobre as barras impressas, causando o borramento da tinta e gerando as manchas conforme pode ser analisado na Figura 26.

Com o efeito de borramento podem ocorrer dois principais problemas. O primeiro problema corresponde ao aumento da largura das barras em algumas regiões. O segundo problema ocorre nas barras brancas do código, onde dependendo da intensidade do borramento, as barras claras do código passam a ter uma coloração semelhante a das barras pretas, podendo gerar problemas na etapa de binarização, pois, dependendo do nível do borramento, as barras claras poderão se unir com as barras escuras formando assim, em alguns locais, uma única barra preta larga.

Figura 26. Códigos de barras manchados utilizados nos testes.




Fonte: Capturas de tela realizadas pelo autor do trabalho, 2009.

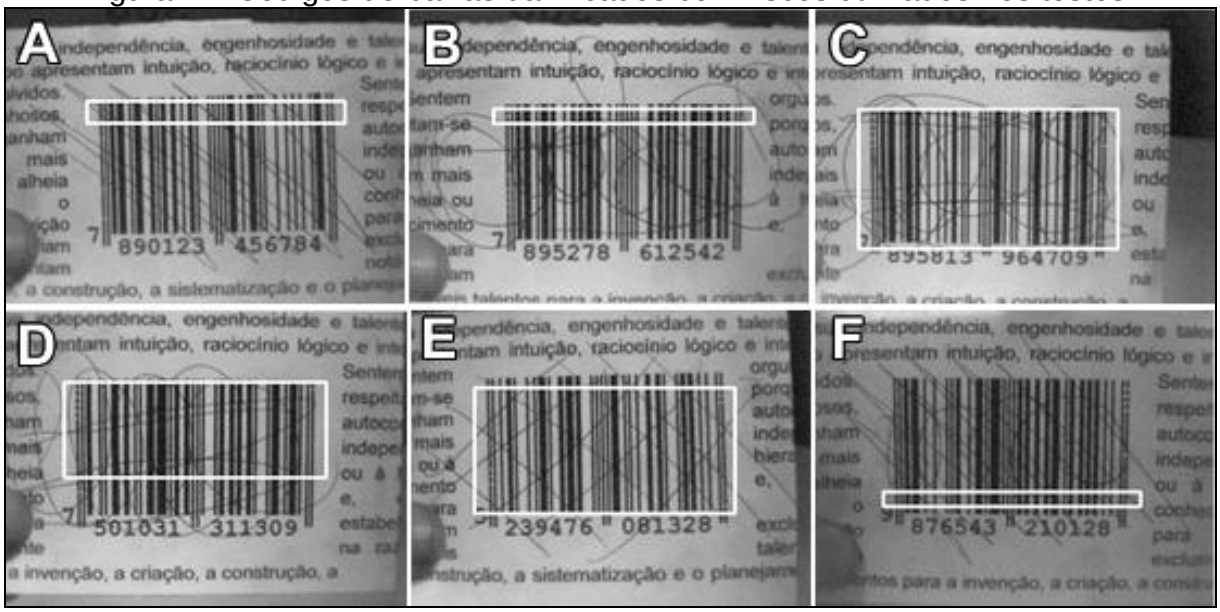
Todos os códigos de barras utilizados neste teste puderam ser decodificados sem a necessidade de alterar o valor do limiar definido como padrão no sistema (limiar = 100). Nos testes realizados com códigos de barras manchados, o tempo mínimo para decodificação de um código foi de 3 segundos e o tempo máximo foi de 13 segundos. A média de tempo foi de 6,17 segundos com desvio padrão de 3,81 segundos.

Foi postado no site *You-Tube* um vídeo (disponível em: <http://www.youtube.com/watch?v=HpqMA_YxJks>. Acesso em: 05 out. 2009) chamado *Testes realizados com códigos de barras manchados* que demonstra os testes realizados.

### 6.3.5. Resultados dos testes realizados com códigos riscados

Para realização dos testes desta categoria, primeiramente todos os códigos foram impressos em boa qualidade e em seguida foram riscados de diversas formas com a utilização de uma caneta esferográfica comum de cor azul. A Figura 27 ilustra as imagens de códigos de barras riscados que foram utilizados nos testes.

Figura 27. Códigos de barras danificados com riscos utilizados nos testes.




Fonte: Capturas de tela realizadas pelo autor do trabalho, 2009

Todos os códigos utilizados neste teste puderam ser decodificados com o valor definido como padrão no sistema (limiar = 100). Nos testes realizados com os códigos de barras riscados, o tempo mínimo para decodificação de um código foi de 3 segundos e o tempo máximo foi de 7 segundos. A média de tempo

para decodificação de um código foi de 4,67 segundos com desvio padrão de 1,63 segundos.

Foi postado no site *You-Tube* um vídeo (disponível em: <http://www.youtube.com/watch?v=kjvFydXvzdg>. Acesso em: 05 out. 2009) chamado *Testes realizados com códigos de barras riscados* que demonstra os testes realizados.

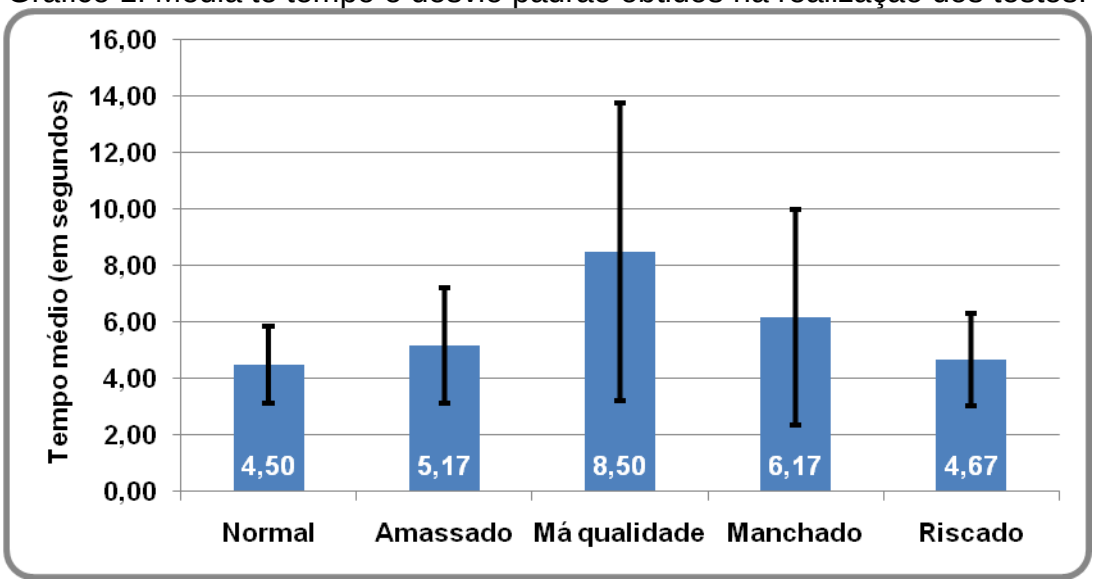
## 6.4. Avaliação dos resultados

Com a realização dos testes o sistema mostrou-se eficiente na decodificação de códigos de barras em bom estado e também em códigos de barras apresentando defeitos causados por amassados, manchas, riscos ou por má qualidade durante o processo de impressão do código. O Gráfico 1 apresenta o resultado da média de tempo necessário para decodificação dos códigos em cada um dos testes realizados no sistema acompanhado do desvio padrão.

Durante os testes o que ocorria constantemente eram códigos inválidos serem dados como válidos pelo sistema. Isto ocorria porque quando uma seqüência binária referente a algum dígito não era compatível com nenhum dos padrões do código *EAN-13*, automaticamente era atribuído o valor zero na posição correspondente da variável COD_EAN_FINAL, e posteriormente no cálculo do dígito verificador, dependendo da combinação dos números, o resultado do cálculo resultava em um dígito verificador que era igual ao ultimo dígito do código e desta forma o código era validado mesmo se tratando de um código de barras

decodificado erroneamente. Para resolver este problema, quando uma seqüência de padrões não for encontrada será armazenado o valor -1 (um negativo) na variável COD_EAN_FINAL e logo depois que todas as seqüência forem decodificadas (ou não) será verificado se alguma posição da variável possui algum valor negativo, caso encontre, mesmo que o dígito verificador aponte para um código válido ele será dado como inválido pelo sistema.

Gráfico 1. Média te tempo e desvio padrão obtidos na realização dos testes.




Fonte: Desenvolvido pelo autor do trabalho mediante resultados obtidos nos testes realizados no sistema desenvolvido, 2009.

Outro problema encontrado durante os testes foi referente ao limiar, pois a técnica utilizada para definir um intervalo de valores para o limiar não se mostrou eficiente durante a realização dos testes. A solução encontrada para solucionar este problema foi criar *track-bar*, onde o limiar deve ser escolhido de forma manual pelo usuário, pois não foi implementado no sistema recursos para definir automaticamente um valor de limiar com base no histograma da imagem.

Devido ao fato da *webcam* possuir um ajuste automático de iluminação houve a necessidade de deixar um fundo branco, pois do contrário, se houvesse um fundo com variações de luminosidade ou escuro, a *webcam* tentaria corrigir a iluminação e gastaria em média 5 segundos para realizar esta tarefa e somente após este tempo o sistema seria capaz de realizar a decodificação do código. Ao colocar um fundo branco, a *webcam* faz o ajuste rapidamente no momento que o sistema for iniciado e posteriormente permanece com a mesma configuração para o ajuste de iluminação.

## CONSIDERAÇÕES FINAIS

Este trabalho apresentou o protótipo de um sistema de reconhecimento e decodificação de códigos de barras padrão *EAN-13*. Para se chegar ao resultado esperado foram realizadas diversas etapas que foram desde a aquisição da imagem pela *webcam*, passando pelas etapas de pré-processamento da imagem como a binarização até se chagar na localização e decodificação das seqüências de padrões que se encontram codificados na forma de barras no código.

Como ferramenta de auxílio no processo de desenvolvimento do sistema foi utilizada a biblioteca *OpenCv*, recomendada para aplicações que envolvam visão computacional e reconhecimento de padrões em tempo real. A biblioteca *OpenCV* mostrou-se satisfatória, atendendo aos requisitos de velocidade e eficiência necessários para sistemas que trabalham em tempo real, pois disponibiliza ótimos recursos para se trabalhar com reconhecimento de padrões e muitos desses recursos podem ser utilizados através da escrita de apenas uma linha de código.

Sistemas de reconhecimento de padrões em imagens estão sendo muito utilizados atualmente, realizando as mais diversas tarefas. Reconhecimento de faces, reconhecimento de caracteres, e o reconhecimento e decodificação de códigos de barras são exemplos desta tecnologia. O sistema desenvolvido neste

trabalho tem a capacidade de encontrar e decodificar o padrão de código de barras *EAN-13* em qualquer local da imagem capturada pela *webcam*, porém, com a utilização das técnicas citadas neste trabalho, o sistema mostrou-se muito sensível as variações de distância entre o código de barras e a *webcam*, também mostrou sensibilidade a variações de inclinação do código de barras, embora estes problemas já fossem previstos, uma vez que não foram implementados no sistema.

Desta forma fica evidente que o ponto mais crítico do sistema é com relação a distância e alinhamento do código de barras em relação a *webcam,* o que representa alguns problemas, pois, para os códigos serem decodificados eles precisam ser colocados em frente a *webcam* da forma mais alinhada possível e em uma distância correta. O intervalo de distancia tolerável para leitura durante os testes se mostrou muito pequeno. O sistema não disponibiliza recursos para calcular a largura mínima dos módulos, pois desta forma poderiam ser feito ajustes na imagem a fim de deixá-la sempre em um tamanho padrão, evitando assim problemas ocasionados por pequenas variações de distância. Portanto pode se concluir que o principal problema notado no sistema desenvolvido corresponde às etapas de pré-processamento da imagem.

Muitas vezes o código de barras era localizado através dos padrões correspondentes as barras de inicio e fim, mesmo com pequenas variações de distância, mas na maioria dos casos em que o código era localizado o sistema não conseguia fazer a decodificação das barras, seja por defeitos ocasionados durante a binarização devido a escolha de um limiar incorreto ou com relação a distância entre o código de barras e a *webcam*, onde as barras do código se tornavam mais finas ou muito largas impossibilitando a decodificação.

## TRABALHOS FUTUROS

Como o principal problema do sistema foi referente às etapas de pré-processamento da imagem, portanto, as principais propostas são para a melhoria do sistema desenvolvido onde serão incluídas funcionalidades que não foram implementadas no sistema desenvolvido.

A primeira proposta é a melhoria do pré-processamento da imagem, onde podem ser incluídos recursos que atualmente não foram implementados como a correção do alinhamento do código e a estimativa de distância para atenuar os problemas ocasionados com a distância entre a *webcam* e o código de barras.

A segunda proposta também corresponde ao pré-processamento da imagem onde se pretende implementar o cálculo automático do limiar, o que iria resolver problemas ocasionados com variações de iluminação, melhorando assim a qualidade da binarização da imagem, tornando o sistema mais eficiente pois não necessitaria que o limiar fosse escolhido de forma manual da maneira que foi utilizado no sistema desenvolvido.

A terceira proposta é desenvolver um banco de dados para armazenar os códigos de barras que forem decodificados ou realizar consultas por códigos já cadastrados. O objetivo principal do sistema é a decodificação de códigos de barras

que estariam na capa de processos da Unemat com o intuito de fazer a protocolização dos processos de forma eletrônica, onde o código de barras seria o número de protocolo. Além do protocolo, poderiam ser armazenados no banco de dados informações como data de protocolo, folhas, partes interessadas, assunto do processo, juntada, destino e data de envio, entre outras informações.

A quarta proposta é desenvolver uma interface para o sistema com a capacidade de realizar operações com banco de dados como cadastramento, consulta e alterações dos registros, impressão de relatórios, etc. No sistema o campo referente ao protocolo poderia ser preenchido de duas formas. Na primeira o código seria preenchido automaticamente mediante a decodificação do código de barras pelo sistema desenvolvido e em um segundo caso, o campo de protocolo seria preenchido manualmente quando o sistema não fosse capaz de decodificar o código de barras.

**REFERÊNCIAS BIBLIOGRÁFICAS**


ALCOBIA, P. **Porque é que os scanners de códigos de barras CCD, avariam menos que os de laser?** Disponível em: <http://st.equi-libra.net/?p=23>. Acesso em 01 ago. 2009.

ALVES, G. T. M. **Um estudo das técnicas de obtenção de forma a partir de estéreo e luz estruturada para engenharia**. Dissertação (mestrado). Pontifícia Universidade Católica do Rio de Janeiro - PUC - Rio. Rio de Janeiro, 2005. Disponível em: <http://www2.dbd.puc-rio.br/pergamum/tesesabertas/0310818_05_postextual.pdf>. Acesso em: 28 ago. 2009.

ALVES, W. ***Mobile marketing*****:** *QR codes* no Brasil. Disponível em: <http://www.comunidade.cn/icox.php?mdl=mensagem&op=ver&idcom=101&id=58356>. Acesso em: 17 jul. 2009.

ARTERO, A. O.; TOMASELLI, A. M. G. **Detecção e afinamento de bordas em direções previamente conhecidas**. Bol. Ciênc. Geod., sec. Artigos, Curitiba, v. 15, n. 2, abr-jun, 2009. Disponível em: <http://ojs.c3sl.ufpr.br/ojs2/index.php/bcg/article/viewFile/14604/9802>. Acesso em: 11 set. 2009.

AYRES, M. **Entenda o funcionamento de uma filmadora digital**. Disponível em: <http://tecnologia.uol.com.br/produtos/ultnot/2007/03/09/ult2880u323.jhtm>. Acesso em: 27 abr. 2009.

BOGOMOLNY, A. ***Barcode encoding***. Disponível em: <http://www.cut-the-knot.org/do_you_know/BarcodeEncoding.shtml#EAN13>. Acesso em: 05 out. 2009.

BOURCHARDT, E. F.; PEZZIN, M. Z. **Identificação de veículos utilizando técnicas de visão computacional**. Universidade do Contestado. Concórdia, 2005. Disponível em: <http://www.dcc.unesc.net/sulcomp/05/Art080SulComp2005.pdf>. Acesso em: 07 set. 2008.

CAMPOS, T. E. **Técnicas de seleção de características com aplicações em reconhecimento de faces**. Dissertação (mestrado). Universidade de São Paulo - USP. São Paulo, 2001. Disponível em: <http://www.vision.ime.usp.br/%7Eteo/publications/dissertacao/node12.html>. Acesso em: 24 jul. 2009.

CAPPELARO, E. A.; *et. al.* **Você sabe o que é uma câmera digital?**. Disponível em: <http://educar.sc.usp.br/licenciatura/trabalhos/camera.htm>. Acesso em: 27 abr. 2009.

CARDOSO*, et. al*. **Glossário.** Disponível em: <http://www.esb.ucp.pt/twt4/motor/display_texto.asp?pagina=glossario200409143814634&bd=sinf>. Acesso em: 15 set. 2009.

CARVALHO, C. A.; MENEGUETE JUNIOR, M.; SILVA, E. A. **A influência do espaço de cores na compressão *JPEG* de imagens orbitais**. Rev. Brasileira de Cartografia, n. 55/01. Disponível em: <http://www.rbc.ufrj.br/_pdf_55_2003/55_1_05.pdf>. Acesso em: 18 ago. 2009.

CASTRO, A. A. M.; PRADO, P. P. L. **Algoritmos para reconhecimento de padrões**. Universidade de Taubaté. Ciências Exatas, Taubaté, v. 5-8, 1999-2002. Disponível em: <http://www.agro.unitau.br/exatas/ojs/include/getdoc.php?id=441&article=110&mode=pdf>. Acesso em: 25 mar. 2009.

CASTRO, E. B. P. **Automação da produção**. Universidade Federal de Juiz de Fora - UFJF. Juiz de Fora, 2007. Disponível em: <http://www.engprod.ufjf.br/epd_automacao/EPD030_CodBarras.pdf>. Acesso em: 25 abr. 2009.

CASTRO, K. A. **Implementação de um sistema de reconhecimento de padrões difusos de imagens de satélite usando redes neurais**. Relatório final de iniciação científica. Universidade Federal do Mato Grosso do Sul. Campo Grande, 2000. Disponível em: <http://www.engprod.ufjf.br/epd_automacao/EPD030_CodBarras.pdf>. Acesso em: 20 jul. 2009.

CIRULLO FILHO, O.; LAFUENTE, J. F. V. **Visão computacional:** métodos de detecção de bordas. Disponível em: <http://www.lti.pcs.usp.br/vc/turma2002/grupo5/pcs_5726.htm>. Acesso em: 22 set. 2008.

CLAUDIO, C. S. FACHINI, G.; RAMOS, V. G. **Laser na medicina**. Disponível em: <http://www.cdcc.sc.usp.br/julianoneto/laser/laser.html#introducao>. Acesso em: 28 set. 2009.

CONTE, L. A.; PEZZIN, M. Z. **Reconhecimento de crachás utilizando técnicas de visão computacional e lógica *fuzzy***. Universidade do Contestado. Concórdia, 2005. Disponível em: <http://www.dcc.unesc.net/sulcomp/05/Art079SulComp2005.pdf>. Acesso em: 01 set. 2008.

COSTA, C. **Como funciona a *web* 2.0:** glossário da web 2.0. Disponível em: <http://informatica.hsw.uol.com.br/web-206.htm>. Acesso em: 10 set. 2009a.

COSTA, W. S. **A GS1 Brasil**. Disponível em: <http://www.ean.org.br/main.jsp?lumChannelId=28040E9A165411DB9E2BDB753E7F9C5C>. Acesso em: 25 jun. 2009b.

ERCOLIN FILHO, L. **Extração semi-automática de feições lineares e a calibração dos parâmetros intrínsecos de câmeras**. Universidade de São Paulo – USP. São Paulo, 2007. Disponível em: <http://www.lps.usp.br/~hae/psi5796-05/seminarios/LeonardoErcolin-artigo.pdf>. Acesso em: 17 ago. 2009.

ESQUEF, I. A. **Técnicas de entropia em processamento de imagens**. Tese (mestrado). Centro Brasileiro de Pesquisas Físicas. Rio de Janeiro, 2002. Disponível em: <http://cbpfindex.cbpf.br/publication_pdfs/TESE_vfinal.2006_05_07_11_04_41.pdf>. Acesso em: 10 set. 2008.

FALCÃO, A. X.; LEITE, N. J. **Fundamentos de processamento de imagem digital**. Disponível em: <http://www.dcc.unicamp.br/~cpg/material-didatico/mo815/9802/curso.pdf.gz>. Acesso em: 18 ago. 2009.

FARIA, A. O. ***Zebra Barcode Reader*:** lendo código de barras com a sua *webcam* no *Linux*. Disponível em: <http://www.vivaolinux.com.br/artigo/Zebra-Barcode-Reader-Lendo-código-de-barras-com-a-sua-Webcam-no-Linux/>. Acesso em: 15 abr. 2009.

FELICIANO, F. F.; SOUZA, I. S.; LETA, F. R. **Visão computacional aplicada a metrologia dimensional automatizada:** considerações sobre sua exatidão. ENGEVISTA, v. 7, n. 2, dez. 2005. Disponível em: <http://www.uff.br/engevista/2_7Engevista04.pdf>. Acesso em: 10 set. 2009.

FERNANDES, J. **Projeto de uma arquitetura genérica de dispositivos de controle de acesso**. Relatório de projeto de conclusão de curso. Universidade Regional Integrado do Alto Uruguai e das Missões - URI. Santo Ângelo, 2001. Disponível em: <http://www.urisan.tche.br/~informatica/PCJocelito.doc>. Acesso em: 25 abr. 2009.

FERREIRA, M.; MORAES, A. **Tutorial *OpenCv*:** manipulação de imagens. TecGraf/PUC-Rio. Rio de Janeiro, 2007. Disponível em: <http://www.tecgraf.puc-rio.br/~malf/opencv/index_files/manip_img5.htm>. Acesso em: 19 ago. 2009.

FLORENCE, R. **Vantagens competitivas no varejo parte 1:** a revolução do RFID. Disponível em: <http://www.dealmaker.com.br/artigos/57_artigos_vantagens.htm>. Acesso em: 10 set. 2009.

FORD, R. ***Barcode Maker***. Disponível em: <http://ranfo.com/software/barcode.zip>. Acesso em: 5 out. 2009.

FREIRE, A.; GRILO, T. **Leitores de códigos de barras**. Disponível em: <http://www.bibliosoft.pt/html/suporte_tecnologias_lcb.htm>. Acesso em 27 abr. 2009.

FREIRE, L. **Código de barras**. Disponível em: <http://www.luizfreire.com/producao/logistica/barcode.php>. Acesso em: 17 set. 2008.

FUNGHI, E. **Vocabulário gráfico**. Disponível em: <http://funghit.multiply.com/journal/item/6/6>. Acesso em: 10 set. 2009.

GÓES, J. A.; SIQUEIRA, C. S. **Sistema de rastreamento de movimento urbano na região metropolitana de Belém**. Dissertação (Bacharelato em Ciências da Computação). Universidade da Amazônia. Belém, 2005. Disponível em: <http://www.cci.unama.br/margalho/portaltcc/tcc2005/PDF/007.pdf>. Acesso em: 26 ago. 2008.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais.** São Paulo: Blücher, 2005.

GUIMARÃES, I. A.; CHAVES NETO, A. **Reconhecimento de padrões:** comparação de métodos multivariados e redes neurais. Negócios e Tecnologia da Informação. Curitiba, 2006. Disponível em: <http://publica.fesppr.br/index.php/rnti/article/view/v1n1ART5/89>. Acesso em: 24 jul. 2009.

LEITE, E. **Captura de imagem digital**. Disponível em: <http://www.focusfoto.com.br/captura.image.digital.htm>. Acesso em: 27 abr. 2009.

LIMA, J. P. M.; et*. al*. **Reconhecimento de padrões de tempo real usando a biblioteca *OpenCV***. Minicurso V WRVA. Bauru, 2008. Disponível em: <https://150.161.192.17/svn/repositorioGPRT/2008/Public/WRVA/JoaoPauloLima_ReconhecimentoPadroesTempoReal.pdf>. Acesso em: 28 ago. 2009.

LUDWIG JUNIOR, O. **Análise comparativa entre métodos de reconhecimento de padrões aplicados ao problema gás-lift intermitente**. Dissertação (mestrado). Universidade Federal da Bahia - UFBA. Salvador, 2004. Disponível em: <http://www.bibliotecadigital.ufba.br/tde_busca/arquivo.php?codArquivo=199>. Acesso em: 17 ago. 2009.

MARCONI, L. **Veja como funcionam as câmeras fotográficas**. Disponível em: <http://focusfoto.com.br/fotografia-digital/index.php/2008/10/18/veja-como-funcioonam-as-cameras-fotograf>. Acesso em: 25 set. 2009.

MATIAS, T.; RODRIGUES, J. **Dicionário de informática**. Disponível em: <http://grupo3ap.googlepages.com/dicion%C3%A1rio>. Acesso em: 10 mai. 2009.

MEDEIROS, C. L. G. **Solução de problemas usando a linguagem Pascal**. Disponível em: <http://www.dsc.ufcg.edu.br/~camilo/cursos/2000.2/icc/03CAP1.DOC>. Acesso em: 10 mai. 2009.

MELLO, A. **Código de barras**. Disponível em: <http://www.imasters.com.br/artigo.php?cn=1793&cc=13>. Acesso em: 08 set. 2008.

MINAS, G. M. H. **Conversor luz-freqüência integrados num chip *CMOS***. Disponível em: <http://alibaba.dei.uminho.pt/~gminas/MaterialAulas/Trabalho_ConvLuzFreq.pdf>. Acesso em: 10 set. 2009.

NASCIMENTO, M. C. **Detecção de objetos em imagens**. Trabalho de Graduação. Universidade Federal de Pernambuco. Recife, 2007. Disponível em: <http://www.cin.ufpe.br/~tg/2007-2/mcn2.doc>. Acesso em: 14 jul. 2009.

NEVES, I. L. *et. al.* ***EAN 13*****:** tecnologia da informação. Fundação de Estudos Sociais do Paraná. Curitiba, 2004. Disponível em: <http://www.fesppr.br/~erico/x%202004%20Trabalhos%20s.210/210%20EAN-13.doc>. Acesso em: 10 ago. 2009.

NICIOLI, D. N. **Câmeras digitais**. Disponível em: <http://www.eletronicosforum.com/cursos/camera_digital/Camera_digital.pdf>. Acesso em: 27 abr. 2009.

NUNES, L. E. N. P.; PRADO, P. P. L. **Reconhecimento de objetos contidos em imagens através de redes neurais**. Universidade de Taubaté. Ciências Exatas. Taubaté, v. 5-8, 1999-2002. Disponível em: <http://www.agro.unitau.br/exatas/ojs/include/getdoc.php?id=425&article=106&mode=pdf>. Acesso em: 15 set. 2008.

OLIVEIRA, R. **O que é *webcam*?**. Disponível em: <http://www.richard.eti.br/duvidas43.html>. Acesso em: 15 jul. 2009.

PARANHOS, L. **Termos mais usados em fotografia digital**. Disponível em: <http://blog.laumidia.com.br/2009/01/02/termos-mais-usados-em-fotografia-digital/>. Acesso em: 15 set. 2009.

PEDROSA, G. V.; BARCELOS, C. A. Z. **O uso de equações diferenciais parciais na eliminação de ruídos e segmentação de imagens**. Universidade Federal de Uberlândia. Uberlândia, 2008. Disponível em: <http://www.horizontecientifico.propp.ufu.br/include/getdoc.php?id=1160&article=485&mode=pdf>. Acesso em: 17 ago. 2009.

PELEGRINA, J. A. **DicWeb dicionário de informática**: a. Disponível em: <http://www.dicweb.com/aa.htm>. Acesso em: 15 set. 2009a.

PELEGRINA, J. A. **DicWeb dicionário de informática**: d. Disponível em: <http://www.dicweb.com/dd.htm>. Acesso em: 15 set. 2009b.

PEREIRA, D. S. A.; *et. al.* **Identificação de faces em imagens bidimensionais**. Revista TECCEN - Edição Especial, v. 2, n. 1, mar. 2009. Disponível em:

<http://www.uss.br/web/hotsites/revista_teccen3/artigo04.pdf>. Acesso em: 01 set. 2009.

PINTO, B. F.; REINBRECHT, C. R. W. **Aplicação de visão computacional em linguagem de programação C#**. Pontifícia Universidade Católica do Rio Grande do Sul. Porto Alegre, 2007. Disponível em: <http://www.inf.pucrs.br/~petinf/homePage/publicacoes/documentos/relatorios%20tecnico/cezar.reinbrecht_2007.pdf>. Acesso em: 01 set. 2008.

RAITTZ, R. T. ***Free associative neurons - FAN***: uma abordagem para reconhecimento de padrões. Dissertação (mestrado). Universidade Federal de Santa Catarina. Florianópolis, 1997. Disponível em: <http://www.eps.ufsc.br/disserta98/raittz/cap2.htm#2>. Acesso em: 20 ago. 2009.

REIS, K. C. **Mini-dicionário técnico de informática:** parte 3. Disponível em: <http://www.juliobattisti.com.br/tutoriais/keniareis/dicionarioinfo003.asp>. Acesso em: 10 set. 2009.

REZENDE, J. A. R.; *et. al.* **Tratamento de imagens para sistemas de reconhecimento facial**. Disponível em: <http://www.emc2008.iprj.uerj.br/down.php?fid=127>. Acesso em: 01 set. 2009.

RIBINIK, S. **Introdução ao código de barras e à identificação**. Disponível em: <http://artesfinais.com/Imagens/Intro_Codigo_Barras_Identifica.pdf.zip>. Acesso em: 12 set. 2009a.

RIBINIK, S. **O que é o código de barras**. Disponível em: <http://www.gs1brasil.org.br/main.jsp?lumPageId=4080818B10B6BEFB0110B6C71BE0027C&lumItemId=CF3B153DA5244C049EDFDEC30869C1FF>. Acesso em: 12 set. 2009b.

RIBINIK, S. **Passo 6:** escolher o tamanho do código de barras. Disponível em: <http://www.gs1brasil.org.br/main.jsp?lumPageId=4080818B10B6BEFB0110B6C71BF0027E&lumI=gs1.bibliotecavirtual.list&bvTopicId=480F89A81781B3DD011782B44F84350B&itemId=480F89A81781B3DD011782EA0B8E41DE>. Acesso em: 12 set. 2009c.

SCURI, A. E. **Fundamentos da imagem digital**. Tecgraf/PUC – Rio. Rio de Janeiro, 1999. Disponível em: <http://www.dcmm.puc-rio.br/download/Fundamentos%20da%20Imagem%20Digital.pdf>. Acesso em: 24 abr. 2009.

SEARA, D. M. **Algoritmos para detecção de bordas**. Tese (mestrado). Universidade Federal de Santa Catarina - UFSC. Florianópolis, 1998. Disponível em: <http://www.inf.ufsc.br/~visao/1998/seara/index.html>. Acesso em: 23 abr. 2009.

SILVA, A. L. **Tópicos sobre processamento de imagens**. Relatório de estagio curricular supervisionado. Centro Federal de Educação Tecnológica do Paraná. Cornélio Procópio, 2004. Disponível em: < http://inf.cp.cefetpr.br/fabricio/ensino/orientacoes/relestagioalbertosilva.pdf >. Acesso em: 10 set. 2008.

SILVA, F.J.V.; ALVES, C.H.F. **Aplicação de técnicas de processamento de imagens digitais em imagens geradas por ultra-som**. VIII ERMAC 8º Encontro Regional de Matemática Aplicada e Computacional. Universidade Federal do Rio Grande do Norte. Natal, 2008. Disponível em: <http://www.dimap.ufrn.br/~sbmac/ermac2008/Anais/Resumos%20Estendidos/aplica%E7%E3o%20de%20tecnicas%20de%20PS_Silva.pdf>. Acesso em: 23 abr. 2009.

SILVA, F. T.; PAPANI, F. G. **Código de barras**. Universidade Estadual do Oeste do Paraná. Cascavel, 2008. Disponível em: <http://projetos.unioeste.br/cursos/cascavel/matematica/xxiisam/artigos/17.pdf>. Acesso em: 25 abr. 2009.

SILVA, J. C. M. **Parâmetros da imagem no domínio digital:** o que é pixel, resolução de imagem, resolução de captura e outras características da imagem digital ou digitalizada. Disponível em: <http://www.novomilenio.br/comunicacoes/1/artigo/12_julio_martins.pdf>. Acesso em: 25 set. 2008.

SOARES, R. C. **Estudo de código de barras por análise de imagens**. Dissertação (Mestrado). Universidade Estadual de Campinas. Campinas, 2001. Disponível em: <http://libdigi.unicamp.br/document/?code=vtls000244042>. Acesso em: 24 ago. 2008.

SOUZA, J. A. **Reconhecimento de padrões usando indexação recursiva**. Tese (Doutorado). Universidade Federal de Santa Catarina (UFSC). Florianópolis, 1999. Disponível em: <http://www.eps.ufsc.br/teses99/artur/cap2.htm>. Acesso em: 15 jun. 2009.

SOUZA, U. L.; PEREIRA, T. R. **Fotodiodos e fototransistores**. Disponível em: <http://www.etb.com.br/professor/materialdeapoio/transistor.pdf>. Acesso em: 10 set. 2009.

STEMMER, M. R.; *et al.* **Apostila de sistemas de visão**. Universidade Federal de Santa Catarina. Florianópolis, 2005. Disponível em: <http://s2i.das.ufsc.br/harpia/downloads/apostila-sistemas-visao.pdf>. Acesso em: 10 set. 2008.

TAVARES, A.; *et. al*. **Desenvolvimento de um projeto de captura e edição de imagens a partir do uso de um processo pesado**. Unibratec - União Brasileira dos Institutos de Tecnologia. Recife, 2007. Disponível em: <http://www.unibratec.com.br/revistacientifica/n2_artigos/n2_tavares.pdf>. Acesso em: 24 abr. 2009.

TREVISAN, A. **Códigos 2D (*i-nigma*)**. Disponível em: <http://www.trevisantecnologia.com.br/produtos_código.jsp>. Acesso em: 17 jul. 2009.

VALE, G. M.; DAL POZ, A. P. **Processo de detecção de bordas de *Canny***. Bol. Ciênc. Geod., sec. Artigos, Curitiba, v. 8, n. 2, 2002. Disponível em: <http://ojs.c3sl.ufpr.br/ojs2/index.php/bcg/article/viewFile/1421/1175>, Acesso em: 15 mai. 2009.

# Creative Commons License Deed
# Atribuição - Uso não-comercial - Compartilhamento pela mesma licença 3.0 Brasil

**Você tem a liberdade de:**

**Compartilhar** - copiar, distribuir e transmitir a obra.
**Remixar** - criar obras derivadas.

**Sob as seguintes condições:**

**Atribuição -** Você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).

**Uso não-comercial** - Você não pode usar esta obra para fins comerciais.

**Compartilhamento pela mesma licença** - Se você alterar, transformar ou criar em cima desta obra, você poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.

**Ficando claro que:**

**Renúncia** - Qualquer das condições acima pode ser renunciada se você obtiver permissão do titular dos direitos autorais.

**Domínio Público** - Onde a obra ou qualquer de seus elementos estiver em domínio público sob o direito aplicável, esta condição não é, de maneira alguma, afetada pela licença.

**Outros Direitos** - Os seguintes direitos não são, de maneira alguma, afetados pela licença:

- Limitações e exceções aos direitos autorais ou quaisquer usos livres aplicáveis;
- Os direitos morais do autor;
- Direitos que outras pessoas podem ter sobre a obra ou sobre a utilização da obra, tais como direitos de imagem ou privacidade.

**Aviso** - Para qualquer reutilização ou distribuição, você deve deixar claro a terceiros os termos da licença a que se encontra submetida esta obra. A melhor maneira de fazer isso é com um link para esta página.