Pontifícia Universidade Católica de Minas Gerais

ENGENHARIA MECÂNICA (ÊNFASE EM MECATRÔNICA)

Código de Barras – Leitura através de uma Câmera

Professor: Denílson Laudares Rodrigues **Disciplina**: Elementos de Robótica

Turma: Sexta- feira, 17:20

Alunos:
Arthur Baião e Cruz (arthurbaiaoecruz@yahoo.com.br)
Marcio Prado (marciopradomp@gmail.com)
Thiago Pinheiro Latorre (latorrethiago@hotmail.com)

Belo Horizonte
Novembro de 2010

Pontifícia Universidade Católica de Minas Gerais

Proposta Inicial para o projeto de um leitor de Código de Barras através de uma câmera.

Trabalho apresentado à disciplina
Elementos de Robótica do curso de Engenharia
Mecatrônica Da Pontifícia Universidade
Católica de Minas Gerais

Orientador: Denílson Laudares Rodrigues

Sumário

Belo Horizonte
2010

1- Introdução

Este projeto visa o desenvolvimento de técnicas para decodificação de código de barras, através de padrões de imagens provenientes de fotos.

O código de barras tem como principal função o armazenamento de dados de forma rápida, segura e otimizada, para posterior identificação do elemento armazenado, podendo este ser desde: números, nomes, data e diversas outras aplicações. O código de barras é tão confiável e tão preciso que o uso deste sistema de codificação encontra-se presente em praticamente todos os produtos comercializados atualmente. Com sua implementação são gerados inúmeros benefícios e o presente projeto busca um melhor entendimento deste assunto.

Os códigos de barras dividem-se em dois grupos: os códigos de barras numéricos e os alfanuméricos (sendo os alfanuméricos capazes de representar números, letras e caracteres de função especial ao mesmo tempo). Os códigos de barras são diferenciados entre si pelas regras de simbologia. Cada simbologia trata como os dados serão codificados.

O código a ser estudado é o código 39, que é uma simbologia de código de barras que codifica letras maiúsculas, dígitos, e alguns símbolos especiais.

2- Objetivo

O projeto visa desenvolver um algoritmo em linguagem MATLAB capaz de reconhecer, decodificar e informar os dados capturados através de padrões de imagem previamente escolhidas do código 39. A captura destas imagens serão através de uma câmera acoplada ao computador. Todos os algoritmos envolvidos neste trabalho estão presentes nos Anexos I,II e III.

A elaboração segue as seguintes etapas:

- Planejamento inicial (definições gerais do projeto);
- Revisão bibliográfica (conteúdo teórico acerca do projeto);
- Detalhamento do código 39 (definições, estrutura, configurações, etc);
- Programação (elaboração de programa em Matlab para decodificação de códigos de barras do tipo 39);
- Processamento (tratamento de determinada imagem para obtenção de dados);
- Decodificação (elaboração de algoritmo para a decodificação do código de barras);
- Conclusões;
- Relatório final.

3- Necessidades básicas do trabalho a ser realizado

Para a elaboração deste trabalho serão necessárias basicamente as imagens (códigos de barras) para decodificação, e o algoritmo que será desenvolvido através do software em Matlab para a interpretação destas imagens.

4- Divisão de tarefas:

Arthur Baião e Cruz : a cargo da programação em Matlab do código 39, Processamento, Análise dos resultados, Relatório parcial e Conclusão.

Marcio Prado: a cargo da programação em Matlab do código 39, Processamento, Testes e Modificações, Decodificação, Relatório Parcial e Conclusão.

Thiago Pinheiro Latorre: a cargo da Conceituação teórica, programação em Matlab do código 39, Decodificação, análise dos dados, Relatório Final e Conclusão.

Nota-se logo, que muitas atividades se repetem, para cada membro do grupo, devido ao fato de que por serem atividades longas ou conclusivas, necessitam de acompanhamento paralelo de todo o grupo, para uma melhor percepção dos dados. Portanto, a programação do código 39, ficou a cargo de todo o grupo, com o intuito de o mesmo ser aprimorado com o tempo de trabalho e também para facilitar correções, no programa. Com isso, diminuímos as perdas e o trabalho pode seguir um fluxo de forma mais coerente, até a data estimada de entrega (ainda não estipulada).

5- Conceitos Básicos

Apresentamos neste item, diversos conceitos que envolvem a descrição dos Códigos de Barras estudados e métodos para identificação precisa de suas medidas. Serão

apresentados também breves históricos da simbologia estudada, assim como as tabelas de especificações que as regem.

5.1- Benefícios oferecidos pelos Códigos de Barras

- Melhoria na eficiência operacional - Uma vez que os Códigos de Barras oferecem um armazenamento de informação mais rápido e seguro,a velocidade de operação é favorecida .
- Redução de Tempo - Dependendo da aplicação, a redução de tempo gasto pode ser um fator importante. O registro automático de informações de lotes de mercadorias transportadas e dos preços de produtos em caixas de supermercados são exemplos deste benefício.
- Redução de Erros - A incidência de erros em uma operação pode ser uma fonte potencial a um aumento de custos e problemas. Existem diversas situações que expressam bem a importância da exatidão dos dados a serem analisados, umas delas estão presentes na área farmacêutica e de análise em bancos de sangue.
- Redução de Custos - Os equipamentos exigidos em um sistema de codificação por meio dos Códigos de Barras são de baixo custo, além disso, a confiabilidade e rapidez de leitura favorece a uma redução de gastos.
- Benefícios ao usuário - os equipamentos de leitura e impressão são flexíveis e fáceis de se instalar. Além disso, o usuário pode padronizar suas próprias etiquetas, interagindo o usuário às suas necessidades.

5.2- Princípios de Codificação

Existem basicamente dois princípios de codificação de informação em uma dimensão. Uma delas consiste na subdivisão de intervalos em ***módulos*** (menor unidade da medida de largura de uma barra ou espaçamento) que assumem valores "1 " ou "0", onde os módulos com valores "1" são expressos em forma impressa de barra escura (doravante denominada barra), e aqueles com valores "0" são interpretados como espaços entre as barras escuras; caracterizando a ***Codificação por Análise de Reflexão***, também denominada "Código Delta". Cada barra ou espaço pode conter um ou mais módulos. A outra metodologia, denominada ***Codificação por Largura de Módulo*** assume cada barra ou espaço como um bit, impondo valor lógico "0" para o elemento (seja barra ou espaço) estreito, e valor lógico "1" para o elemento largo. Este tipo de codificação é também denominado Código Binário, uma vez que são usadas apenas duas larguras pré-definidas, geralmente na proporção 3:1.

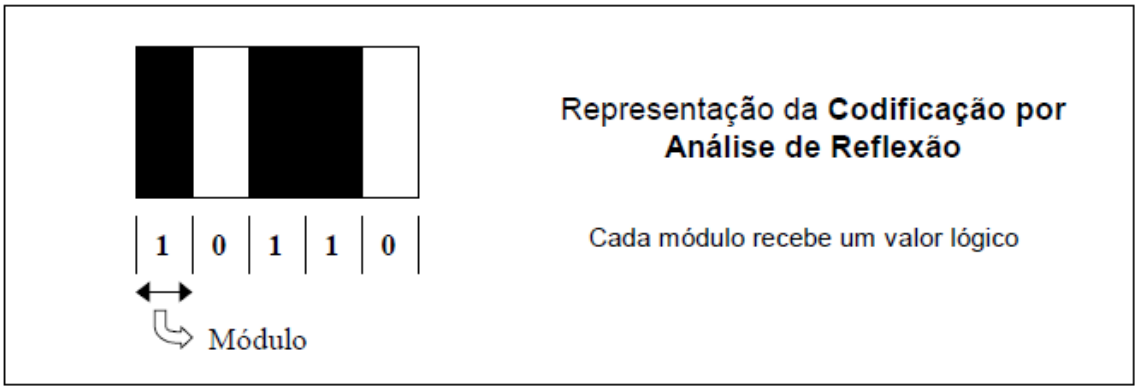

Figura 1(a)

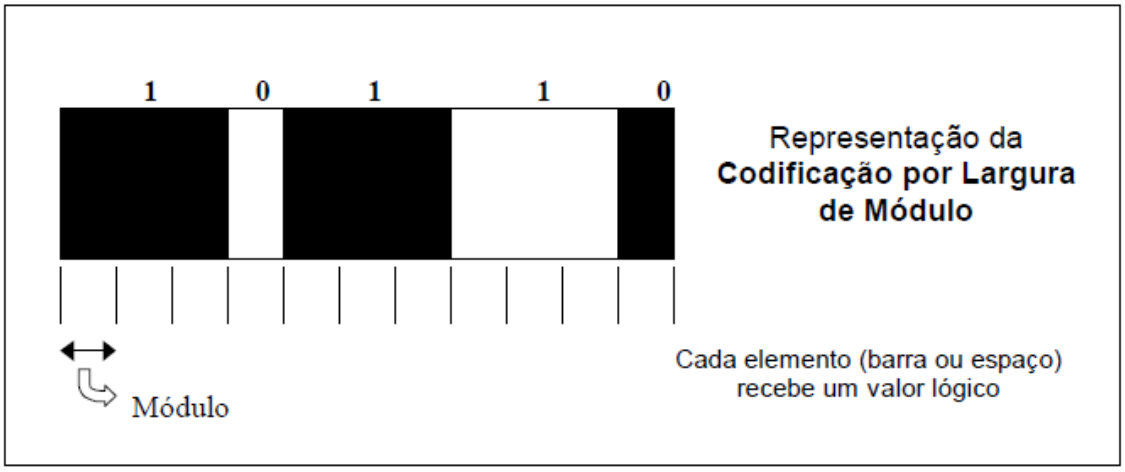

Figura 2(b)

Analisando a figura 2(a), observamos que para codificar a ***string*** 10110, pelo método de codificação por reflexão, são necessários cinco módulos, enquanto que para o método de análise de largura de módulo, são necessários onze módulos se a proporção de um elemento estreito para um largo for 1:3.

5.3- Código 39 (Código 3 de 9)

Esta simbologia surgiu em 1975, adotada pelo Departamento de Defesa dos Estados Unidos, que oficializou o Código 39 para todos os suprimentos, mediante a Norma de Padronização Militar 1189, em 1982. Atualmente esta simbologia é adotada em diversos países nas áreas de serviços hospitalares, transportes aéreos, bibliotecas, entre outras.

5.3.1- Definição

O Código 39 é um código binário alfanumérico capaz de codificar 44 caracteres

(26 letras, 10 algarismos numéricos, 1 caractere de espaçamento, e 7 caracteres de símbolos). Um código binário possui elementos com apenas duas larguras possíveis para expressar os valores lógicos “0” e “1”, ou seja, para uma barra ou espaçamento, somente são possíveis espessuras largas (valor lógico “1”) ou estreitas (valor lógico “0”).

5.3.2- Estrutura

O Código 39 é constituído por palavras-código que possuem 9 elementos (barras e espaçamentos) dos quais 3 deles são largos. Esta estrutura justifica o nome da simbologia. Cada palavra-código que expressa um caractere possui cinco barras e quatro espaçamentos. Existe um espaçamento intercaractere na concatenação das palavras código, caracterizando esta simbologia como “*discreta*”. A largura do espaçamento intercaractere deve ser aproximadamente idêntica à largura de um elemento estreito. A princípio, não há um tamanho fixo para a representação do Código 39, ou seja, é possível codificar quantos caracteres forem necessários. Esta simbologia usa o caractere asterisco (*) como padrão de início e fim do símbolo impresso. A Tabela 1 mostra o conjunto de padrões para cada um dos 44 caracteres possíveis de codificação e a Figura 3 ilustra a estrutura de um Código 39.

Caractere	Padrão	Barras	Espaços
1		10001	0100
2		01001	0100
3		11000	0100
4		00101	0100
5		10100	0100
6		01100	0100
7		00011	0100
8		10010	0100
9		01010	0100
0		00110	0100
A		10001	0010
B		01001	0010
C		11000	0010
D		00101	0010
E		10100	0010
F		01100	0010
G		00011	0010
H		10010	0010
I		01010	0010
J		00110	0010
K		10001	0001
L		01001	0001

Caractere	Padrão	Barras	Espaços
M		11000	0001
N		00101	0001
O		10100	0001
P		01100	0001
Q		00011	0001
R		10010	0001
S		01010	0001
T		00110	0001
U		10001	1000
V		01001	1000
W		11000	1000
X		00101	1000
Y		10100	1000
Z		01100	1000
-		00011	1000
.		10010	1000
ESPAÇO		01010	1000
***		00110	1000
$		00000	1110
/		00000	1101
+		00000	1011
%		00000	0111

Tabela 1- Caracteres do código 39

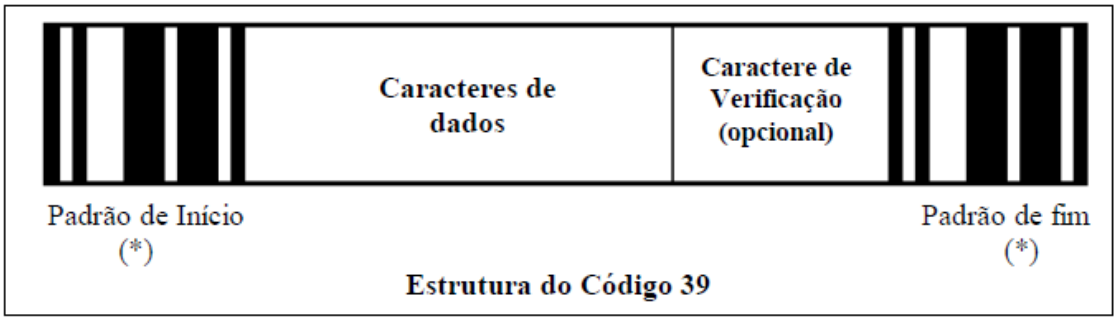

Figura 2 – Estrutura do código 39

Como ilustrado na Figura 3, após os padrões referentes à codificação dos dados, existe a possibilidade da adoção de um padrão destinado a um caractere de verificação. Além disso, existe uma margem de silêncio presente nas laterais do símbolo, que devem ser obedecidas para evitar erros de leitura pelo dispositivo ótico.

A estrutura do Código de Barras é mostrada na Figura 1

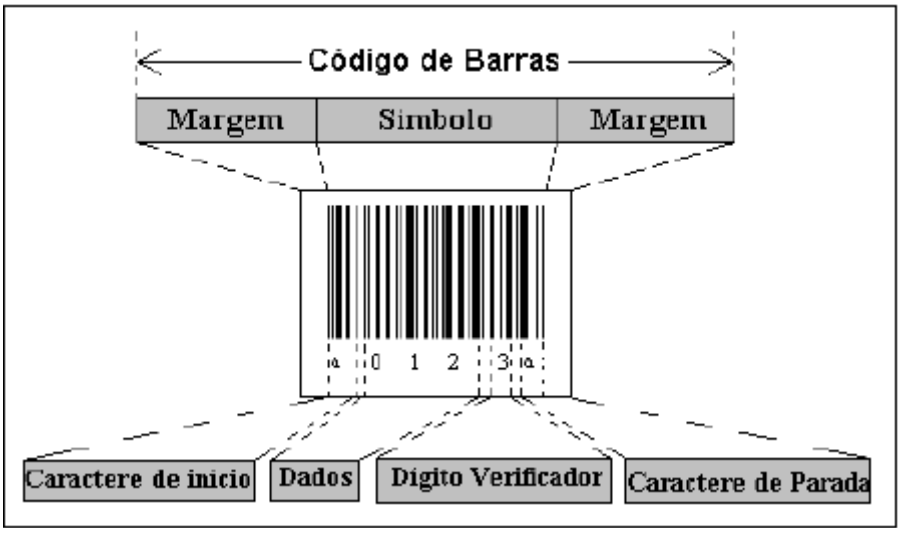

Figura 3

- Margens Inicial e Final de Silêncio (ou Zonas de Silêncio) - são espaços geralmente brancos, onde nada é impresso. Estas margens devem possuir dez vezes a largura da barra mais estreita do código.
- Símbolo - consiste na área composta de barras verticais paralelas espaçadas de forma variada.Caractere de Início - indica o início do dado. Este caractere varia conforme o sistema de codificação.
- Dados - área destinada ao conteúdo de informações a serem codificadas.
- Dígito Verificador – certifica a ocorrência erro na leitura através de um dígito de dado interpretado.
- Caractere de Parada - indica o término do dado.

5.3.3- Decodificação

A fim de habilitar o leitor ótico a distinguir elementos largos e estreitos, é necessário adotar uma razão de proporção. Mediante a resolução adotada no processo de impressão, a largura de um elemento largo deverá ser no mínimo duas vezes a largura e

m elemento estreito. A proporção mais adequada é 3:1. Deve-se seguir o critério de uniformidade, ou seja, os elementos do mesmo tipo devem possuir a mesma medida. Dessa forma, a largura de uma barra estreita deve ser idêntica à largura de um espaçamento estreito, e vice-versa. No momento da leitura, os padrões de início e fim do símbolo, definidos pelo caractere asterisco (*), indicam o sentido da varredura do leitor ótico, uma vez que as leituras deste padrão em sentidos opostos geram resultados distintos. As Figuras 4 e 5 ilustram como o sentido de varredura é determinado através dos padrões laterais.

Figura 4 – Sentido de varredura

Figura 5 – Sentido de varredura

6- Processos para Decodificação

Os códigos de barras são analisados e interpretados segundo diversos dispositivos de leitura: desde leitores manuais de feixe pontual aos dispositivos CCD (Charge Coupled Device). Mediante a sua popularização e vasta aplicação para fins de economia de tempo, o código de barra necessita de um sistema de análise de imagens aprimorado, a fim de permitir uma decodificação diretamente a partir de uma imagem construída pelo dispositivo ótico.

Este sistema envolve conceitos de reconhecimento ótico de padrões, classificação de objetos e controle de qualidade. Para uma otimização de decodificação, é necessária a aplicação de algumas técnicas de processamento de imagens. No presente trabalho, o sistema de análise consiste em alguns conceitos matemáticos e a captura das imagens é feita através de uma câmera acoplada a um computador.

7- Princípios de Análise

A partir da informação de entrada, que será uma imagem composta de um símbolo em uma posição aleatória, uma análise deve ser feita a fim de se obter uma série de informações contidas nesta imagem. Independentemente da simbologia alvo de decodificação, uma seqüência de passos deve ser seguida. O passo inicial consiste na aquisição do símbolo pelo dispositivo ótico. Um pré-processamento é aplicado na imagem inicial capturada, a fim de solucionar problemas de enquadramento do símbolo. A partir daí, a imagem, que inicialmente possui 256 níveis de cinza, é convertida em uma imagem binária (níveis preto e branco). Após a binarização, conforme a simbologia

presente na imagem inicial, é necessária a obtenção de medidas específicas para a classificação dos padrões encontrados na leitura do Código de Barras.

8- Enquadramento

No momento da captura do símbolo pelo leitor ótico, um fator importante que deve ser levado em consideração é o posicionamento do eixo longitudinal do símbolo, ou seja, o ângulo do eixo longitudinal com a horizontal.

O desalinhamento característico do Código de Barras com a horizontal, no momento da digitalização deve ser determinado e corrigido. Existem algumas técnicas que detectam o ângulo de desalinhamento. No nosso trabalho, utilizaremos a técnica de estimação do ângulo, o ângulo do eixo longitudinal com relação à horizontal deve ser nulo ou muito próximo de zero, a fim de evitar problemas com a imagem digitalizada, que podem provocar erros de decodificação. A Figura 6 ilustra o processo de enquadramento.

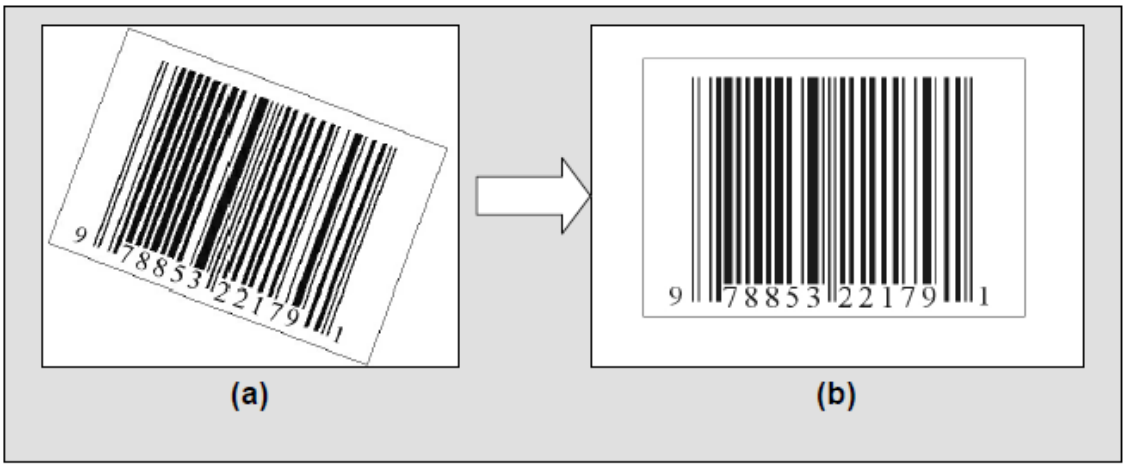

Figura 6 - Enquadramento

9- Técnicas de Binarização Avaliadas

A detecção de barras e espaços está sujeita a erros conforme a metodologia adotada para esse fim, a variação de contraste no momento da varredura, ou mesmo a iluminação do ambiente. Com isso, deve-se monitorar mudanças de intensidade de luz refletida durante a varredura, a fim de se estabelecer uma uniformidade das condições de leitura. Ou seja, é necessário um sistema de controle de ganho automático de luminosidade no dispositivo de leitura.

As técnicas de detecção de barras e espaços consistem em um processo de binarização, que transforma a imagem de 256 níveis adquirida pelo leitor ótico em uma nova imagem de 2 níveis de cinza (imagem binária: preto e branco), destacando as transições entre barras e espaços, necessárias para a decodificação. Este processo pode ser realizado através de alguns métodos, tais como *Limiarização* (Threshold global).

10- Threshold (Limiarização)

O processo de Limiarização será a primeira, e talvez a mais crítica técnica de processamento de imagem aplicada antes do processo de decodificação do símbolo impresso. Limiarização é um método de segmentação de imagens em níveis de cinza que extrai do plano de fundo de uma imagem, objetos de interesse, partindo-se do princípio que o objeto e o plano de fundo da imagem possam ser diferenciados entre si pelos seus níveis de cinza. A determinação de um valor para a aplicação de uma Limiarização na imagem de um Código de Barras requer critérios bastante flexíveis, pois existem muitos fatores que influenciam de forma diversificada o sinal de saída no

leitor ótico. O problema de espalhamento de tinta é um desses fatores e que dificulta muito a eficiência da escolha do valor de limiar, pois a intensidade do espalhamento de tinta não é um dado exato.

11- Fluxograma de programação

Abaixo colocamos nosso fluxograma para o funcionamento do programa.

- Iniciar função de câmera no matlab;
- Captura de Imagem através de uma tecla;
- Leitura de imagem: reconhecimento da imagem pelo programa MatLab;
- Tratamento de imagem: filtragem para melhor definição da imagem capturada;
- Cálculo do threshold (transformar imagens em níveis de cinza);
- Binarização: transformação da imagem com 256 variações de cores em binário (0 e 1);
- Corrigir rotação: enquadramento centrado da imagem;
- Remoção dos retalhos ou bordas do enquadramento;
- Medição da largura das faixas brancas e pretas: varredura da imagem para a obtenção da largura das faixas;
- Vetorização de acordo com o tamanho das faixas: análise de largura das faixas;
- Agrupamento dos bits: para obtenção de grupos que formam os caracteres;
- Reconhecimento dos caracteres: comparação com um banco de dados previamente armazenados;
- Leitura do código: Decodificação do código, que será mostrado na tela.
- Mostra palavra em tela

12- Conclusão

O presente trabalho abordou um tema de grande utilização atualmente, que é a utilização de códigos de barras para identificação de diversos objetos. Com isso foi possível observar o processo de reconhecimento destes códigos.

Nosso trabalho descreveu uma abordagem para decodificação de códigos de barras unidimensionais, usando análise de imagens.

Ao longo do desenvolvimento do presente trabalho, elaboramos algoritmos destinados às etapas de decodificação, tais como captura de um símbolo presente em uma imagem, análise de validação do símbolo, tratamento do símbolo a ser analisado, alinhamento do eixo longitudinal do símbolo, verificação de características relevantes do símbolo, e determinação do dado codificado.

Os algoritmos foram desenvolvidos em linguagem MATLAB. No decorrer do desenvolvimento destes algoritmos foram encontrados inúmeros obstáculos e dificuldades, pois fez-se necessário um aprofundamento acerca das funções do programa MATLAB.

13- Bibliografia

- Trabalhos dos alunos de Engenharia Mecatrônica realizados anteriormente
- Michael Fairhurst C. **Computer vision for robotic systems: na introduction**
- HOW STUFF WORKS. Disponível em http://www.howstuffworks.com/
- WIKIPEDIA. Disponível em http://pt.wikipedia.org/wiki/C%C3%B3digo_de_barras/.
- Vera Lúcia Pinheiro da Silva, Aplicações Práticas de Códigos de Barras, São Paulo: Nobel, 1989.
- http://www.macoratti.net/07/12/crys_c39.htm
- http://www.barcodesinc.com/free-barcode-font/
- http://www.mathworks.com/image-video-processing/demos.html
- Arquivos do Help do programa Matlab.

Anexo I - Algoritmo principal

O algoritmo abaixo é o algoritmo principal do programa que serve para a inicialização da câmera, captura e tratamento da imagem no matlab. Um detalhe importante é que utilizamos a câmera do laboratório de robótica, “matrox”, por este motivo usa-se este nome no programa. Segue o programa comentado.

Algoritmo: código_barras.m

```
%%% limpar a tela e a memoria
clear all
clc

%iniciar camera
vid=videoinput('matrox');
vi.FramesPerTrigger=1;
preview(vid)
pause
start(vid)
pause(0.1)
foto=getdata(vid);

%discartar dimençoes da imagem (reescrever imagem apenas em 2D)
aux=size(foto);
for x=1:aux(2)
    for y=1:aux(1)
        img1(y,x)=foto(y, x, 1, 1);
    end
end

%%%%%% tratamento de imagem %%%%%%
%filtro passa baixa
matr=ones(3,3)/10;
img2=imfilter(img1,matr);



%%% calculo do threshold e binarizaçao
level = graythresh(img1);
img3 = im2bw(img1,level);

%mostrar imagem original, imagem suavizada, imagem binarizada e
histograma
subplot(2,2,1)
imshow(img1)
```

```
subplot(2,2,2)
imshow(img2)
subplot(2,2,3)
imshow(img3)
subplot(2,2,4)
imhist(img2)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% corrigir rotaçao
offset_x=100;           % espaçamento do meio da figura
aux=size(img3);         % variavel auxiliar (tamanho da figura)
a=round(aux(2)/2);      % valor da metade da figura


y=2;                    % valor inicial para varredura da imagem

%varre a imagem ate a borda no primeiro ponto
while (img3(y,a-offset_x)==1)||(y==aux(1))
    y=y+1;
end
y1=y;

y=2;                    % valor inicial para varredura da imagem

%varre a imagem ate a borda no segundo ponto
while (img3(y,a+offset_x)==1)||(y==aux(1))
    y=y+1;
end
y2=y;

%calcula angulo
ang=(tan((y2-y1)/(2*offset_x)));

%converte o valor do angulo de radianos para graus
angulo=rad2deg(ang);

%usa funçao de rotaçao do matlab
img4=imrotate(img3,angulo,'bilinear','loose');
figure, imshow(img4)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% retirar retalhos da borda
%canto inferior esquerdo

%inicializaçao dos limites para reescrever a imagem
a1=1;     %limite minimo de X
b1=1;     %limite minimo de Y
a2=1;     %limite maximo de X
b2=1;     %limite maximo de Y

%hipotese para rotaçao anti horaria
if angulo>0
    x=1;                        % valor inicial para varredura da imagem
    while img4(aux(1),x)==0   %verredura crescente ate primeiro pixel
branco em X
        x=x+1;
    end
    a1=x;   %salva valor do limite minimo de X
```

```
    y=1;    % valor inicial para varredura da imagem
    while img4(y,3)==0   %verredura crescente ate primeiro pixel
branco em Y
        y=y+1;
    end
    b1=y;   %salva valor do limite minimo de Y

    x=aux(2);    % valor inicial para varredura da imagem
    while img4(3,x)==0  %verredura decrescente ate primeiro pixel
branco em X
        x=x-1;
    end
    a2=x;        %salva valor do limite maximo de X

    y=aux(1);    % valor inicial para varredura da imagem
    while img4(y,aux(2))==0   %verredura decrescente ate primeiro
pixel branco em Y
        y=y-1;
    end
    b2=y; %salva valor do limite maximo de Y
end

%hipotese para rotaçao horaria
%homologo ao metodo da primeira hipotese
if angulo<0
    x=1;
    while img4(3,x)==0
        x=x+1;
    end
    a1=x;

    y=1;
    while img4(y,aux(2))==0
        y=y+1;
    end
    b1=y;

    x=aux(2);
    while img4(aux(1),x)==0
        x=x-1;
    end
    a2=x;

    y=aux(1);
    while img4(y,3)==0
        y=y-1;
    end
    b2=y;
end

%hipotese para imagem sem rotaçao
if angulo==0
    img5=img4;  % transcreve a imagem na integra
else            % reescreve a imagem sem bordas utilizando os limites
    c=1;        % variavel auxiliar para indice da nova imagem
    for x=a1:a2
        d=1;    % variavel auxiliar para indice da nova imagem
        for y=b1:b2
            img5(d,c)=img4(y,x);
```

```
            d=d+1;
        end
        c=c+1;
    end
end
figure, imshow(img5)

%%% reconhecer codigo de barras %%%
Cbarras %% agoritmo chamado para conclusão do programa
```

Anexo II - Algoritmo secundário

O presente algoritmo tem a função de complementar o algoritmo principal, fazendo a leitura e decodificação dos dados obtidos pela câmera.

Algoritmo :cbarras.m

```
clc
clear img codigo largura palavra lido
img=img5;

%%% mostrar imagem a ser trabalhada
imshow(img)

%%% iniciar variáveis
n=0;
b=1;
x=1;

%%% pegar dimensões da imagem e definir a altura onde será lido
aux=size(img);
h=round(aux(1)/2);

%%% loop para medir a quantidade de pixels pretos e brancos na altura
h
while x<aux(2)-10

% loop para pixels brancos
    while (img(h,x)==1)&(x<aux(2))
        n=n+1;
        x=x+1;
    end
% salvar quantidade de pixels no vetor se for diferente de zero
    if n~=0
        largura(b)=n;
        n=0;
        b=b+1;
    end
% loop para pixels pretos
    while (img(h,x)==0)&(x<aux(2))
        n=n+1;
        x=x+1;
    end
% salvar quantidade de pixels no vetor se for diferente de zero
    if n~=0
        largura(b)=n;
        n=0;
        b=b+1;
    end
end

%%% eliminar margem %%%
% ver quais medidas são falsas antes do código
```

```
a=2;
while largura(a)<11*min(largura)
    a=a+1;
end

% ver ultima medida valida
b=a+1;
while largura(b)<10*min(largura)
    b=b+1;
end


%%% transformar em código binario de acordo com largura da faixa
% faixa estreita=0; faixa larga=1
c=1;
for x=a+1:b
    if rem(x-a,10)~=0
        if (largura(x)<=3*min(largura))
            codigo(c)=0;
            c=c+1;
        else
            codigo(c)=1;
            c=c+1;
        end
    end
end

%%% interpretar o codigo binario uma palavra de 9 bits por vez
% divisao para obter grupos de 9 bits
for x=1:length(codigo)/9
    aux(2)=x*9;
    aux(1)=aux(2)-8;
    b=1;
%%% salvar os nove bits no vetor palavra
    for a=aux(1):aux(2)
    palavra(b)=codigo(a);
    b=b+1;
    end
    reconhece
end
%%% mostrar na tela os caracteres lidos
lido


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
%largura -> vetor contendo a espessura de cada faixa
%
%codigo -> vetor binario representativo das faixas
%
%palavra -> vetor binario com a palavra de 9 bits referente a 1
caracter  %
%lido -> vetor contendo todos os caracteres lido
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
```

Anexo III – Banco de dados da imagem

O algoritmo abaixo é chamado no algoritmo secundário, este algoritmo tem a função de organizar e identificar a imagem capturada comparando com o banco de dados para a leitura do código armazenado nas barras.

Algoritmo: reconhece.m

```
%%% banco de dados dos caracteres em binario (segundo o codigo 39)
 codigo0=[0 0 0 1 1 0 1 0 0];%0
 codigo1=[1 0 0 1 0 0 0 0 1];%1
 codigo2=[0 0 1 1 0 0 0 0 1];%2
 codigo3=[1 0 1 1 0 0 0 0 0];%3
 codigo4=[0 0 0 1 1 0 0 0 1];%4
 codigo5=[1 0 0 1 1 0 0 0 0];%5
 codigo6=[0 0 1 1 1 0 0 0 0];%6
 codigo7=[0 0 0 1 0 0 1 0 1];%7
 codigo8=[1 0 0 1 0 0 1 0 0];%8
 codigo9=[0 0 1 1 0 0 1 0 0];%9
codigo10=[1 0 0 0 0 1 0 0 1];%a
codigo11=[0 0 1 0 0 1 0 0 1];%b
codigo12=[1 0 1 0 0 1 0 0 0];%c
codigo13=[0 0 0 0 1 1 0 0 1];%d
codigo14=[1 0 0 0 1 1 0 0 0];%e
codigo15=[0 0 1 0 1 1 0 0 0];%f
codigo16=[0 0 0 0 0 1 1 0 1];%g
codigo17=[1 0 0 0 0 1 1 0 0];%h
codigo18=[0 0 1 0 0 1 1 0 0];%i
codigo19=[0 0 0 0 1 1 1 0 0];%j
codigo20=[1 0 0 0 0 0 0 1 1];%k
codigo21=[0 0 1 0 0 0 0 1 1];%l
codigo22=[1 0 1 0 0 0 0 1 0];%m
codigo23=[0 0 0 0 1 0 0 1 1];%n
codigo24=[1 0 0 0 1 0 0 1 0];%o
codigo25=[0 0 1 0 1 0 0 1 0];%p
codigo26=[0 0 0 0 0 0 1 1 1];%q
codigo27=[1 0 0 0 0 0 1 1 0];%r
codigo28=[0 0 1 0 0 0 1 1 0];%s
codigo29=[0 0 0 0 1 0 1 1 0];%t
codigo30=[1 1 0 0 0 0 0 0 1];%u
codigo31=[0 1 1 0 0 0 0 0 1];%v
codigo32=[1 1 1 0 0 0 0 0 0];%x
codigo33=[0 1 0 0 1 0 0 0 1];%z
codigo34=[1 1 0 0 1 0 0 0 0];%
codigo35=[0 1 1 0 1 0 0 0 0];%
codigo36=[0 1 0 0 0 0 1 0 1];%
codigo37=[1 1 0 0 0 0 1 0 0];%
codigo38=[0 1 1 0 0 0 1 0 0];%
codigo39=[0 1 0 1 0 1 0 0 0];%
```

```
codigo40=[0 1 0 1 0 0 0 1 0];%
codigo41=[0 1 0 0 0 1 0 1 0];%
codigo42=[0 0 0 1 0 1 0 1 0];%
codigo43=[0 1 0 0 1 0 1 0 0];%

%%% matriz para organizar os caracteres e facilitar a busca e
comparaçao
codigox=[codigo0; codigo1; codigo2; codigo3; codigo4; codigo5;
codigo6; codigo7; codigo8; codigo9; codigo10; codigo11; codigo12;
codigo13; codigo14; codigo15; codigo16; codigo17; codigo18; codigo19;
codigo20; codigo21; codigo22; codigo23; codigo24; codigo25; codigo26;
codigo27; codigo28; codigo29; codigo30; codigo31; codigo32; codigo33;
codigo34; codigo35; codigo36; codigo37; codigo38; codigo39; codigo40;
codigo41; codigo42; codigo43];
%%% vetor com os caracteres
caracter=char('0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'a',
'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o',
'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', '-', '.', ' ',
'$', '/', '+', '%',  '*');

%%% comparaçao da palavra proveniente do codigo de barras lido com o
banco de dados
for c=1:length(codigox)
    if palavra==codigox(c,:)
        lido(x)=caracter(c,:);
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
%codigo0 a codigo 43 -> vetor contendo a representaçao do caracter em
binario %
%codigox -> matriz contendo todas as representaçoes dos caracteres em
binario %
%caracter -> vetor contendo os caracteres
%
%palavra -> vetor binario com a palavra de 9 bits referente a 1
caracter      %
%lido -> vetor contendo todos os caracteres lido
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
```