Universidade Tecnológica Federal do Paraná – UTFPR
Departamento Acadêmico de Eletrônica – DAELN
Engenharia Eletrônica
Disciplina: IF69D – Processamento Digital de Imagens
Semestre: 2017/1
Prof: Gustavo B. Borba

RELATÓRIO
Detecção e decodificação de código de barras

Fábio Crestani / 1360817

Jun.2017

1 Objetivo

Este trabalho tem o objetivo de desenvolver um algoritmo para detecção e decodificação de códigos de barras a partir de arquivos de imagem. O algoritmo foi implementado em Matlab seguindo a especificação do código de barras do tipo EAN-13. O projeto foi dividido em três etapas:

- **Decodificação em imagens artificiais:** O código de barras é decodificado a partir de imagens geradas artificialmente e cujo código é conhecido e usado para a validação do algoritmo.
- **Decodificação em imagens reais:** O algoritmo aplicado no item anterior foi modificado para poder ser aplicado em um conjunto de imagens de códigos de barras recortados manualmente de fotos.
- **Detecção em imagens reais:** O código de barras é localizado e extraído de uma foto que retrata um cenário mais realista.

2 Fundamentação Teórica

2.1 Especificação do padrão EAN-13

Figura 1: Exemplo de código de barras EAN-13.

Existem vários padrões de códigos de barras disponíveis, porém, este trabalho limitou-se ao escopo da especificação EAN-13 por se tratar de um padrão amplamente utilizado no mercado brasileiro. O European Article Number-13 (EAN-13) [1] é um código de barras composto por 13 dígitos que codificam através da largura das barras, as informações

de país de origem, código do fabricante e código do produto, além de um último dígito de verificação de erro (CRC). A figura 1 mostra como o EAN-13 é organizado.

Do ponto de vista de codificação, o código de barras é dividido em três grupos: o primeiro dígito, o primeiro grupo e o segundo grupo, exemplificados na figura 2.

Figura 2: Divisão dos grupos da codificação do código de barras EAN-13.

A codificação do segundo grupo é sempre a mesma, porém, a codificação do primeiro grupo depende do primeiro dígito. O segundo grupo é codificado com o *R-CODE*, enquanto que o primeiro grupo é codificado com o *L-CODE* e o *G-CODE*, alternadamente. A tabela 1 lista qual código deverá ser usado a depender do valor do primeiro dígito. Os valores da segunda e da terceira coluna da tabela correspondem aos dígitos de cada grupo, da esquerda para a direita.

O valor primeiro do primeiro dígito, é por sua vez, determinado a partir da paridade dos dígitos do primeiro grupo. Para cada dígito do primeiro grupo, conta-se o número de '1's (barras pretas, neste caso) do total de 7 bits do dígito. O valor do primeiro dígito é, portanto, localizado em uma tabela que relaciona as paridades (tabela 2).

Cada dígito é formato por 7 bits, ou seja, 7 barras verticais que são agrupadas lado-a-lado. A tabela 3 mostra como é feita a codificação de cada dígito para os três códigos possíveis. Nesta tabela, o valor '1' indica uma barra preta, enquanto que o valor '0' indica uma barra branca.

Tabela 1: Codificação dos grupos do código de barras de acordo com o primeiro dígito

Primeiro dígito	Primeiro grupo	Segundo grupo
0	LLLLLL	RRRRRR
1	LLGLGG	RRRRRR
2	LLGGLG	RRRRRR
3	LLGGGL	RRRRRR
4	LGLLGG	RRRRRR
5	LGGLLG	RRRRRR
6	LGGGLL	RRRRRR
7	LGLGLG	RRRRRR
8	LGLGGL	RRRRRR
9	LGGLGL	RRRRRR

Tabela 2: Paridades dos dígitos do primeiro grupo para determinação do primeiro dígito

Valor do primeiro dígito	Dígito 1	Dígito 2	Dígito 3	Dígito 4	Dígito 5	Dígito 6
0	Ímpar	Ímpar	Ímpar	Ímpar	Ímpar	Ímpar
1	Ímpar	Ímpar	Par	Ímpar	Par	Par
2	Ímpar	Ímpar	Par	Par	Ímpar	Par
3	Ímpar	Ímpar	Par	Par	Par	Ímpar
4	Ímpar	Par	Ímpar	Ímpar	Par	Par
5	Ímpar	Par	Par	Ímpar	Ímpar	Par
6	Ímpar	Par	Par	Par	Ímpar	Ímpar
7	Ímpar	Par	Ímpar	Par	Ímpar	Par
8	Ímpar	Par	Ímpar	Par	Par	Ímpar
9	Ímpar	Par	Par	Ímpar	Par	Ímpar

Tabela 3: Codificação dos códigos L-CODE, G-CODE e R-CODE

Dígito	L-code	G-code	R-code
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

3 Implementação

3.1 Detecção do código de barras

A detecção toma como premissa que o código de barras está na posição horizontal e não sofre oclusão ou distorção de perspectiva. A figura 3 mostra o diagrama de blocos do algoritmo implementado.

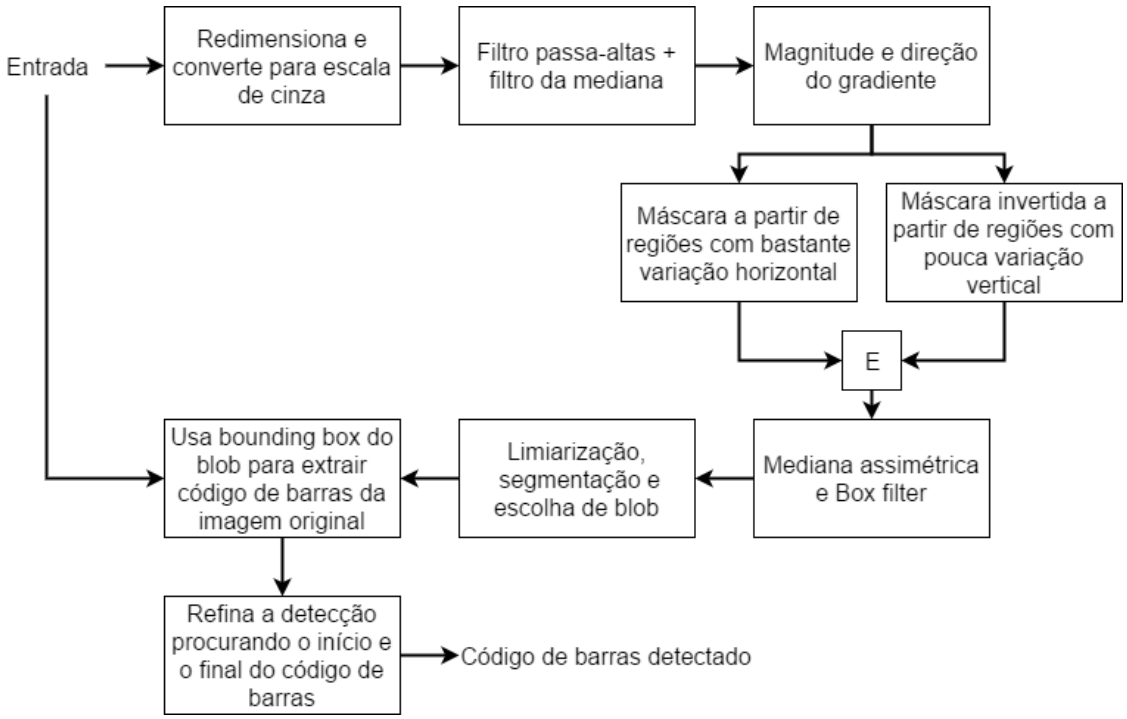


Figura 3: Diagrama de blocos da detecção.

A imagem de entrada é primeiramente redimensionada, se maior do que o tamanho esperado, e convertida para escala de cinza, caso seja colorida (Figura 4(a)). É aplicado um filtro passa-altas e em seguida, um filtro da mediana, para remoção do ruído (Figura 4(b)).

A detecção do código de barras baseia-se no fato de que a região do código de barras apresenta maior variação no sentido horizontal, devido à mudança brusca do branco para o preto entre as barras. Por esse motivo, são calculadas a magnitude a direção do gradiente da imagem. O gradiente é uma grandeza vetorial que pode ser decomposta nas componentes em x (figura 5(a)) e em y (figura 5(b)). Essa decomposição é feita através da função *imgradientxy* do Matlab [3] e é utilizada para realçar as regiões com maior variação horizontal e menor variação vertical. Estas regiões são manipuladas a fim de serem utilizadas como máscara para extração do código de barras.

À esta máscara é aplicado um filtro de mediana assimétrica (retangular) (figura 6(a)), a fim de eliminar o ruído e evitar a remoção de partes do código de barras da máscara. A máscara é então suavizada com um filtro da média e limiarizada para que seja utilizada como uma máscara lógica (0s ou 1s) (figura 6(b)).

São segmentados os blobs da máscara e são computadas algumas propriedades dos blobs através da função *regionprops* do Matlab [4]. Estas propriedades são utilizadas para a seleção do blob que apresente as características mais próximas das esperadas para o código de barras. São elas:

(a) *(b)*

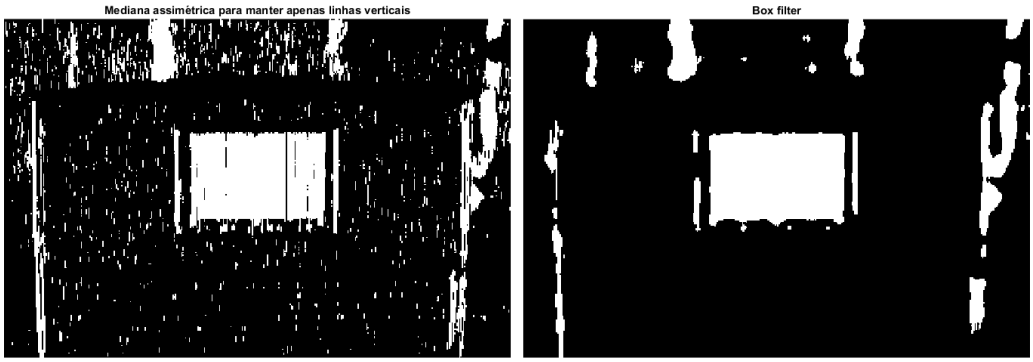
Figura 4: (a) Imagem redimensionada e convertida para escala de cinza. (b) Aplicado filtro passa-altas e filtro da mediana.

(a) *(b)*

Figura 5: (a) Máscara das regiões com maior gradiente horizontal. (b) Máscara negada das regiões com menor gradiente vertical.

- Área mínima: Assume-se que a área do código de barras não é muito menor que a área total da imagem.
- Razão largura-altura: Assume-se que a largura do código de barras é maior do que a altura.
- Razão área do blob pela área da bounding box: Assume-se que o blob referente ao código de barras possui área muito próxima à área do retângulo que engloba todas as extremidades do blob (chamado de bouding box).

Uma vez escolhido o blob (figura 7(a)), a sua bounding box é utilizada para a extração do código de barras a partir da imagem original (figura 7(b)).



(a) *(b)*

Figura 6: (a) Máscaras dos gradientes combinadas e filtradas por um filtro de mediana assimétrica (retangular). (b) Máscara resultante após aplicação do filtro da média e limiarização.

(a) *(b)*

Figura 7: (a) Segmentação e seleção de blob que mais se aproxime das características esperadas para o código de barras. (b) Sobreposição da região do código de barras detectado na imagem original.

3.2 Decodificação do código de barras

A figura 8 mostra um diagrama de blocos do algoritmo de decodificação. A partir de uma imagem com o código de barras, o algoritmo recorta uma região central (figura 9(a)), evitando assim o ruído dos números na parte inferior do código de barras. A imagem limiarizada e é calculada a média de cada coluna (figura 9(b)).

É determinado o início e o final exatos do código de barras, bem como as faixas de controle centrais. Assim, o código de barras é separado nos grupos 1 (esquerda) e 2 (direita) (figura 9(c)).

Para cada grupo, o vetor resultante é percorrido da esquerda para a direita, e são computadas as larguras de cada barra encontrada. Por fim, as larguras são normalizadas, dividindo-se todo o vetor de larguras pela soma de toda as larguras. As larguras são ainda arredondadas para valores inteiros e limitadas entre 1 e 4. A figura 10 mostra um exemplo da contagem da largura das barras de um dos grupos do código de barras.

O vetor de larguras resultante é serializado, ou seja, para cada barra preta de largura y são inseridos y '0's, e para cada barra branca de largura y são inseridos y '1's.

O segundo grupo tem codificação fixa é apenas comparado com todas as entradas da tabela do código R (tabela 3) para determinação dos dígitos.

Já o primeiro grupo tem codificação dependente do primeiro dígito. Para tanto, é necessário a decodificação do primeiro dígito com base na paridade dos dígitos do primeiro

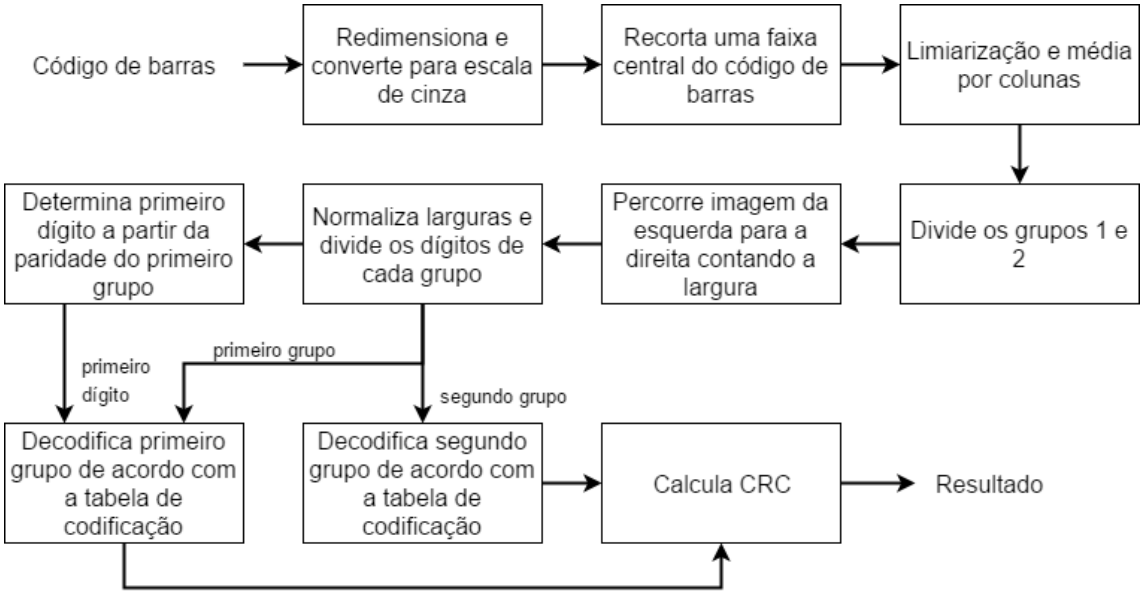


Figura 8: Diagrama de blocos da decodificação.

grupo. Isto é feito contando-se o número de '1's para cada dígito e determinando se este número é par ou ímpar. É gerado um vetor com as paridades, que é comparado com todas as entradas da tabela (tabela 2) para determinação do primeiro dígito.

Uma vez determinado o primeiro dígito, o primeiro grupo é decodificado, comparando com a tabela do código L ou G.

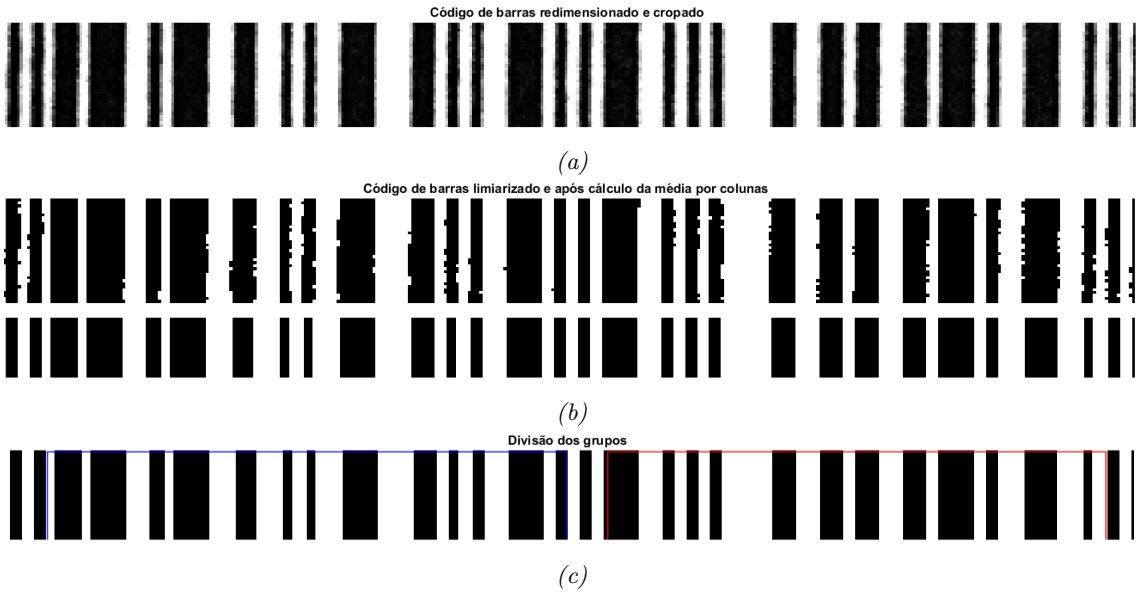


Figura 9: (a) Região de interesse do código de barras. (b) Região limiarizada (superior) e calculada a média de cada coluna (inferior). (c) Divisão dos dois grupos do código de barras.

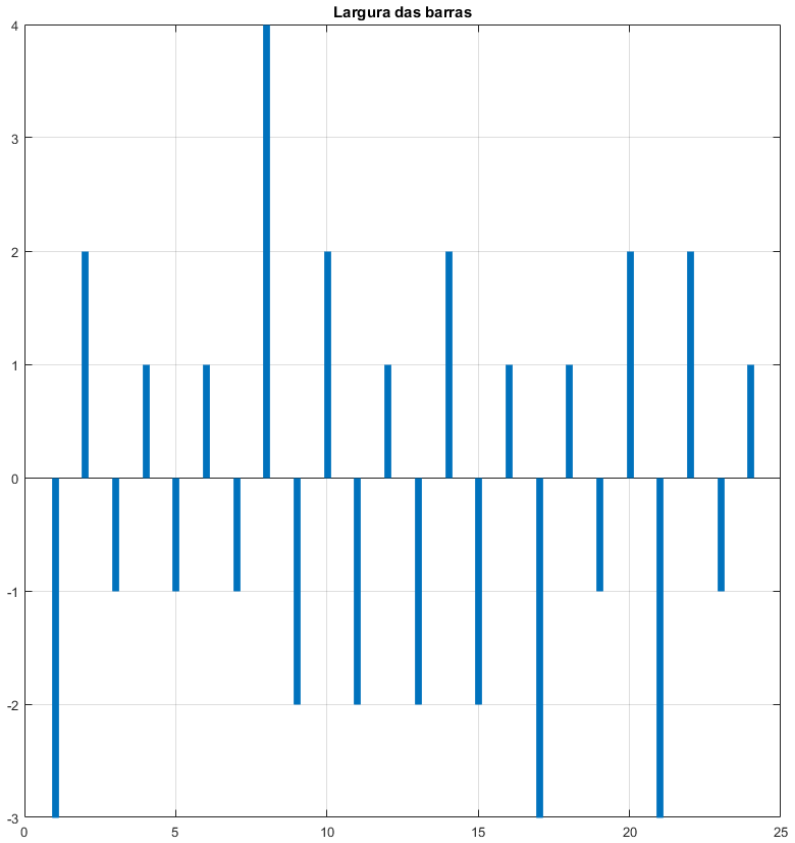


Figura 10: Exemplo de larguras das barras determinadas. Onde um valor positivo significa uma barra branca e um valor negativo significa uma barra preta..

4 Resultados e conclusões

4.1 Resultados da decodificação de imagens artificiais

Para esta etapa foram geradas 220 imagens de códigos de barras a partir de números aleatórios formados por 13 dígitos e com o dígito de CRC válido. O nome do arquivo de cada imagem gerada contém o número equivalente ao código e é usado para fins de validação do algoritmo. Para este conjunto de imagens a taxa de acertos foi de 100%.

4.2 Resultados da decodificação de imagens reais

Nesta etapa foram testados códigos de barras recortados a partir de fotos reais. A taxa de acerto foi 37%. Considera-se um acerto se, e apenas se todos os dígitos decodificados forem iguais aos dígitos esperados. Na maioria dos casos, poucos dígitos foram detectados erroneamente, porém, a média de dígitos errados por código de barras foi de 5.3. Observou-se que o maior problema do algoritmo está na determinação da largura das barras, uma vez que a operação de limiarização da forma como foi utilizada, não é robusta o suficiente para diferentes cenários de iluminação das fotos testadas.

Além disso, não foi levado em conta o fato de que cada dígito possui apenas 4 barras, sendo duas brancas e duas pretas. Esta informação poderia ser aplicada melhorar a robustez do algoritmo de determinação das larguras das barras.

4.3 Resultados da detecção em imagens reais

A detecção encontrou o código de barras corretamente em 74% dos casos.
** POR QUE?
bla bla bla
A figura 11 mostra um exemplo de foto na qual não foi possível encontrar o código de barras.

(a) *(b)* *(c)*

Figura 11: (a) Exemplo de imagem testada na detecção. (b) Etapa intermediária da obtenção da máscara para detecção do código de barras. (c) Etapa resultante da obtenção da máscara para detecção do código de barras.

4.4 Trabalho futuro

A partir dos resultados e conclusões, são listadas possíveis melhorias para serem realizadas em trabalhos futuros:

- Decodificação: Aprimorar a detecção das barras de controle e separação dos grupos.
- Decodificação: Aprimorar a robustez da determinação da largura das barras, o que se mostrou como sendo a maior causa de problemas.
- Decodificação: É possível determinar se o código de barras está invertido, ou seja, se começa da esquerda para a direita, ou da direita para a esquerda. Uma vez detectada a inversão, o código de barras poderia ser decodificado em ambos os casos.
- Detecção: Usar uma estratégia mais robusta para determinar o início e o final exatos do código de barras.
- Detecção: Melhorar correção de rotação e adicionar correção de perspectiva do código de barras.
- Por fim, unir detecção e decodificação e validar o algoritmo em um cenário real.

Referências

[1] International article number.

[2] Oge M. *Practical Image and Video Processing Using Matlab.* John Wiley & Sons, 2011.

[3] MathWorks. imgradientxy - directional gradients of an image, 2017.

[4] MathWorks. regionprops - measure properties of image regions, 2017.