Melhorias do processo de selecionamento de grãos para indústria alimentícia

Felipe Fernandes de Miranda
Departamento de Engenharia da Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal,RN
Email: ffm1994.1@gmail.com

Ormazabal Lima do Nascimento
Departamento de Engenharia da Computação e Automação
Universidade Federal do Rio Grande do Norte
Natal,RN
Email: ormazaball@yahoo.com.br

***Abstract*—The Brazilian industry that invest the most in research is the grain processing industry, its purpose is quite clear: having a competitive product in and out of the country. Thinking about it, this article presents a solution for the processing of beans, which by measuring their size, it is quickly found if the grain is fit to go to the shelves**

***Keywords*—selection of beans, image processing, opencv**

I. Introdução

A Indústria de alimentos é uma das que mais cresce no cenário nacional. Ela emprega aproximadamente 20% dos trabalhadores da indústria de transformação do país. Representa algo em torno de 10% no produto interno bruto (PIB) brasileiro. Estima-se um faturamento anual de R$ 480 bilhes, sendo desses, 52,8% são de fornecedores de café, chá e cereais.

Também é uma das indústrias que mais investe em pesquisa e desenvolvimento no Brasil. Segundo levantamento feito pelo Instituto Brasileiro de Geografia e Estatística (IBGE), o setor é responsável por 15% do total das empresas envolvidas em pesquisa no país. Todo esse investimento tem o objetivo de tornar os produtos produzidos por elas mais competitivos, inclusive no mercado externo.

Todo o investimento em pesquisa que esse setor alavanca e o crescimento que ele proporciona ao país foram as motivações principais que nos impulsionaram á realizar esse trabalho. O objetivo é trazer ao mercado uma forma mais eficiente de seleção de grãos, a fim de melhorar a qualidade do produto final e tornar mais rápida a sua produção.

II. Escolha do OpenCV

O OpenCV foi desenvolvido pela Intel em 2000 e é uma biblioteca para processamento de imagens e vídeo totalmente livre para uso acadêmico e comercial.

Essa biblioteca se mostrou a melhor escolha para a execução desse trabalho, não somente por fazer parte de um padrão aberto, mas principalmente pela sua robustez e facilidade na implementação e no tratamento das imagens. O tempo que a ferramenta é solução no mercado também foi de grande importância nessa decisão.

Diversos algoritmos já estão implementados na ferramenta, como o *FloodFill* que foi utilizado no código e será visto mais a frente. Esses detalhes tornaram a implementação mais rápida e livre de possiveis erros ou de atrasos de uma possivel má

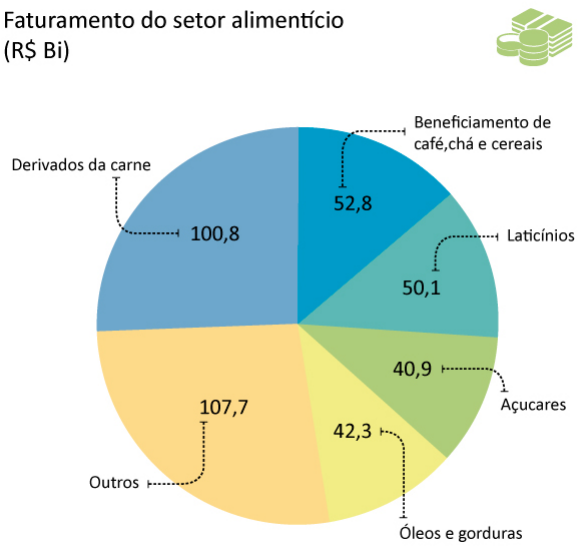


Fig. 1. Faturamento do setor alimentício brasileiro

heuristica, caso fossem implementados pelo programador da solução, visto que precisariamos executar testes para validar esse pedao de código em si. Dessa maneira ganha-se muito em tempo de implementação e processamento.

III. Solução

Uma das caracteristicas que deve ser levado em consideração na escolha de um produto alimentício é o aspecto estético. Tanto que diversas pesquisas vêm sido realizadas nesse nicho, seja para chamar a atenção como no caso da melancia quadrada como também sendo um critério de seleção para a qualidade de frutas, grãos, entre outros.

Pensando nisso foi desenvolvida uma solução para seleção de feijões que baseado no tamanho da amostra e no desvio da média dela indica quais grãos estão bons o suficiente para serem comercializados. O produtor tem total liberdade para escolher quais os parâmetros para considerar um feijão apto para venda e o software tratará do resto.

O algoritmo foi implementado usando OpenCV e devido a essa ferramenta pode-se obter tamanha velocidade nessa seleção, já que é preciso apenas uma foto das amostras e depois

de passado os parâmetros o programa precisará percorrer a imagem para chegar á um resultado que indicará os feijões que deverão ou não entrar na fase de comercialização.

IV. PREPARAÇÃO DO EXPERIMENTO

Para que o algoritmo funcione corretamente deve se ter alguns cuidados na preparação da imagem. Os testes realizados foram executados somente em feijões pretos. Neles foram tiradas fotos dos feijões em cima de uma superficie branca, com o objetivo de não existir confusão entre as cores dos feijões e a mesa e assim dificultar na varredura que o algoritmo fará.

A iluminação também influi no funcionamento do programa, pois sombras podem gerar confusão no algoritmo. É recomendável manter os grãos separados, de modo que a olho nú se verifique essa separação. Mais á frente veremos como funciona o algoritmo de contagem de grãos e como esses fatores são essenciais para que a contagem seja feita corretamente.

V. IMPLEMENTAÇÃO

Ao iniciar o algoritmo, o usuário deverá passar como argumento o endereço da imagem que contém a amostra de feijões. Para uma melhor análise, logo após receber a imagem, há a conversão da foto para preto e branco e depois a transformação dela em uma imagem binária.

```
Image = imread(argv[1],CV_LOAD_IMAGE_GRAYSCALE);
threshold(image, image, 60, 255, CV_THRESH_BINARY);
```

A função *threshold* responsável pela conversão binária funciona obedecendo a seguinte lógica: Quando o pixel ultrapassar a cor T, que na implementação corresponde á 60, o programa o pintará da cor maxValue, que no código será 255, correspondente á cor branca; caso contrário ele pintará de 0, que é a cor preta.

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{maxValue} & \text{if } \mathtt{src}(x,y) > \mathtt{T}(x,y) \\ 0 & \text{otherwise} \end{cases}$$

A conversão binária é um ponto importante pois facilita a identificação, nos próximos passos, de quais pixels são feijões ou não na imagem. Devido á isso também é importante que os feijões estejam dispostos em uma superfcie branca, a fim de facilitar a identificação deles pelo algoritmo.

Na parte seguinte identificamos quantos feijões existem na amostra usando o algoritmo *FloodFill*. Trata-se de um algoritmo de contagem de objetos através de rotulação. É realizada uma busca por toda a imagem por pixels de cor preta, dos quais sabemos indicam que ali existe um feijão. O algoritmo o marca com uma flag *nobjects* que indica o numero do feijão.

Depois disso ele sai procurando os 8 vizinhos desse ponto e os marca com a mesma flag. Como explicado na fig2, os 8 vizinhos são compostos por todos os pixels ao redor dele, incluindo as direções direita, esquerda e diagonais. Nesse passo, encontrar um pixel que não seja cor preta indica que os limites daquele feijão terminaram e que a variável *nobjects* deve ser incrementada para marcar o próximo feijão á ser identificado.

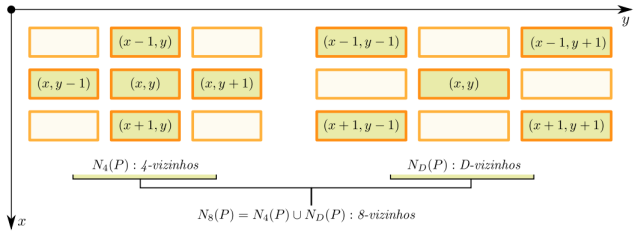


Fig. 2. Identificação de 4 ou 8 vizinhos de um pixel

Por esse motivo os feijões devem estar bem separados na imagem. Caso dois feijões estejam juntos o suficiente para que não seja notada a linha de separação, o *FloodFill* os contará como sendo o mesmo feijão, influindo drasticamente no resultado final do programa.

```
nobjects=0;
  for(int i=0; i<height; i++)
  {
    for(int j=0; j<width; j++)
    {
      if(image.at<uchar>(i,j) == 0)
      {
      p.x=j;
      p.y=i;
      floodFill(image,p,nobjects++);
      }
    }
  }
```

Após esse passo há uma verificação dos feijões identificados. Isso se torna necessário pois podem haver rudos na imagem processada, por isso foi estipulado que se for identificado um objeto com 1000 ou menos pixels se trata provavelmente de um erro na imagem e o objeto identificado não será incluído na contagem.

O primeiro passo para fazer essa verificação é passar por todos os objetos identificados como feijões e armazenar quantos pixels cada um deles tem.

```
int vetor[nobjects];
  int numPixelsN = 0;
  for (int n = 0; n < nobjects; n++)
  {
    for(int i=0; i<height; i++)
    {
      for(int j=0; j<width; j++)
      {
         if(image.at<uchar>(i,j) == n)
         {
            numPixelsN++;
         }
      }
    }
    vetor[n] = numPixelsN;
    numPixelsN = 0;
  }
```

Depois percorremos esse *array* e finalmente verificamos se é ou não ruído com base no teste descrito anteriormente. Se não for ruído, o contador *numFeijoes* incrementa, caso contrário o ruído será pintado com a cor branca, para coincidir com a já recomendada superfície em que se encontram os feijões.

```
int numFeijoes = 0;
  for (int n = 0; n < nobjects; n++)
  {
    if(vetor[n] < 1000)
    {
      for(int i=0; i<height; i++)
      {
        for(int j=0; j<width; j++)
        {
           if(image.at<uchar>(i,j) == n)
           {
              image.at<uchar>(i,j) = 255;
           }
        }
      }
    }
    else
    {
      numFeijoes++;
    }
  }
```

Retirado o ruído, os passos anteriores são repetidos. É feita outra varredura na imagem e nela os feijões são detectados usando *FloodFill*, depois o tamanho de cada um deles é armazenado num *array*. Finalmente o programa retorna para o usuário o numero de feijões identificados, o tamanho de cada um deles e a média de tamanho dos feijões considerando toda a amostra.

```
int coresDosFeijoes = 0;
  for(int i=0; i<height; i++)
  {
    for(int j=0; j<width; j++)
    {
      if(image.at<uchar>(i,j) == 0)
      {
        coresDosFeijoes++;
        p.x=j;
        p.y=i;
        floodFill(image,p,coresDosFeijoes);
      }
    }
  }

  int tamanhoDosFeijoes[coresDosFeijoes];
  for (int n = 1; n <= coresDosFeijoes; n++)
  {
    for(int i=0; i<height; i++)
    {
      for(int j=0; j<width; j++)
      {
        if(image.at<uchar>(i,j) == n)
        {
          tamanhoDosFeijoes[n]++;
        }
      }
    }
  }
```

Por fim, o usuário deve informar qual a porcentagem de desvio da média calculada do tamanho dos feijões que fará com que eles sejam aceitos no teste de qualidade. Após isso, o algoritmo retornará o tamanho mínimo e máximo, o qual o feijão deve estar nesse intervalo para passar no teste, assim como uma imagem com marcações verdes, indicando os feijões que passaram no teste, e vermelhas, indicando os feijões que foram reprovados.

```
int valorMin = media - (desvio/100)*(media);
int valorMax = media + (desvio/100)*(media);
```

A imagem retornada para o usuário é colorida e para isso utiliza-se três vetores representando RGB. O algoritmo de preenchimento irá percorrer toda a imagem original, e caso o pixel seja branco, pintará de branco a imagem final; caso contrário irá verificar se ele está dentro dos limites de tamanho estipulado, se sim pintará de verde, senão pintará de vermelho.

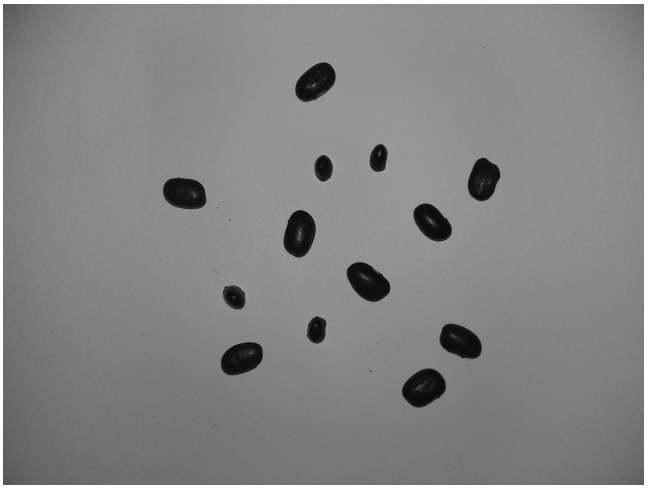


Fig. 3. Primeira amostra de feijões usada como entrada

```
for(int i=0; i<height; i++)
  {
    for(int j=0; j<width; j++)
    {
      if(image.at<uchar>(i,j) == 255)
      {
        resultado.at<cv::Vec3b>(i, j)[2] = 255;
        resultado.at<cv::Vec3b>(i, j)[1] = 255;
        resultado.at<cv::Vec3b>(i, j)[0] = 255;
      }
      else
      {
     int tamanho=tamanhoDosFeijoes[image.at<uchar>(i,j)];
        if(tamanho <= valorMin || tamanho >= valorMax)
        {
          resultado.at<cv::Vec3b>(i, j)[2] = 255;
          resultado.at<cv::Vec3b>(i, j)[1] = 0;
          resultado.at<cv::Vec3b>(i, j)[0] = 0;
        }
        else
        {
          resultado.at<cv::Vec3b>(i, j)[2] = 0;
          resultado.at<cv::Vec3b>(i, j)[1] = 255;
          resultado.at<cv::Vec3b>(i, j)[0] = 0;
        }
      }
    }
  }
```

VI. RESULTADOS

Nessa parte do relatório apresentaremos testes feitos com o programa com o objetivo de provar a validade da heuristica escolhida. Partiremos da primeira parte que trata da contagem dos feijões.

Usando como entrada a fig 3, o algoritmo conseguiu contar corretamente os treze feijões e retornou um tamanho médio de 26678 pixels da amostra. A porcentagem escolhida para o selecionamento dos feijões foi de 50%, que faz com que os escolhidos tenham tamanhos variando entre 13339 e 40017 pixels.

Fig. 4. Saída do programa após teste de qualidade da primeira amostra

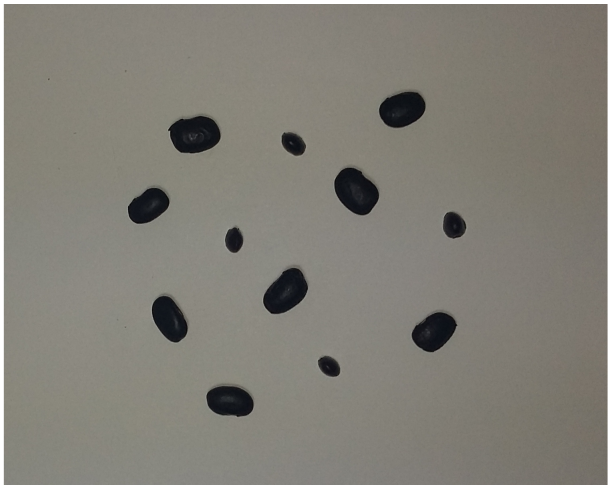


Fig. 5. Segunda amostra de feijões usada como entrada

```
feijao 1: 27531
feijao 2: 13769
feijao 3: 41889
feijao 4: 27723
feijao 5: 28203
feijao 6: 24869
feijao 7: 28717
feijao 8: 31534
feijao 9: 9921
feijao 10: 12033
feijao 11: 27299
feijao 12: 42285
feijao 13: 31052
tamanho medio dos feijoes: 26678
```

Se observarmos, o tamanho entre a maioria dos feijões varia muito pouco, fazendo com que muitos deles estejam bem próximos da média. Assim, até com uma porcentagem tão alta como a escolhida, a maior parte da amostra passará no teste de qualidade, como pode ser comprovado na fig4.

Na próxima amostra temos outro numero de feijões disposto de maneira diferente da primeira. Como esperado o algoritmo executa a contagem sem dificuldades, visto que foram observados os critérios recomendados de separação entre os feijões e o uso da superfície branca no fundo da imagem.

Esse teste terá caráter de validação do algoritmo, por isso usando a mesma entrada, geraremos várias saidas apenas variando os parâmetros do algoritmo. A contagem dos feijões, como esperado, aconteceu sem erros.

```
numero de feijoes: 12
feijao 1: 40556
feijao 2: 9720
feijao 3: 2416
feijao 4: 9161
feijao 5: 6381
feijao 6: 3795
feijao 7: 2053
feijao 8: 9434
feijao 9: 6928
feijao 10: 42090
feijao 11: 1941
feijao 12: 7081
tamanho medio dos feijoes: 11796
```

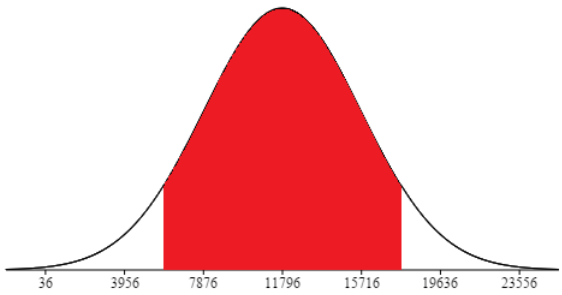
Nos testes que se sucedem foram aplicados desvios de 50%, 70% e 90%. Como sabemos, quanto maior o desvio, mais feijões passaram no controle de qualidade. Os resultados foram:

Desvio: 50% Minimo: 5898 Máximo:17694

Desvio: 70% Minimo: 3538 Máximo: 20053

Desvio: 90% Minimo: 1179 Máximo: 22412

Como foi observado já no primeiro experimento, a maioria dos feijões da amostra tem tamanhos similares e ficam com valores um tanto próximos da média. Quanto mais nos afastamos da média, menos feijões são contados, entretanto pode-se notar que não se trata de uma queda linear. Nesse caso, a função que melhor representará a variação dos feijões será uma gaussiana.



VII. ALGUMAS CONSIDERAÇÕES

Acreditamos que para um primeiro momento, a métrica utilizada pelo algoritmo foi suficiente para trazer bons resultados. Obviamente, além do tamanho, outras caractersticas do feijão poderiam ter sido coletadas como sua forma e sua coloração. Essas medidas ficam como sugestão para trabalhos futuros ou para uma próxima implementação desse mesmo programa.

A solução se mostrou extremamente rápida diante dos testes realizados. A biblioteca OpenCV contribuiu muito para isso, visto que já existem algoritmos prontos para uso nela como o *FloodFill*, e esse fator contribuiu muito para a implementação desse programa num tempo razoável.

Apesar de eficiente para nossos testes, sabemos que numa indústria em que a produção é massiva, se faz necessário

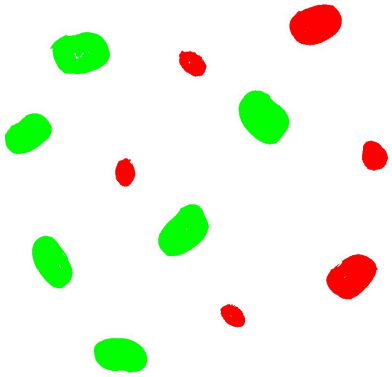


Fig. 6. Saída do programa com desvio de 50%

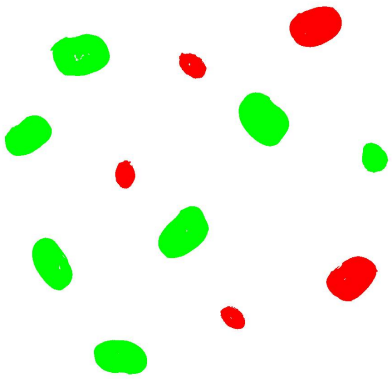


Fig. 7. Saída do programa com desvio de 70%

amostras com quantidades de feijões bem superiores ás testadas. Nesse ponto o algoritmo em si pode necessitar de melhorias em sua implementação.

Considerando H como altura e W como largura, uma varredura numa imagem completa tem complexidade T(HW). No código foram feitas varreduras completas na imagem para os seguintes objetivos:

Pintar os pontos diferentes de branco

Salvar o tamanho de pixels que representam cada cor

Contagem dos feijões e verificação de ruído

Pintar os feijões na imagem de saída

Ao final do programa serão realizadas 7 varreduras para que seja apresentado o resultado, sendo algumas delas de complexidade T(*quantidadedefeijoes*HW). Para uma quantidade de feijões ou uma imagem de tamanho considerável, deverá

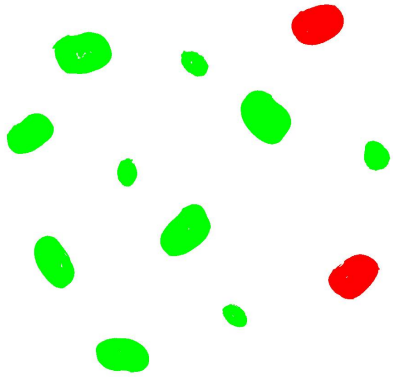


Fig. 8. Saída do programa com desvio de 90%

existir uma demora no processamento do resultado, entretanto melhorias não são difíceis de serem realizadas, e essa demora só será visível em casos extremos de entrada.

VIII. CONCLUSÃO

O algoritmo teve um bom funcionamento para o que foi proposto. A identificação e o selecionamento dos feijões ocorre sem problemas, caso sejam seguidas as recomendações de preparação da fotografia.

A seleção automatizada dos grãos e a rapidez no processamento da resposta do programa parecem ser de grande interesse dos usuários do setor. Por isso acreditamos na eficácia desse produto, e que embora numa fase inicial, consegue realizar a seleção de modo satisfatório.

IX. AGRADECIMENTOS

Gostaríamos de agradecer a oportunidade de realizar esta pesquisa e poder utilizar os conhecimentos adquiridos na disciplina em uma solução que pode ser aplicada ao cotidiano de engenheiros e da sociedade em geral.

As técnicas de processamento de imagens são de extrema valia para o leque de ferramentas que nós, futuros engenheiros, poderemos utilizar na solução de problemas. Por isso essa experiência está sendo tão importante para nossas carreiras.

REFERENCES

[1] Agostinho Brito Jr, *http://www.dca.ufrn.br/ ambj/dca0445-projetos.html*, [Acesso em 03 junho,2016]

[2] *http://www.portaldaindustria.com.br/cni/iniciativas/programas/brazil-4-business/2014/09/1,60192/alimentos-e-bebidas.html*, [Acesso em 03 junho,2016]

[3] *http://www.gazetadopovo.com.br/economia/setor-de-alimentos-investe-muito-mas-ainda-inova-pouco-ec9ehr0phnhrb6p3gsw3bpfta*, [Acesso em 03 junho,2016]

[4] Agostinho Brito Jr, *http://agostinhobritojr.github.io/tutoriais/pdi/*, [Acesso em 03 junho,2016]

[5] *http://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous$_{t}rans formations$*, [