

Detecção de Objetos com Python e OpenCV

Jones Granatyr

Gabriel Alves



Conteúdo do curso

- Teoria básica sobre haarcascades
- Detecção de faces
- Detecção de canecas (3 experimentos)
- Detecção de logos
- Tópicos complementares

Pré-requisitos

- Lógica de programação (if e for)
- Conhecimentos básicos sobre Python são desejáveis
- Conhecimentos sobre detecção de faces são desejáveis
- Nível do curso: iniciante

O que não veremos

- Detalhes muito teóricos sobre haarcascade (foco na prática)
- Aplicativo comercial ou interface gráfica

OpenCV

- OpenCV
 - Fazer o download do OpenCV com instalador para Windows (Win pack)
 - Executar e extrair os arquivos
 - Na pasta extraída, localizar /build/x64/vc15/bin
 - Copiar para a pasta do projeto os arquivos: opencv_createsamples.exe, opencv_traincascade.exe, opencv_world331.dll
- Python
- Pycharm
- Cmder

OpenCV

- Biblioteca mais popular para detecção de faces
- OpenCV = Open source computer vision
- Intel 1999
- Escrita em C/C++
- <https://opencv.org/>

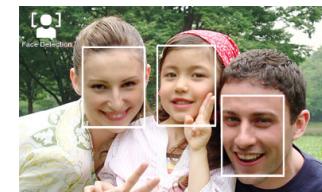
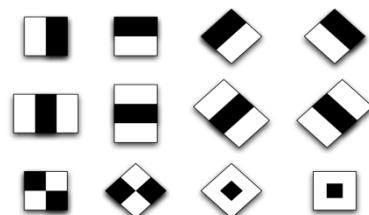
Teoria básica sobre haarcascades

Classificador Cascade



Treinamento
com AdaBoost

Seleção das
características



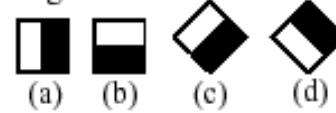
Componentes

- Haar cascades
- AdaBoost
- Cascade

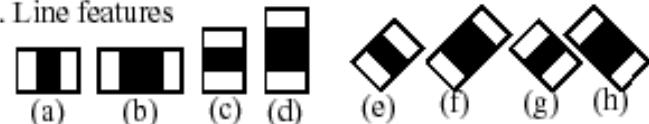
Haar cascades

- Combinação de features haar para formar um classificador
- Padrão retangular nos dados
- Diferenças na intensidade das regiões retangulares da imagem

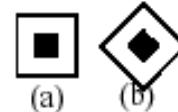
1. Edge features



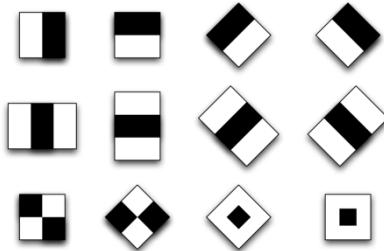
2. Line features



3. Center-surround features



Haar cascades



Soma pixels brancos –
soma pixels pretos

Mais de 160.000
combinações em uma
imagem 24 x 24!



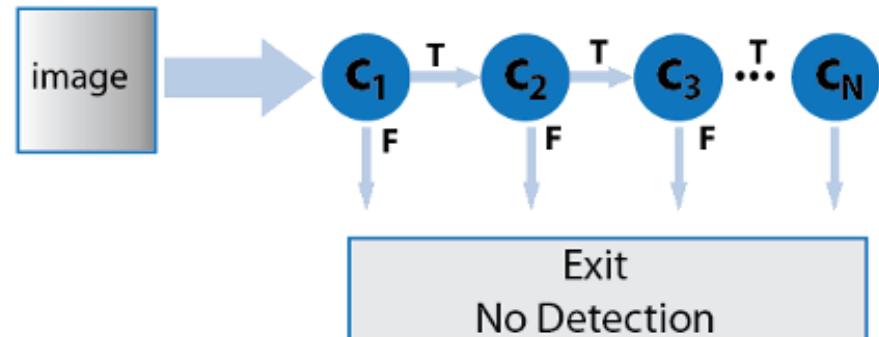


AdaBoost

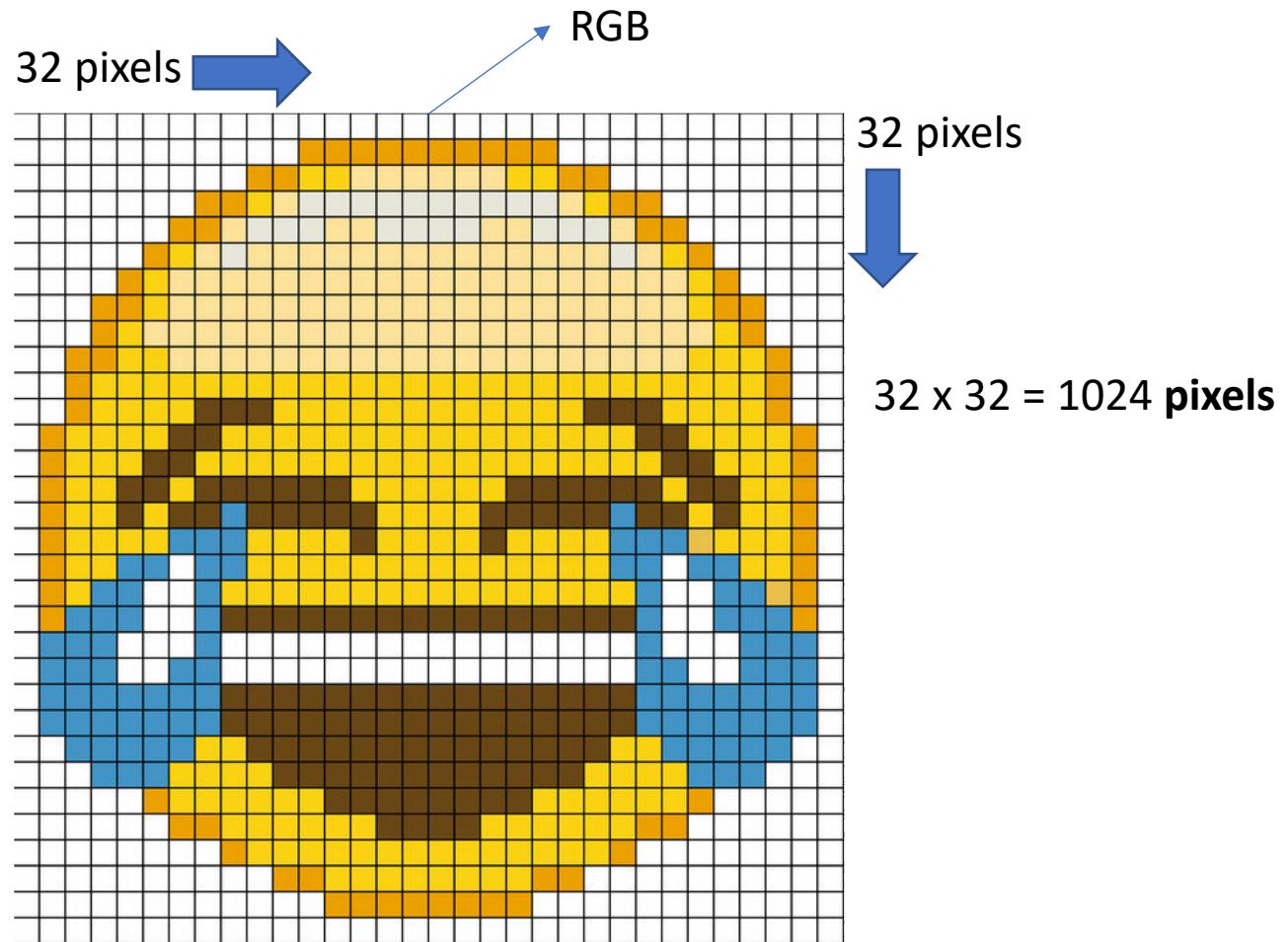
- AdaBoost remove as características não necessárias
- Combina vários classificadores fracos em um classificador forte

Cascade

- Desliza pela imagem
- Computa a média dos valores dos pixels na área branca e preta
- Se a diferença entre as áreas é abaixo de um limiar, a característica coincide (match)
- Aprendizagem supervisionada

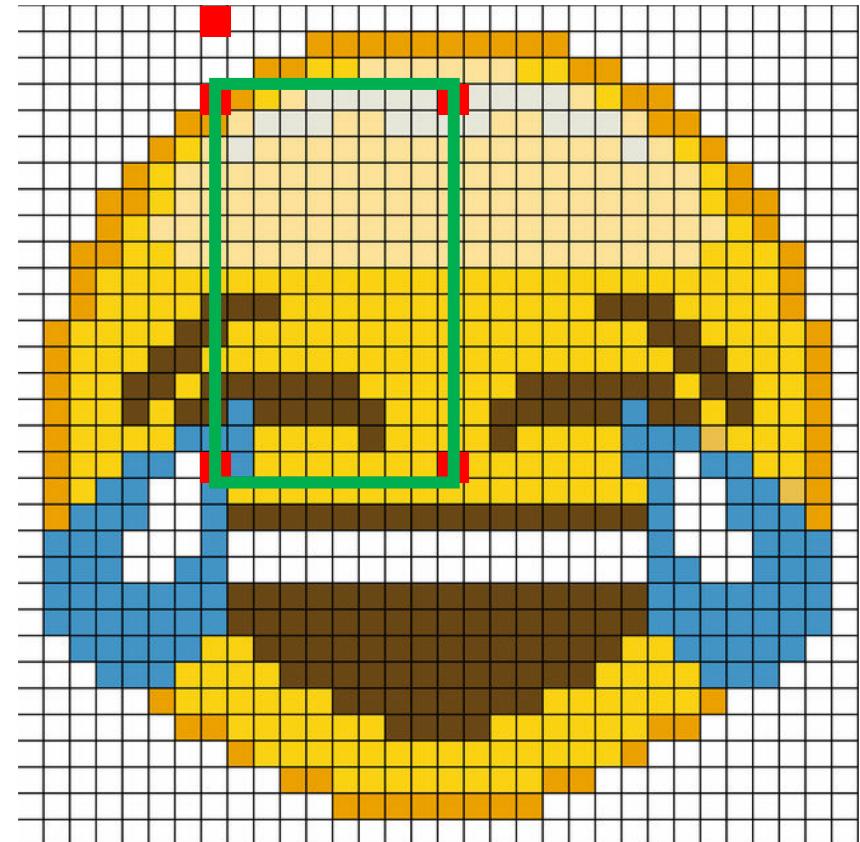


Pixels



Bounding box (caixas delimitadoras)

- Haarcascade (OpenCV)
 - Left, top, width, height
 - 8, 4, 10, 15



Parâmetros cascadeClassifier OpenCV

Scale factor

- Quando objetos estão perto da câmera, eles serão maiores que os objetos ao fundo da imagem
- Especifica quanto o tamanho da imagem é reduzido em cada escala de imagem
- Redimensionar um objeto maior para um menor
- Mais lento se o valor for menor

minNeighbors

- Quantos vizinhos cada retângulo candidato deve ter para mantê-lo
- Valores altos
 - Menos detecções, porém apresenta maior qualidade

minSize

- Especifica o menor objeto a ser reconhecido
- 30×30 é o valor padrão

maxSize

- Especifica o tamanho máximo de um objeto
- Exemplo: um objeto grande na tela

Criação de haarcascades personalizados

Etapas para criação de haarcascade personalizado



Etapa 1 – Escolher o objeto

- Objetos mais rígidos (como uma logo) ou com grandes variações (cadeiras, copos, celulares)
- Objetos que possuem uma quantidade muito grande de variações vão exigir uma grande quantidade de imagens para o treinamento
- Ao treinar tantas variações pode ser que o classificador fique mais fraco como um todo
- Objetos que a cor é um fator fundamental para diferenciar também não são recomendados

Etapa 2 – Selecionar imagens negativas

- Imagens que não contém o objeto a ser reconhecido
- De preferência devem ser maiores que as imagens positivas, especialmente se vão ser criadas amostras
- Podem ser fotos de qualquer outra coisa



Etapa 2 – Selecionar imagens negativas

- Fotos de prováveis fundos do objeto a ser detectado (não é obrigatório)
- Ainda assim recomenda-se usar também imagens de outros objetos aleatórios



Etapa 3 – Selecionar imagens positivas

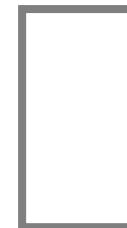
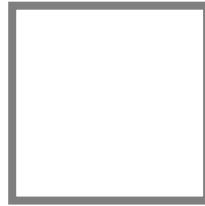


Quantas imagens positivas?

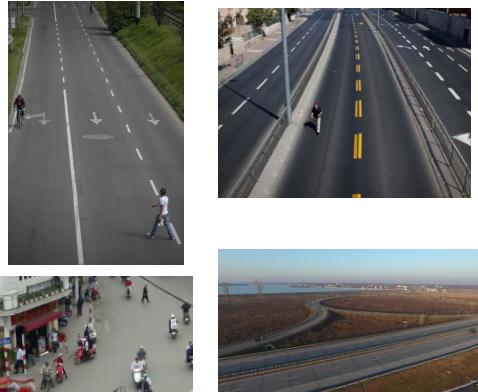
- Depende
 - Qualidade das imagens
 - Tipo do objeto a ser reconhecido
 - Poder computacional disponível

Tamanho das imagens

- Definir um tamanho único para todas as imagens
- Não precisa ser necessariamente quadrada, a largura e altura podem ser diferentes (exemplo 25x15)
- O recomendado é que o maior lado não ultrapasse 30px para não elevar muito o tempo de treinamento (para nossos exemplos recomendamos no máximo 25px)
- A proporção precisa ser a mesma para todas as imagens (caso contrário o OpenCV redimensionará automaticamente, o que pode não ser o ideal)
- Caso seja necessário treinar com imagens maiores para captar muitos detalhes o processamento pode chegar a levar dias ou semanas
- 25x25 (proporção 1:1) 24x20 (1:1.2) 20x10 (2:1) 10x20 (1:2)



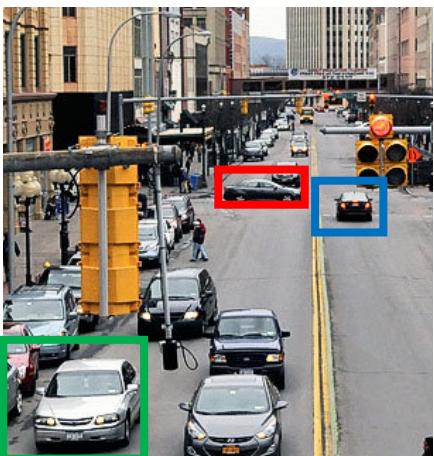
Classificador A



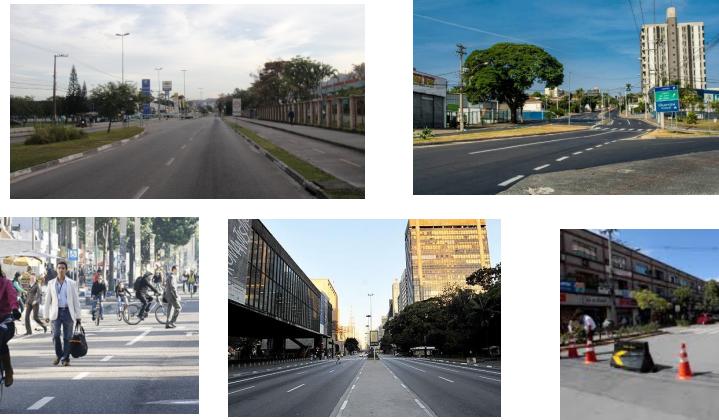
Negativas



Positivas



Classificador B



Negativas



Positivas

- Provável que apenas o classificador A reconheça
- Provável que os dois classificadores reconheçam
- Talvez o classificador B reconheça mas o A deve reconhecer melhor por ser mais semelhante às imagens positivas (pois são de um ângulo mais de cima)

Etapa 3 – Selecionar imagens positivas

- As imagens positivas devem estar de acordo com o objetivo do classificador
- Detectar carros usando câmeras de segurança
 - As imagens de carros devem ser tiradas de um ângulo de cima
- Detectar carros da visão de um pedestre
 - As imagens de carros devem ser tiradas de frente

Etapa 3 – Selecionar imagens positivas (carro perspectiva pedestre)

- Adicionar imagens de carros vistos de lado. Porém, todas as imagens positivas serão redimensionadas para um tamanho único na etapa de treinamento
- Problema: as imagens de carros de frente ou de trás tendem a ser mais quadradas
- As imagens de carros de lado serão com uma outra proporção, não serão tão “quadradas” e serão mais retangulares pois terão uma largura maior e altura menor em comparação
- Todas as imagens positivas devem ter a mesma proporção de tamanho (50x50, 100x100, 25x25, por exemplo)
- Na etapa de preparação de treinamento todas serão redimensionadas para ter o mesmo tamanho. Se o tamanho definido for 25x25 então não vai ter problemas com uma imagem 100x100, mas se tiver uma imagem mais retangular o OpenCV achatará a imagem (ou alongar, caso seja o contrário), descaracterizando o objeto
- Solução: criar dois classificadores distintos: um para carros de frente/trás e outro para carros de lado



Redimensionada
de 100x50 para
25x25

Tempo de treinamento

- Tamanho da imagem
- Quantidade de imagens positivas e negativas
- Estágios
- Memória disponível
- Processamento

Ângulos

maxxangle



maxyangle



maxzangle



Compressão

Example of Lossy Compression



**Original Lena Image
(12KB size)**



**Lena Image,
Compressed (85%
less information,
1.8KB)**



**Lena Image, Highly
Compressed (96%
less information,
0.56KB)**

Fonte: <https://tylerbrownblog.wordpress.com/2016/06/16/jpg-mp3-mpegpdf-and-lossy-compression/>

Estágios

- Quanto mais estágios teoricamente mais preciso será o classificador, porém será mais demorado
- Mesmo aumentando o número de estágios o treinamento pode terminar em um estágio anterior ao determinado. Significa que o treinamento atingiu uma boa taxa de acertos ou que a quantidade de imagens positivas não é o suficiente
- Pode ocorrer overfitting com muitos estágios
- Menos estágios para menos imagens
- Quando o conjunto de imagens positivas é pequeno e é especificado uma taxa de acerto alto, são necessário mais estágios para alcançar essa taxa (pode ocorrer o overfitting)
- Ao aumentar o número de imagens positivas a generalização do modelo também aumentará, pois procurará por melhores características que descrevam o objeto de forma mais abrangente (encontrar o objeto mesmo com variações)
- Falsos positivos: quando o detector diz que o objeto está na imagem quando na verdade não está
- Falsos negativos: quando a imagem contém o objeto porém ele não é detectado

Estágios

- Hit Rate (HR)
 - Taxa de acerto mínima desejada para cada estágio
 - Possível controlar o HR com o parâmetro minHitRate
 - O valor 0,998 significa que o algoritmo pode cometer 2 erros de 1000 naquele estágio
- False Alarm (FA)
 - Porcentagem de falsos positivos permitidos em cada estágio
 - Possível controlar o FA com o parâmetro maxFalseAlarmRate
 - Se o valor diminuir o algoritmo treinará mais vezes
 - Se o valor for 0 pode ser impossível terminar o treinamento
- HT alto e FA baixo

Melhorias

- Não há uma regra sobre a quantidade de imagens positivas ou negativas
- Primeiro teste: dobro de positivas em relação as negativas
- Se houver muitos falsos positivos, adotar a regra inversa
- Regras gerais
 - Se há muitos falsos positivos é preciso adicionar mais imagens negativas
 - Se há muitos falsos negativos é preciso adicionar mais imagens positivas ou ajustar os parâmetros
- Para um classificador mais robusto: coletar imagens manualmente e em vários cenários
- Para objetos mais rígidos (logos) pode-se usar somente o createsamples

Melhorias

- Ao aumentar o número de imagens positivas vai aumentar a generalização do modelo (melhores características que descrevam o objeto)
- Em imagens nas quais o objeto está um pouco diferente do modelo treinado o algoritmo tem mais chances para encontrar o objeto
- Evita o sobreajuste (overfitting) do modelo, que ocorre quando o modelo se ajusta tão bem a esse conjunto de imagens de treinamento mas é ineficaz para prever novos resultados

Melhorias

- -minHitRate 0.999 -maxFalseAlarmRate 0.1 -maxWeakCount 1000
 - -minHitRate = quanto mais perto do 1 melhor
 - -maxFalseAlarmRate = quanto menor o valor melhor
 - -maxWeakCount = quanto maior o valor melhor
- O tempo de treinamento pode ficar extremamente lento, principalmente quando começa a chegar nos últimos estágios. Caso o número de imagens positivas e/ou negativas seja maior que 5000 o treinamento poderá levar dias ou até mesmo semanas
- Não há garantias!! As imagens positivas precisam ter uma alta qualidade
- Treinamento com LBP (muitas imagens e mais rápido)

-bgthresh

- Determina a quantidade de tolerância que a cor no parâmetro -bgcolor tem para ser transformada em transparência
- Todos os pixels dentro do $\text{bgcolor} - \text{bgthresh}$ e $\text{bgcolor} + \text{bgthresh}$ serão interpretados como transparente
- Se definirmos a cor branca como $-\text{bgcolor}$ (255), quanto maior esse valor mais pixels "próximos ao branco" vão ficar transparentes

-bgthresh 0



+

+

Se aumentar mais...



-inv e -randinv

- -inv = inverte a cor da imagem
- -randinv = inverte randomicamente (algumas estarão na cor original e outras na cor invertida)
- Pode ser útil na criação de classificadores de logos, pois a logo as vezes pode conter o símbolo na cor preta ou branca (variabilidade maior)



Parâmetros do traincascade

-minHitRate [valor padrão: 0.995]

- Taxa de acerto mínima desejada para cada estágio do classificador
- Resultados melhores com valores maiores, ou seja, quanto mais próximo do 1 melhor
- Um bom valor usado para testes é 0.998

-maxFalseAlarmRate [valor padrão: 0.5]

- Taxa máxima de alarme falso desejado para cada estágio do classificador (definição de quantos recursos precisam ser adicionados)
- A ideia é que cada classificador fraco tenha uma taxa de acertos muito boa nos pontos positivos e, em seguida, permita que eles removam “janelas” negativas o mais rápido possível com uma adivinhação melhor do que a aleatória
- O valor 0.5 significa que é aplicado um palpite aleatório. Melhor do que isso significa que são removidas sucessivamente as janelas negativas como negativas muito cedo, usando apenas algumas avaliações de recursos, permitindo que outros negativos sejam descartados pelos estágios seguintes
- Se diminuir o valor vai produzir classificadores fracos (weak classifiers) maiores e também mais recursos a serem avaliados em mais janelas antes do algoritmo tomar uma decisão inicial naquela janela e passar para próxima
- Quanto menor teoricamente melhor será o classificador

Considerações haarcascades

- Executa rápido (vários detectores pela webcam)
- Recomendável para objetos com pouca variação (logos) – poucas imagens para o treinamento
- Não muito recomendável para objetos com uma grande variação
 - Celular ligado, com capinha
 - Treinando muitas variações o classificador pode ficar fraco
 - Objetos que a cor é determinante
 - Canecas com estampa diferente
 - Necessárias muitas imagens e muitos testes
- Outras técnicas
 - HOG + SVM (Dlib)
 - Redes neurais convolucionais

Coletas de imagens positivas

Como coletar imagens positivas

- Imagens de repositórios e bases de imagens da internet -
<http://image-net.org/>
- Criação de imagens a partir do método createsamples
- Fotos já existentes
- Busca direta e manual de imagens (sites de pesquisa)

Imagens positivas selecionadas manualmente

- A qualidade das imagens impacta na qualidade da detecção
- Imagens com variação na iluminação e com fundos diversos
- Recomenda-se pelo menos 1.000 imagens positivas (ideal é em torno de 5.000 imagens)
- Lightshot - <https://prnt.sc/>

Pré-processamento das imagens

Tratamento das imagens

- Softwares para edição de imagens, como GIMP ou Photoshop
- Ferramentas online
- Dar preferência para imagens que o fundo esteja totalmente branco e com iluminação para identificar bem o objeto
- Imagens que contenham somente o objeto, pois assim não será necessário realizar outros pré-processamento

Imagen original

- Fundo branco
- Remover a sombra



Aumento do brilho

- Aumentar o brilho com o valor 30
- Não aumentar muito o brilho, pois o contraste ainda precisa ser aumentado



Aumento de contraste

- Aumentamos o contraste até o fundo ficar branco
- Aumentar o contraste para tirar a sombra: pode descaracterizar a imagem (pixels claros mais claros e pixels escuros mais escuros)



Removendo o restante

- Se a imagem ainda contiver alguma coisa que não seja o objeto pode-se usar a ferramenta borracha



Conclusão

