INF394 - Processamento Digital de Imagens

Trabalho Prático 3 - Versão 1

Prof. Marcos Henrique Fonseca Ribeiro

Junho/Julho de 2017

1 Observações Gerais Sobre o Trabalho

1. Formato de entrega: arquivo *.zip*, contendo os arquivos pedidos ao longo do roteiro.
2. Meio de entrega: PVANet
3. Equipe: mesma do primeiro trabalho
4. Valor da atividade: 50 pontos

2 Parte I

Nesta primeira parte, deve-se utilizar a estrutura básica de programa com interface gráfica disponibilizada no PVANet ou a linguagem/tecnologia que preferir para desenvolver um software com as funcionalidades especificadas nas seções a seguir, valendo as seguintes observações **importantes:**

- Caso não vá utilizar uma linguagem multiplataforma, como Python ou Java, disponibilize juntamente com os arquivos entregues as instruções para instalação, compilação ou qualquer outra que for necessária para que uma pessoa consiga instalar e executar o produto final em Windows, Linux ou Mac.
- **Está vetado** o uso de qualquer biblioteca, módulo pronto, framework ou coisa que o equivalha que já implemente os algoritmos e técnicas de Processamento Digital de Imagens propriamente dito como, por exemplo, *OpenCV*. Você deve implementar estes recursos.
- **Está liberado** o uso de qualquer biblioteca, módulo, framework etc. que não seja para a finalidade descrita acima. Por exemplo, esteja à vontade para utilizar bibliotecas para manipulação de interface gráfica, manipulação de arranjos, matemática, gerenciamento de arquivos no SO etc.
- A **exceção** à regra acima está no uso de soluções para a carga de uma imagem a partir de um arquivo para um objeto, estrutura de dados ou

afins; para a visualização de imagens e para o salvamento das mesmas. Sendo assim, por exemplo, pode-se utilizar o módulo Python PIL (*Python Image Library*) no trabalho para ler dados de arquivos, salvar arquivos, exibir imagens na tela. Porém não se poderá utilizar os recursos já presentes na mesma no que diz respeito ao processamento da imagem como, por exemplo, a conversão para uma imagem binária.

Por fim, apesar de não ser obrigatório, recomenda-se a utilização da estrutura software desenvolvida para o primeiro trabalho. Porém, para fins de avaliação, será considerada só a parte que disser respeito ao conteúdo da disciplina, isto é, processamento de imagens, e for referente ao conteúdo novo do trabalho. Para fins de simplicidade, envie o resultado final completo, incluindo o conteúdo do primeiro trabalho, se for o caso. A especificação do trabalho encontra-se na Seção 2.1.

2.1 Funcionalidades a serem Implementadas

As funcionalidades descritas aqui seguem uma sequência lógica didática. Esteja livre para enquadrá-las na organização de menus da interface do software da forma que bem desejar.

- Filtro de Butterworth, usando domínio da frequência (para o filtro, especificar o raio de suavização)
- Filtro de passa-baixa, usando domínio da frequência (para o filtro, especificar o raio de corte)
- Filtro de passa-alta, usando domínio da frequência (para o filtro, especificar o raio de corte)
- Filtro de passa-faixa, usando domínio da frequência (para o filtro, especificar os dois raios de corte)
- Limiarização de Otsu
- Limiarização Global Adaptativa
- Limiarização por Média Móvel
- Quantização para k níveis de cor usando cluster (ver exemplo de pseudo-binária disponibilizado no esqueleto original)

Para os filtros do domínio da frequência, pode-se utilizar os módulos disponíveis nas bibliotecas *scipy* ou *numpy*. Para as limiarizações, implemente as mesmas usando como base a versão em tons de cinza da imagem original. Sempre priorize o método de conversão da luminância (*luminosity*) para fazer a conversão para escala de cinza.

3 Parte II

Atenção: como esta segunda parte não é focada totalmente em implementação, **está liberado** o uso de qualquer biblioteca disponível que conseguir, incluindo as de processamento de imagens, como *OpenCV*.

Recomenda-se a instalação de uma biblioteca chamada *scikit-learn* (`http://scikit-learn.org/stable/`) a fim de facilitar as tarefas propostas. Esta biblioteca traz consigo implementações de algoritmos e ferramentas clássicas e bastante úteis para o contexto de aprendizado de máquina.

O grupo deve criar um classificador automático de imagens, conduzindo um experimento conforme descrito a seguir. Primeiramente, deve-se escolher um contexto de aplicação e um conjunto de imagens para servirem de dados de treinamento e testes. Selecione imagens para compor dois experimentos:

1. Base 1: Imagens de 3 ou tipos de animais diferentes. Por exemplo: cão, gato, peixe, ave. Priorize imagens cujo maior conteúdo seja o animal propriamente dito, com o mínimo possível de plano de fundo. Corte, usando um software como o GIMP, os animais de imagens com mais objetos e elementos, se achar necessário. Minimamente, obtenha cerca de 150 imagens.

2. Base 2: Imagens de personagens de histórias em quadrinhos, desenhos animados e afins. O importante é que os personagens tenham estilo artístico parecido. As mesmas observações a respeito de priorizar imagens contendo somente o personagem vale aqui.

Observações importantes:

- As imagens obtidas **não** precisam ser do mesmo tamanho
- Lide apenas com imagens coloridas. Os testes contemplarão análise de imagens em tons de cinza, mas obtidas via aplicação de filtros
- Algumas imagens podem não estar no padrão RGB, mas com outra codificação, como RGBA, indexado etc. Converta todas as imagens para o padrão RGB durante a criação da base. A biblioteca PIL, do Python, tem um método simples de realizar a conversão, o mesmo valendo para a biblioteca OpenCV.
- Separe cerca de 10% das imagens (após a padronização do formato RGB) em uma pasta separada. Este será o **conjunto de validação** final para cada experimento e suas figuras só serão processadas pelos algoritmos após terminado todo o processo de treinamento dos classificadores
- Crie algum esquema que facilite a rotulação automática das imagens, para associa-las às classes correspondentes. Uma sugestão é impor algum padrão de nomenclatura nos arquivos das imagens, contendo o nome da classe da mesma. Assim, pode-se facilmente aplicar uma função sobre o nome do arquivo que extrai deste o nome da classe correspondente àquela imagem. Por exemplo: *ddd-classe*.jpg, onde *ddd* seria um número inteiro contador de imagens, com três dígitos e *classe* seria o nome da classe daquela imagem. A bilbiotca *os*, do Python, pode auxiliar em tarefas que envolvam o sistema de arquivos do Sistema Operacional, como varrer pastas, analisar nomes de arquivos, verificar permissões etc.

O objetivo será encontrar o melhor classificador que a equipe conseguir para cada tipo de base de dados e comparar/analisar os resultados obtidos. Nas

próximas seções são descritos os passos que devem ser executados para os experimentos, que essencialmente seguirão o modelo mostrado nas aulas da disciplina.

3.1 Passo 1: Filtragem, Geração de Imagens Sintéticas e Preparação da Matriz de Dados

3.1.1 Filtros

Testaremos o desempenho dos classificadores mediante à aplicação prévia de diferentes filtros (já implementados pelos alunos em etapas anteriores do projeto) sobre as imagens das bases de dados. Incluiremos aqui, como etapa de todos os filtros, o redimensionamento de todas as imagens para um tamanho padrão. Sugere-se utilizar o paradigma de Orientação a Objetos, a fim de modularizar melhor o programa e prover melhor reaproveitamento de código. Implemente os seguintes filtros descritos adiante.

Antes, porém, uma observação a respeito do tamanho padrão a ser utilizado nas imagens. O tamanho padrão deve corresponder à largura igual a menor das larguras entre todas as imagens da base de dados, e a altura idem, a menor dentre todas as alturas de imagens presentes. Isso poderá distorcer formas de objetos nas imagens originais, mas não há problemas em relação a isso, é até, de certa forma, desejável, pois não deixa de ser uma forma indireta de se adicionar ruídos nos dados, sempre úteis para se obter classificadores robustos. Sendo assim, temos a seguinte lista de filtros para implementação:

- Básico: apenas redimensiona a imagem para o tamanho padrão, sem qualquer processamento adicional.
- Cinza: converte a imagem para tons de cinza e aplica o redimensionamento.
- Limiar: converte a imagem para tons de cinza, executa a Limiarização Global Adaptativa e redimensiona a mesma.
- Mediana: aplica filtro de mediana sobre a imagem (fixe a máscara em $5x5$) e em seguida aplica o redimensionamento
- MedCinza: aplica o filtro de mediana (mesma máscara), converte para tons de cinza e redimensiona
- MedLimiar: aplica o filtro de mediana (mesma máscara), converte para tons de cinza, aplica limiarização (mesma anterior) e redimensiona o resultado

3.1.2 Geração de Imagens Sintéticas

Após a aplicação do filtro sobre as imagens, deve-se ampliar a base de dados obtida, gerando-se sinteticamente alguns arquivos adicionais à base de dados, com base já nas imagens filtradas. Estas imagens sintéticas nada mais são do que cópias de algumas das imagens da base de dados (já filtradas), acrescidas de alguma variabilidade, como por exemplo, a adição de ruído gaussiano.

Sendo assim, sugere-se a seguinte metodologia para geração de imagens sintéticas:

- Selecione aleatoriamente 30% das imagens da base de dados e crie cópias das mesmas com espelhamento horizontal. Mantenha estas cópias em memória principal, para que sejam acrescidas à matriz de dados do classificador
- Repita a instrução do item acima, porém com espelhamento vertical agora
- Gere cópias de imagens com aplicação de ruído gaussiano de leve intensidade em cerca de 30% da base de dados já resultante da aplicação dos itens anteriores. Mantenha todas as imagens em memória principal para a montagem da matriz de dados

3.1.3 Preparação da Matriz de Dados

Para que os classificadores funcionem adequadamente, necessitamos de uma matriz, a qual chamaremos de X, deste ponto em diante, e um vetor coluna ao qual chamaremos de Y. Considere que a base de dados resultante dos passos das seções 3.1.1 e 3.1.2 possua n imagens, com m pixels cada uma no total (já que estão todas redimensionadas para o mesmo tamanho).

A matriz X será de dimensões $n \times m$, onde cada linha corresponderá a uma imagem, que já foi devidamente linearizada para corresponder a uma única linha desta matriz. Vale lembrar que, portanto, $m = l \times c$, onde l é o número de linhas da imagem e c o número de colunas da mesma. Sendo assim, os dados de todas as imagens estarão representados nesta matriz, que por fim terá um tamanho razoável em memória.

O vetor coluna Y terá dimensão de tamanho n, que também corresponderá ao número de imagens presentes na base de dados, porém seu conteúdo não será numérico, mas conterá os nomes das classes associadas à cada imagem. Sendo assim, para se saber a classe ao qual a imagem $X[i]$ da matriz, correspondendo à i-ésima linha de X, basta que se acesse a i-ésima linha de Y, ou seja $Y[i]$.

Todos os modelos de classificador são capazes de lidar com matrizes e vetores de classes nestes formatos, daí a conveniência desta etapa de preparação. Porém, apesar de em tese os dados já estarem prontos para o aprendizado de máquina ao término desta etapa, há um fator de conveniência que deve ser levado em conta.

Mesmo em uma imagem bastante pequena, digamos por exemplo, de 80×95 pixels, ao ser linearizada, gera um número de colunas considerável para X ($80 \times 95 = 7600$ pixels), tornando a classificação um problema de alta dimensionalidade. Pois, ao se enxergar X como uma base de dados em formato tabular, teríamos um número muito grande de atributos para cada dado para serem analisados, aumentando bastante o custo computacional do processo todo.

Desta forma, é conveniente que se reduza a dimensionalidade do problema. Para isto, utilizaremos a técnica de Análise de Componentes Principais (*Principal Components Analysis* - PCA, em inglês). Conforme explicado em sala, esta técnica visa, abrindo mão da precisão na representação dos dados, capturar o que há de mais relevante na variação e capacidade de explicar os dados, gerando novas dimensões para os mesmos, levando-se em conta apenas os fatores principais, cuja quantidade é informada pelo usuário.

A biblioteca *scikit-learn*, mencionada anteriormente, provê, no módulo *decomposition*, uma classe chamada *PCA* que encapsula e realiza este processo de redução das dimensões. Ao se instanciar o objeto, informa-se um parâmetro

n_components, o qual indica com quantas componentes principais deseja-se que a base de dados transformada fique.

Digamos que se deseje trabalhar com apenas 25 componentes principais (uma espécie de "pseudo-atributos" mais relevantes dos dados). Intancia-se o objeto com 25 como valor para o referido parâmetro e, em seguida, executa-se o método *fit_transform* sobre X para que se faça a redução, resultando em uma nova matriz $X'_{n\times 25}$ que, esta sim, será utilizada para treinar o classificador.

Para efeitos de simplificação, chamaremos deste ponto em diante X' apenas de X, tendo-se em conta que X é a matriz final, resultante dos passos descritos nesta seção e nas duas anteriores. Para o trabalho, padronize o número de componentes no valor aqui sugerido, isto é, 25. Pode-se, se o grupo desejar, realizar alguns testes com valores maiores e menores que este e ver o impacto desta redução no resultado final. Porém, tenha em mente que isto tomará mais tempo de execução do experimento, embora simples de implementar. Além disso, não se trata de um teste obrigatório, portanto, somente implemente se estiverem seguros de que isso não comprometerá o prazo de entrega do trabalho.

3.2 Passo 2: Preparação dos Classificadores e Treinamento

Esta seção e as seguintes serão disponbilizadas na Versão 2 deste texto. Aguarde liberação pelo professor.

4 Entrega

O arquivo compactado *.zip* a ser enviado deve conter:

- Todos os arquivos *.py* (ou na linguagem escolhida) resultantes da versão final do programa desenvolvido pelo grupo.
- Arquivo de texto (*.pdf*) com um **relatório**, que deve:
  - Identificar o(s) aluno(s) que realizou(aram) o trabalho, com nome e matrícula
  - Conter instruções para instalação e execução do software desenvolvido como, por exemplo, quais pacotes necessitam estar instalados para o funcionamento, como compilar etc.
  - Breve explicação a respeito de cada item implementado, descrevendo, em linhas gerais, como o método foi traduzido para a linguagem em questão, quais otimizações foram feitas, quais simplificações (se for o caso) foram feitas, decisões de projeto tomadas etc. Lembre-se de incluir as fontes utilizadas, caso lide com algo que fuja ao material dado em sala.
  - Coloque também exemplos com imagens antes e depois de cada recurso implementado.
- Arquivos das imagens originais (porém em tamanho não muito grande) utilizadas pelo grupo para exemplificar os recursos no relatório, caso venham a utilizar imagens além das fornecidas pelo professor.

Bons estudos.
Prof. Marcos