UNIVERSIDADE ESTADUAL DO NORTE FLUMINENSE
LABORATÓRIO DE ENGENHARIA E EXPLORAÇÃO DE PETRÓLEO

PROJETO ENGENHARIA

DESENVOLVIMENTO DO SOFTWARE DETERMINAÇÃO DE PROPRIEDADES FÍSICAS DE ROCHAS SEDIMENTARES UTILIZANDO DADOS OBTIDOS POR ANÁLISE DE IMAGENS DIGITAIS

TRABALHO DA DISCIPLINA PROGRAMAÇÃO PRÁTICA

Versão 1:

JULIANA REZENDE ÁVILA

Prof. André Duarte Bueno

MACAÉ - RJ
Junho - 2017

Sumário

Capítulo 1

Introdução

No presente projeto de engenharia desenvolve-se o *software* aplicado a engenharia de petróleo que utiliza o paradigma da orientação a objetos. Ele realiza a determinação de propriedades físicas de rochas sedimentares através de dados obtidos pela análise digital de imagens (ADI) utilizando o software ImageJ. Espera-se obter, utilizando valores de porosidade, curvas de distribuição do tamanho de poros, estimativas da superfície de poro específica e valores de permeabilidade das amostras de lâminas petrográficas das rochas.

1.1 Escopo do problema

O estudo de rochas através de análise microscópia consiste na utilização de amostras de testemunhos retirados de campo. Essas são cortadas e preparadas como lâminas delgadas para posteriores observações via microscópio ótico [Cunha et al., 2012]. Quando o microscópio é conectado a um computador é possível capturar imagens da lâmina em diferentes resoluções. A partir dessas imagens, utilizando o *software* gratuito ImageJ é possível extrair parâmetros de grande valia para caracterizar a rocha [Rasband, 2014].

Os parâmetros extraídos na análise provém de cada poro da imagem e são a área e o perímetro de cada poro em pixels. Através desses dados é possível fazer correlações para encontrar a porosidade, a permeabilidade, a distribuição de tamanho de poros na rocha e estimativa de superfície de poro específica. É de grande valia determinar essas características para realizar um bom estudo do reservatório que se espera produzir.

Nesse *software*, os dados gerados no ImageJ serão processados e transformados em informações importantes sobre a rocha analisada.

1.2 Objetivos

Os objetivos deste projeto de engenharia são:

- Objetivo geral:

  - Desenvolver um *software* que receba dados obtidos do ImageJ e converta-os em caracterização da rocha em análise.

- Objetivos específicos:

  - Fornecer distribuição de tamanho de poros da amostra.
  - Fornecer distribuição de tamanho de poros acumulada da amostra.
  - Estimar superfície específica dos poros.
  - Calcular a permeabilidade através do modelo de Kozeny-Carman.

Capítulo 2

Especificação

Apresenta-se neste capítulo do projeto de engenharia a concepção, a especificação do sistema a ser modelado e desenvolvido.

2.1 Nome do sistema e produto

Nome	Propriedades Fisicas
Componentes principais	Determinação de propriedades físicas de rochas sedimentaresa partir de dados de área e perímetro obtidos por análise de imagens digitais pelo *software* ImageJ.
Missão	Uma ferramenta para auxiliar e otimizar os estudos de lâminas petrográficas na caracterização do reservatório.

2.2 Especificação

O sistema deve ser capaz de ler arquivo de disco com dados de poros provindos do ImageJ, utilizar os dados de área e perímetro para estimar distribuição de tamanho dos poros da rocha e sua acumulada. Também deverá estimar a superfície específica de poro e o valor de permeabilidade utilizando método de Kozeny-Carman. Após o processamento as distribuições deverão ser apresentadas em forma de gráfico e os valores dos parâmetros calculados em arquivos de saída no formato ASCII.

Os gráficos serão gerados pelo *software* externo gnuplot (http://www.gnuplot.info).

Este *software* tem licenca GPL, podendo ser livremente distribuído e copiado.

2.2.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	O programa deve saber ler o arquivo de saída do *software* ImageJ com dados de área e perímetro (Figura 3.1).

RF-02	Os dados gerados/calculados serão salvos em arquivo de texto (formato ASCII) para uso posterior. (Veja exemplo no capítulo 7)

RF-03	Plotar dados de poro como frequência do raio de poro e fração acumulada do volume poroso. (Exemplo nas Figuras 7.10 e 7.12)

2.2.2 Requisitos não funcionais

RNF-01	Os cálculos de permeabilidade deverão ser feitos utilizando-se o modelo de Kozeny-Carman. (Equação 3.3)

RNF-02	O programa deverá ser multi-plataforma, podendo ser executado em *Windows*, *GNU/Linux* ou *Mac*.

2.3 Casos de uso

Nessa seção, apresenta-se caso de uso do sistema. Na Tabela 2.1 é explicitado o caso de uso geral e na Tabela 2.2 um caso de uso específico.

Tabela 2.1: Caso de uso geral

Nome do caso de uso:	Geral
Resumo/descrição:	Todas etapas do programa desenvolvido.
Etapas:	1. Carregar dados da imagem (exportados ImageJ). 2. Preencher vetores (área, perímetro e raio). 3. Converter unidade dos dados. 4. Calcular a área e perímetro total. 5. Calcular superfície específica. 6. Estimar permeabilidade. 7. Plotar histograma de frequência de raio de poro. 8. Acumular e normalizar dados de área. 9. Plotar fração acumulada do volume poroso.
Cenários alternativos:	Um caso alternativo seria se o usuário entrasse com o arquivo de dados incompatível (sem formato ASCII), o programa não rodaria.

Tabela 2.2: Exemplo de caso de uso específico

Nome do caso de uso:	Cálculo da permeabilidade da rocha por Kozeny-Carman.
Resumo/descrição:	Determinação da permeabilidade da rocha.
Etapas:	1. Criar objeto permeabilidade. 2. Receber dados de porosidade, perímetro e área do poro. 3. Calcular a superfície específica de poro (S_{pv}). 4. Calcular a área da superfície específica por unidade de volume de grão (S_{vgr}). 5. Calcular a permeabilidade (k_{kc}) pela correlação de Kozeny-Carman. 6. Gerar arquivo com resultados.
Cenários alternativos:	Um caso alternativo seria se o usuário entrasse com o arquivo de dados incompatível (sem formato ASCII), o programa não rodaria.

2.3.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário gerando os dados de entrada e importando para o *software,* aqui desenvolvido, para os resultados serem calculados, exportados e plotados.

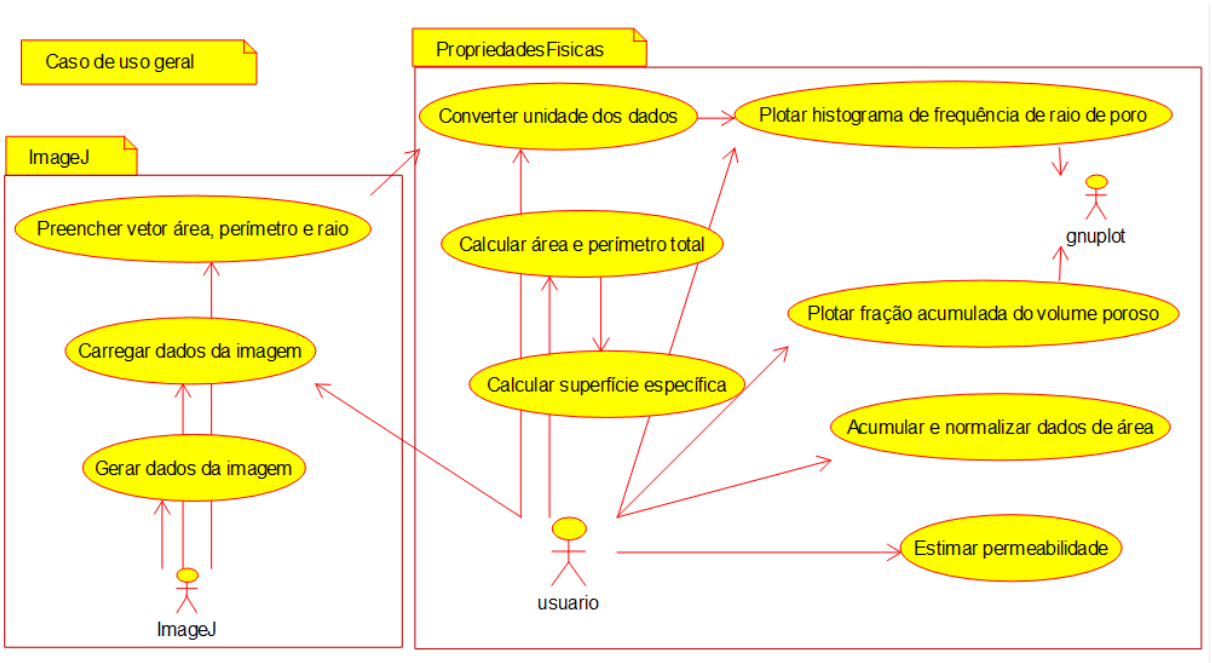

Figura 2.1: Diagrama de caso de uso – Caso de uso geral

2.3.2 Diagrama de caso de uso específico

Nesse caso de uso o usuário tem como finalidade estimar a permeabilidade da rocha. Ele irá gerar os dados necessários no ImageJ, carregá-los no *software* e receber os valores requeridos. Esse processo é visto na Figura 2.2.

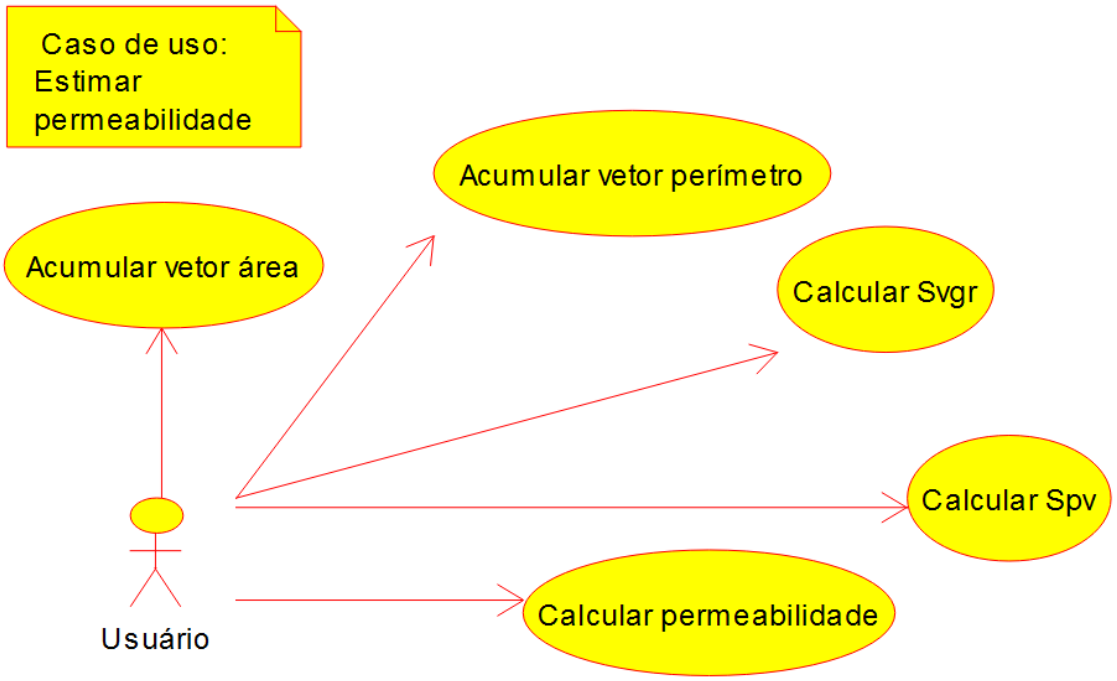

Figura 2.2: Diagrama de caso de uso específico – Calculo de permeabilidade

Capítulo 3

Elaboração

Neste capítulo apresenta-se o estudo de conceitos relacionados ao sistema a ser desenvolvido, a análise de domínio e a identificação de pacotes.

3.1 Análise de domínio

A escolha do desenvolvimento desse *software* teve como principal motivo otimizar estudos feitos de análise de imagens com lâminas petrográficas. Os resultados necessários para estudo eram trabalhosos e repetitivos, assim demandavam um tempo que não é necessário. Para diminuir o tempo de pesquisa da área de petrofísica da rochas esse projeto foi elaborado.

Após estudo dos requisitos/especificações do sistema, algumas entrevistas, estudos na biblioteca e disciplinas do curso foi possível identificar o domínio desse trabalho:

- Caracterização de reservatórios;
- Petrofísica fundamentada na física da rocha;
- Análise digital de imagens de lâminas petrográficas com microscopia ótica;
- Segmentação de imagens;
- Caracterização de poros.

3.2 Formulação teórica

A indústria de exploração de petróleo vem desenvolvendo diversas ferramentas e técnicas que auxiliam na caracterização petrofísica de campos petrolíferos. Essa caracterização consiste em determinar parâmetros de armazenamento e escoamento do fluido. Entre esses destacam-se a porosidade e a permeabilidade [Tiab and Donaldson, 2015]. Esses projetos visam à aproximação de um modelo de reservatório mais adequado ao quadro real. O

estudo de rochas através de análise microscópica consiste na utilização de lâminas extraídas de testemunhos retirados de campo, essas são cortadas e preparadas para posteriores observações via microscópio ótico [Cunha et al., 2012]. Na fase de preparação das lâminas, uma resina epóxi, de cor azulada, é injetada na amostra delgada a vácuo, de modo a ocupar os poros da rocha. Este procedimento faz com que seja ressaltado o espaço poroso ao ser observado num microscópio ótico. Quando esse tem acoplado uma câmera digital e é conectado a um computador, é possível capturar imagens da lâmina em diferentes resoluções.Utilizando-se *softwares* de tratamento de imagens é possível extrair parâmetros de grande valia para caracterizar a rocha.

A análise digital de imagem (ADI) de lâminas petrográficas é uma técnica que permite uma interpretação quantitativa do espaço poroso das rochas. O procedimento básico consiste em binarizar a imagem nas cores preta e branca, onde uma represente o espaço poroso e a outra o arcabouço, a matriz e o cimento da rocha.

Dentre diversas aplicações do *software* de tratamento é possível extrair informações gerais sobre a imagem binarizada, porosidade, ou mais específicas como área (A_p) e perímetro (L_p) de cada aglutinação individual de pixels pretos, ou seja de cada poro. Com estas informações a superfície específica de poros (S_{pv}) e a superfície específica por volume do grão (S_{vgr}) podem ser determinadas [Tiab and Donaldson, 2015]. E por fim, a permeabilidade pode ser estimada utilizando o modelo de Kozeny-Carman (k_{kc}) visto que os valores de porosidade (ϕ) são determinados pelo *software*.

$$S_{pv} = \frac{4L_p}{\pi A_p} \tag{3.1}$$

$$S_{vgr} = S_{pv}\left(\frac{\phi}{1-\phi}\right) \tag{3.2}$$

$$k_{kc} = \left(\frac{1}{5S_{vgr}^2}\right)\left(\frac{\phi^3}{(1-\phi)^2}\right) \tag{3.3}$$

Além disso, através do valor da área de cada poro reconhecido, pelo ImageJ, e adotando os poros com uma geometria circular é possível obter uma aproximaçãodo diâmetro dos poros (d), e assim uma distribuição do tamanho dos poros.

$$d = 2\sqrt{\frac{A_p}{\pi}} \tag{3.4}$$

Na Figura 3.1 é visto o arquivo de saída do *software* que será usado como dado de entrada do projeto aqui desenvolvido. Nele contém informações de área e perímetro de cada poro da imagem binarizada em unidade de *pixel*.

	Area	Perim.
1	383	185.765
2	9	10.485
3	4	8.485
4	10675	2.865.386
5	18	36.527
6	4	8.485
7	6450	1.236.511
8	5	11.314
9	2237	521.737
10	35	32.042
11	15	32.284
12	4	11.314
13	199	109.539
14	11	22.385
15	4	5.657
16	376	146.267
17	11	25.799
18	4	7.657
19	6	9.899
20	8	16.971
21	12	22.385
22	4	7.657
23	4	9.657
24	5	11.899
25	4	7.071
26	4	9.657

Figura 3.1: Exemplo do arquivo exportado pelo ImageJ (formato ASCII e extensão .txt)

Assim é possível plotar gráficos de histograma de frequência de raio de poros e da distribuição acumulativa do volume poroso (%) em função do raio de poro para realizar uma análise mais completa da rocha.

3.3 Identificação de pacotes – assuntos

- ImageJ (https://imagej.nih.gov/ij/): *software* livre usado para gerar os dados necessários para o *software* em desenvolvimento. Ele fornece os dados para a criação de objetos poros.
- VetorAreaPerimetro: dados dos poros da lâmina examinada.
- PermeabilidadeKC: cálculo da permeabilidade pelo modelo de Kozeny-Carman. Precisa dos dados da VetorAreaPerimetro para concluir as operações.

- Histograma: plotar gráficos para análises de propriedades físicas da rocha em questão. Depende dos dados de VetorAreaPerimetro.
- Gnuplot: software externo usado para plotar gráficos.

3.4 Diagrama de pacotes – assuntos

A representação dos pacotes se encontra na Figura 3.2.

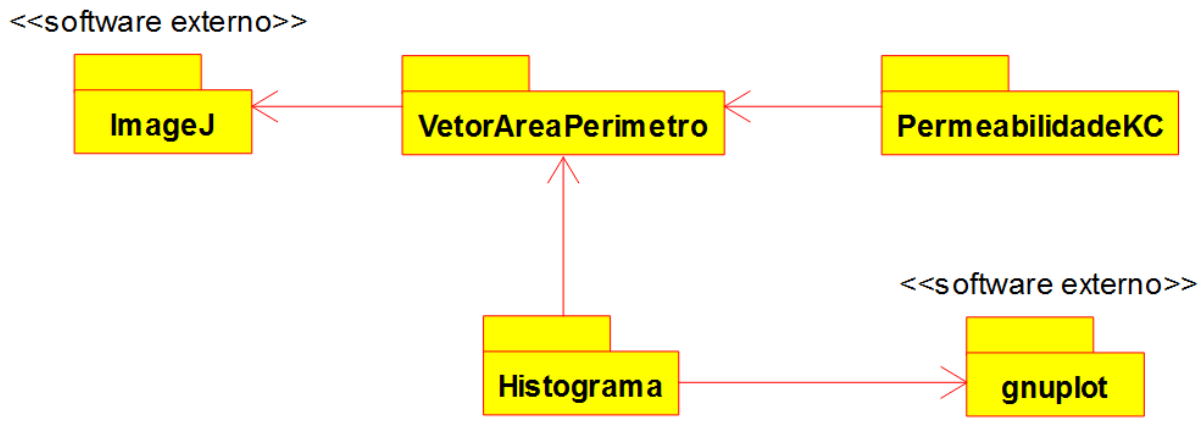

Figura 3.2: Diagrama de Pacotes

Capítulo 4

AOO – Análise Orientada a Objeto

A terceira etapa do desenvolvimento de um projeto de engenharia, neste caso um *software* aplicado a engenharia de petróleo, é a AOO – Análise Orientada a Objeto. A AOO utiliza algumas regras para identificar os objetos de interesse, as relacões entre os pacotes, as classes, os atributos, os métodos, as heranças, as associações, as agregações, as composições e as dependências. O resultado da análise é um conjunto de diagramas que identificam os objetos e seus relacionamentos.

4.1 Diagramas de classes

O diagrama de classes é apresentado na Figura 4.1.

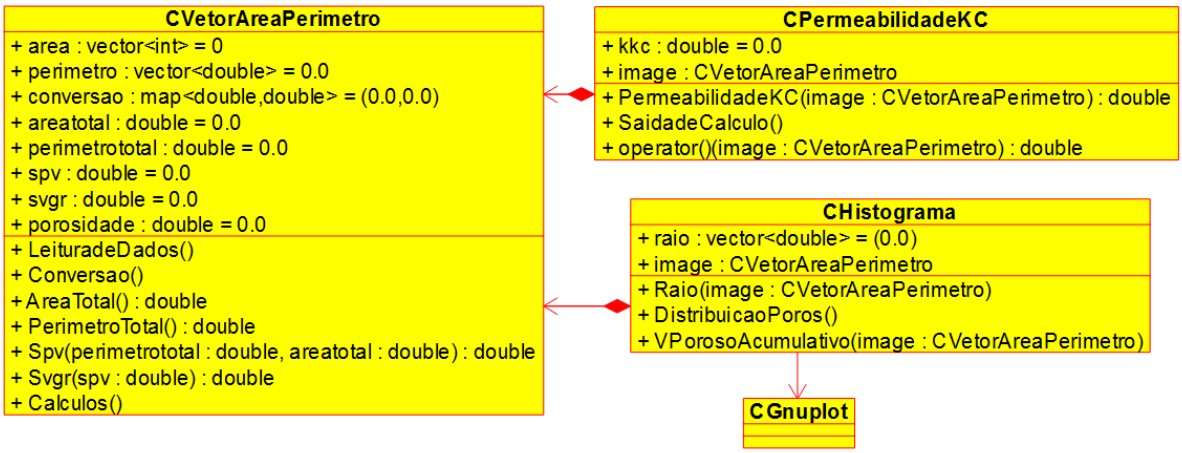

Figura 4.1: Diagrama de classes

4.1.1 Dicionário de classes

- Classe CVetorAreaPerimetro: representa a criação dos vetores com os dados importados do *software* ImageJ, sua sequente conversão para unidades usuais e cálculo de parâmetros.

- Classe CHistograma: representa a criação de gráficos para a distribuição do tamanho de poros na amostra e distribuição do volume poroso acumulado.
- Classe CPermeabilidadeKC: representa o cálculo da permeabilidade da amostra utilizando o método de Kozeny-Carman.

4.2 Diagrama de sequência – eventos e mensagens

O diagrama de seqüência enfatiza a troca de eventos e mensagens e sua ordem temporal. Contém informações sobre o fluxo de controle do software. Costuma ser montado a partir de um diagrama de caso de uso e estabelece o relacionamento dos atores (usuários e sistemas externos) com alguns objetos do sistema.

4.2.1 Diagrama de sequência geral

Veja o diagrama de seqüência na Figura 4.2. Primeiramente a leitura dos dados de entrada é realizada atribuindo valores para parâmetros. É realizada a conversão das unidades e em seguida os cálculos são executados (passos 4, 3, 5, 6 e 7). Logo depois um vetor de raio de poros é criado e a partir dele dois gráficos são construídos (passo 10 e 11).

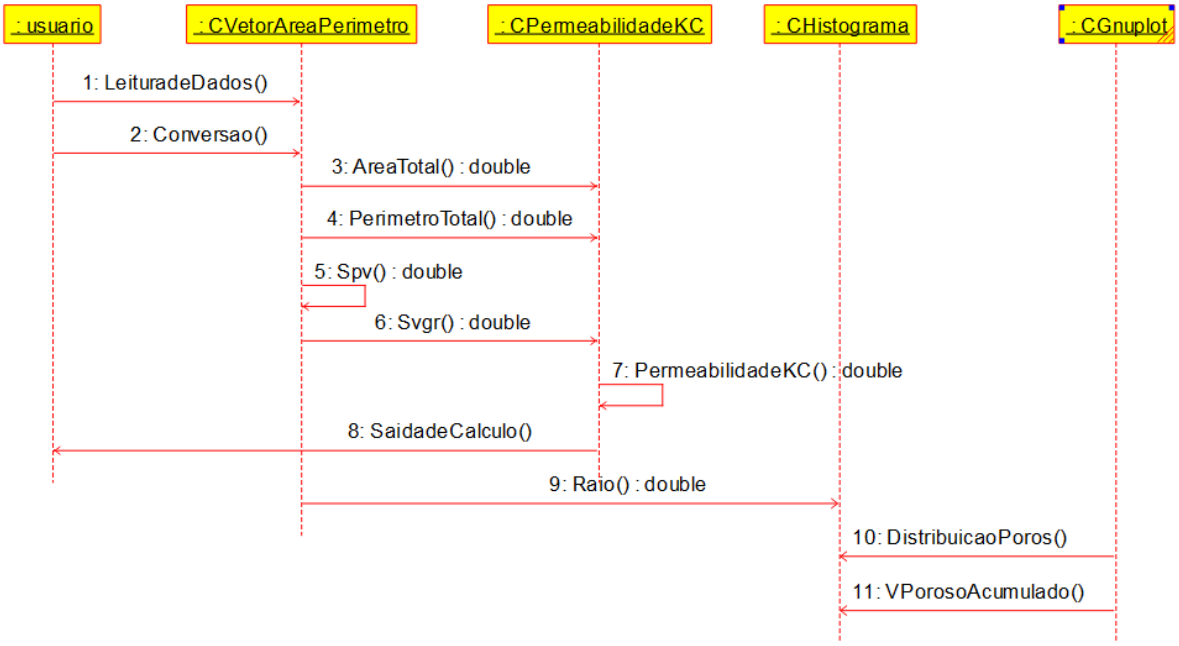

Figura 4.2: Diagrama de sequência

4.2.2 Diagrama de sequência específico

Veja o diagrama de sequência específico do cálculo da permeabilidade na Figura 4.3.

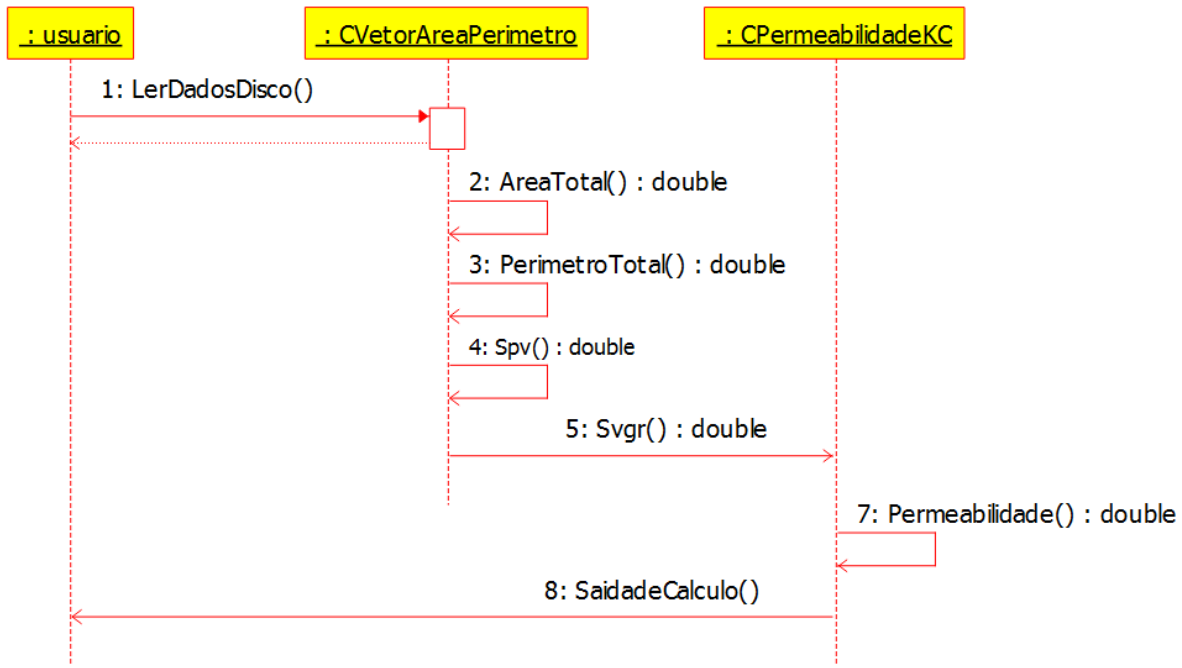

Figura 4.3: Diagrama de sequência específico

4.3 Diagrama de comunicação – colaboração

No diagrama de comunicação o foco é a interação e a troca de mensagens e dados entre os objetos.

Veja na Figura 4.4 o diagrama de comunicação mostrando a sequência em que os dados são utilizados no programa. Observe que os dados de entrada são essenciais para qualquer cálculo realizado.

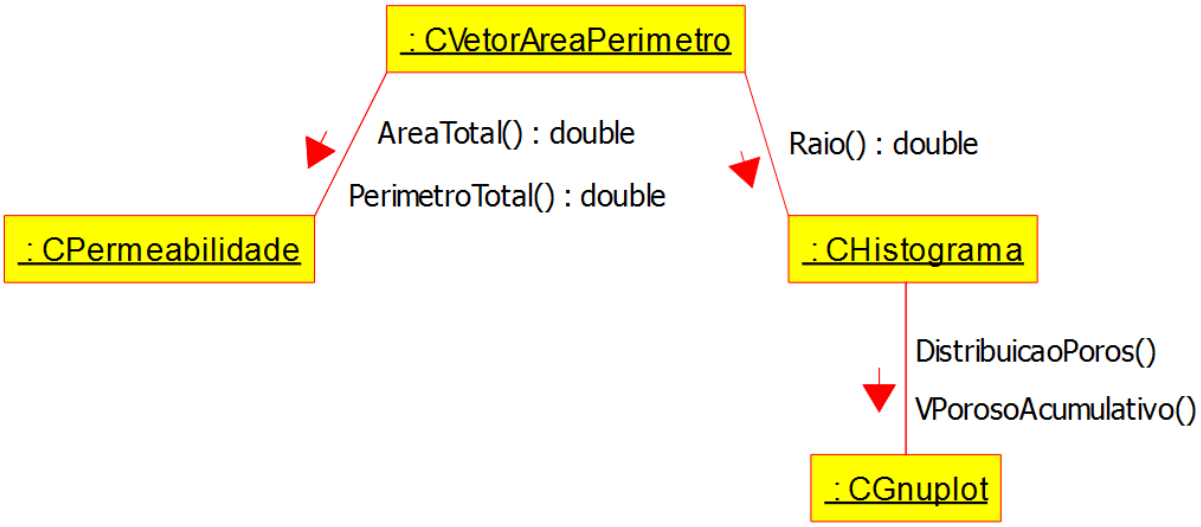

Figura 4.4: Diagrama de comunicação

4.4 Diagrama de máquina de estado

Um diagrama de máquina de estado representa os diversos estados que o objeto assume e os eventos que ocorrem ao longo de sua vida ou mesmo ao longo de um processo (histórico do objeto). É usado para modelar aspectos dinâmicos do objeto.

Veja na Figura 4.5 o diagrama de máquina de estado para o objeto. Observe que todos os dados embutidos no objeto são manipulados.

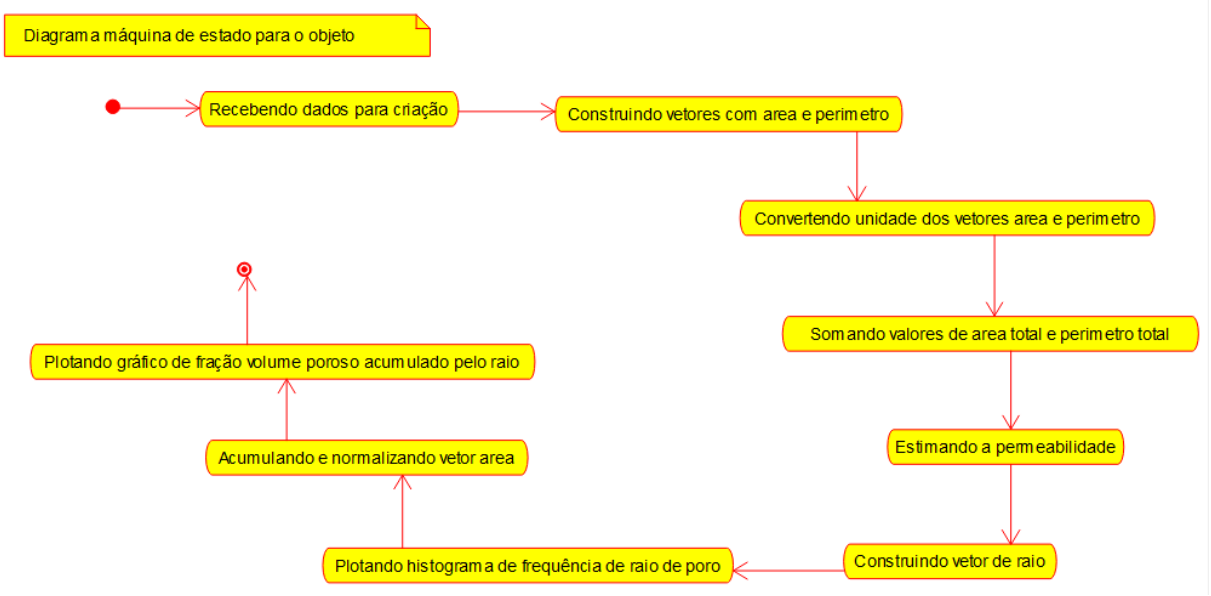

Figura 4.5: Diagrama de máquina de estado do objeto

4.5 Diagrama de atividades

Veja na Figura 4.6 o diagrama de atividades correspondente a uma atividade específica do diagrama de máguina de estado que é o cálculo da permeabilidade.

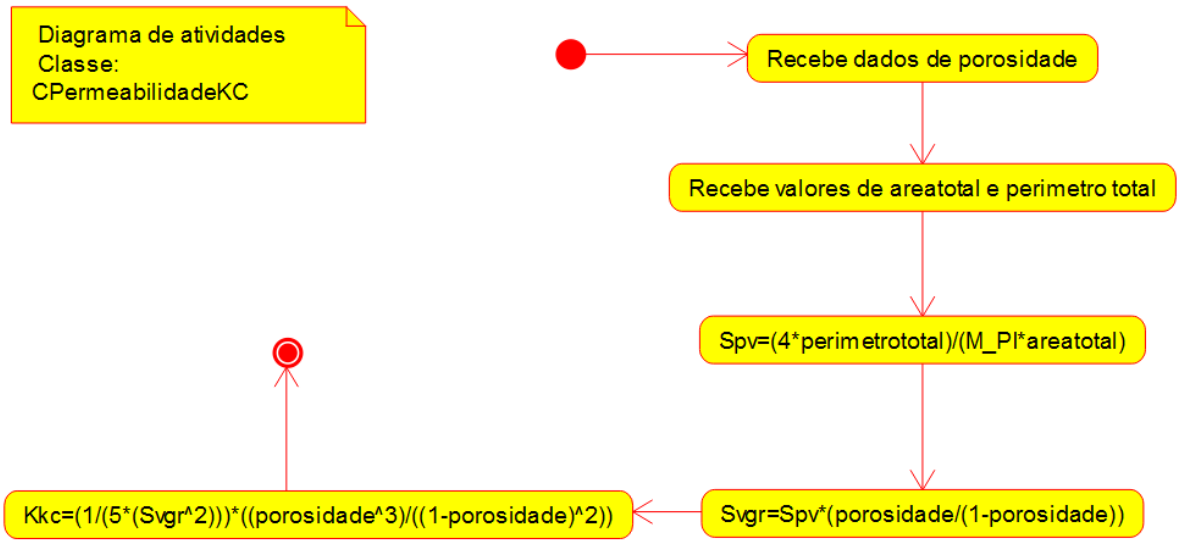

Figura 4.6: Diagrama de atividades do método PermeabilidadeKC da classe CPermeabilidadeKC

Capítulo 5

Projeto

Neste capítulo do projeto de engenharia veremos questões associadas ao projeto do sistema, incluindo protocolos, recursos, plataformas suportadas, implicações nos diagramas feitos anteriormente, diagramas de componentes e implantação. Na segunda parte revisamos os diagramas levando em conta as decisões do projeto do sistema.

5.1 Projeto do sistema

Depois da análise orientada a objeto desenvolve-se o projeto do sistema, o qual envolve etapas como a definição dos protocolos, da interface API, o uso de recursos, a subdivisão do sistema em subsistemas, a alocação dos subsistemas ao hardware e a seleção das estruturas de controle, a seleção das plataformas do sistema, das bibliotecas externas, dos padrões de projeto, além da tomada de decisões conceituais e políticas que formam a infraestrutura do projeto.

Deve-se definir padrões de documentação, padrões para o nome das classes, padrões de retorno e de parâmetros em métodos, características da interface do usuário e características de desempenho.

Segundo [Blaha and Rumbaugh, 2006, Rumbaugh et al., 1994], o projeto do sistema é a estratégia de alto nível para resolver o problema e elaborar uma solução. Você deve se preocupar com itens como:

1. Protocolos

    - A ligação com elementos externos é com a utilização do software ImageJ ao qual exportará os dados para entrada desse programa.
    - O programa utilizará biblioteca padrão C++ e o Gnuplot.
    - O formato dos dados de saída gerados pelo software serão arquivos .dat.

2. Recursos

- Esse projeto necessitará de uma máquina computacional com HD, processador, um arquivo .xml para entrada de dados além do teclado para o mesmo fim e o monitor para a saída de dados e plots.
- Haverá uilização do Gnuplot a fim de plotar gráficos.
- Como recurso essencial o arquivo com os dados de entrada é necessário para o uso do programa.

3. Plataformas

- O software será desenvolvido na linguagem C++ utilizando o conceito de orientação a objeto.
- Por ser uma linguagem universal, o programa é multiplataforma, pode ser executado em GNU/Linux, MAC OS X e Windows.
- Utilizará como interface de desenvolvimento o software Dev-C++ versão 4.9.9.2 presente no sistema operacional Windows 7 de 32 Bits com processador Intel Core i3-2310M.

5.2 Projeto orientado a objeto – POO

O projeto orientado a objeto é a etapa posterior ao projeto do sistema. Baseia-se na análise, mas considera as decisões do projeto do sistema. Acrescenta a análise desenvolvida e as características da plataforma escolhida (hardware, sistema operacional e linguagem de softwareção). Passa pelo maior detalhamento do funcionamento do software, acrescentando atributos e métodos que envolvem a solução de problemas específicos não identificados durante a análise.

Envolve a otimização da estrutura de dados e dos algoritmos, a minimização do tempo de execução, de memória e de custos. Existe um desvio de ênfase para os conceitos da plataforma selecionada.

Efeitos do projeto no modelo estrutural

- Somente foi necessário instalar o programa Gnuplot na máquina computacional.

Após revisão, foi visto que não houve a necessidade de alterar os diagramas da análise orientada a objeto, então os sequintes itens não precisaram de alterações:

- Efeitos do projeto no modelo dinâmico
- Efeitos do projeto nos atributos
- Efeitos do projeto nos métodos

- Efeitos do projeto nas heranças
- Efeitos do projeto nas associações
- Efeitos do projeto nas otimizações

Depois de revisados os diagramas da análise é possível montar dois diagramas relacionados à infraestrutura do sistema. As dependências dos arquivos e bibliotecas podem ser descritos pelo diagrama de componentes, e as relações e dependências entre o sistema e o hardware podem ser ilustradas com o diagrama de implantação.

5.3 Diagrama de componentes

O diagrama de componentes mostra a forma como os componentes do software se relacionam, suas dependências. Inclui itens como: componentes, subsistemas, executáveis, nós, associações, dependências, generalizações, restrições e notas. Exemplos de componentes são bibliotecas estáticas, bibliotecas dinâmicas, dlls, componentes Java, executáveis, arquivos de disco, código-fonte.

Veja na Figura 5.1 o diagrama de componentes. Note que os únicos componentes extras, além do software são o arquivo de entrada e o Gnuplot.

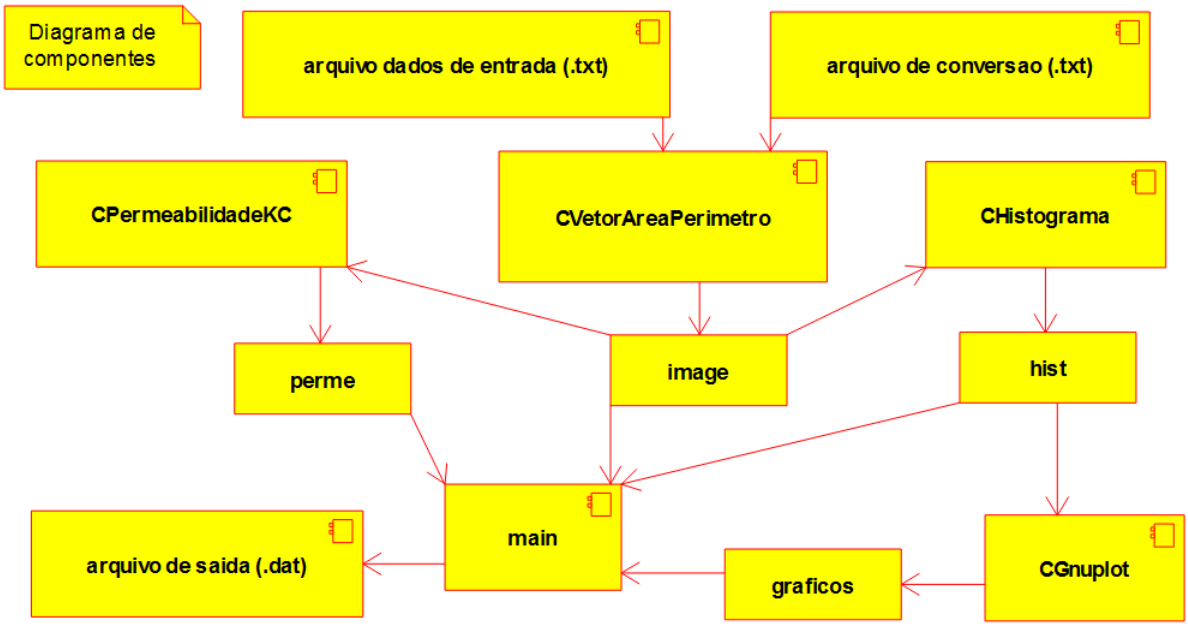

Figura 5.1: Diagrama de componentes do sistema

5.4 Diagrama de implantação

O diagrama de implantação é um diagrama de alto nível que inclui relações entre o sistema e o hardware e que se preocupa com os aspectos da arquitetura computacional escolhida. Seu enfoque é o hardware, a configuração dos nós em tempo de execução.

Veja na Figura 5.2 o diagrama de implantação do programa.

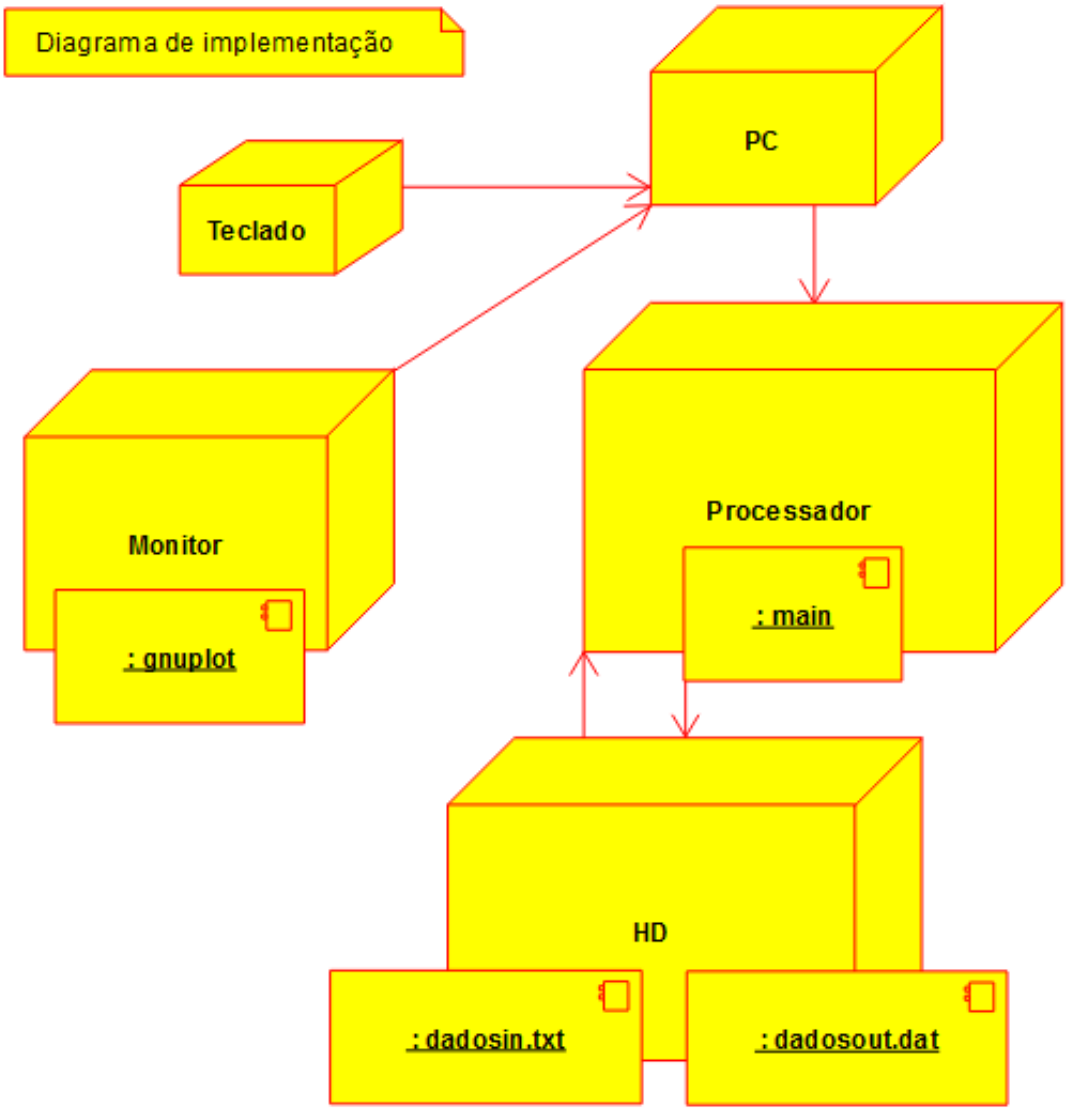

Figura 5.2: Diagrama de implantação

Capítulo 6

Implementação

Neste capítulo do projeto de engenharia apresenta-se os códigos fonte que foram desenvolvidos.

6.1 Código fonte

Apresenta-se a seguir um conjunto de classes (arquivos .h e .cpp) além do programa `main`.

Apresenta-se na listagem 6.1 o arquivo com código da classe `CVetorAreaPerimetro`.

Listing 6.1: Arquivo de cabeçalho da classe CVetorAreaPerimetro.

```
1 /**
2 @autor Juliana Avila
3 @file CVetorAreaPerimetro.h
4 @brief CVetorAreaPerimetro: cria um vetor de poros contidos na imagem,
      ja convertido, com as informacoes importadas do arquivo gerado pelo
      ImageJ
5 */
6
7 #ifndef CVetorAreaPerimetro_h
8 #define CVetorAreaPerimetro_h
9
10 #include <vector>
11 #include <map>
12 #include <string>
13 #include <iostream>
14 #include <fstream>
15 #include <cmath>
16
17 class CVetorAreaPerimetro {
18
19 public: //main nao acessa qnd e privado
```

```
20          std::vector<double> area;
21          std::vector<double> perimetro;
22          std::map<double,double> conversao;
23          double areatotal;
24          double perimetrototal;
25          double spv;
26          double svgr;
27          double porosidade;
28
29  public:
30          //construtores e destrutor
31          //construtor default
32          CVetorAreaPerimetro():area(0.0), perimetro(0.0), areatotal(0.0),
                perimetrototal(0.0), spv(0.0), svgr(0.0) {}
33
34          //construtor de copia
35          //CVetorAreaPerimetro(const CVetorAreaPerimetro& image) {
36                                      //area=image.area; perimetro=image.
                                            perimetro; conversao=image.conversao;
                                             areatotal=image.areatotal;
                                            perimetrototal=image.perimetrototal;}
37          //construtor sobrecarregado
38          //CImagemBinarizada(std::vector<int> _area, std::vector<double>
                _perimetro, map<double,double> _conversao, double _so, double
                _areatotal, double _perimetrototal):
39                                       //area(_area), perimetro(_perimetro),
                                            conversao(_conversao), areatotal(
                                            _areatotal), perimetrototal(
                                            _perimetrototal) {}
40          //destrutor
41          ~CVetorAreaPerimetro(){};
42
43          void LeituradeDados();
44          void Conversao(); //usar arquivo para conversao do tipo
                especificado
45          double AreaTotal();
46          double PerimetroTotal();
47
48          double Spv(double perimetrototal, double areatotal) {return spv
                =((4.0*perimetrototal)/(M_PI*areatotal));}
49
50          double Svgr(double spv) {return svgr=(spv*(porosidade/(1.0-
                porosidade)));}
51
52          void Calculos();
53
54          friend class CHistograma;
55          friend class CPermeabilidadeKC;
```

```
56
57 };
58 #endif
```

Apresenta-se na listagem 6.2 o arquivo de implementação da classe `CVetorAreaPerimetro`.

Listing 6.2: Arquivo de implementação da classe CVetorAreaPerimetro.

```
60 /**
61 @autor Juliana Avila
62 @file CVetorAreaPerimetro.cpp
63 @brief CVetorAreaPerimetro: cria um vetor de poros contidos na imagem,
     ja convertido, com as informacoes importadas do arquivo gerado pelo
     ImageJ
64 */
65
66 #include "CVetorAreaPerimetro.h"
67 #include <iostream>
68 #include <fstream>
69 #include <iomanip> //setw
70 #include <algorithm> //sort
71 #include <string>
72 #include <cmath>
73 using namespace std;
74
75 void CVetorAreaPerimetro::LeituradeDados() {
76     string arquivodados;
77     cout << "\nEntre com o arquivo de dados exportados do ImageJ" <<
         endl;
78     getline (cin, arquivodados);
79     ifstream fin (arquivodados.c_str()); //abre arquivo solicitado na
         construcao do objeto com os dados
80     if (!fin) {
81         cout << "\nFalha acesso ao arquivo. Encerrando!" << endl;
82         exit (0);
83             }
84
85     //preenche vetor area e perimetro
86     double a;
87     double p;
88     area.clear(); //zera vetor de area
89     perimetro.clear(); //zera vetor de perimetro
90     fin.ignore(256, '\n'); //ignora a primeira linha
91
92     while (!fin.eof()) {
93         fin >> a;    // numero do poro
94         fin >> a;    area.push_back(a);
95         fin >> p;    perimetro.push_back(p);
96     }
97
```

```
98      fin.close(); //fecha arquivo de dados
99      sort(area.begin(),area.end());
100     sort(perimetro.begin(),perimetro.end());
101
102     //conferindo se deu certo
103     ofstream fout;
104     system ("mv␣saida.dat␣saida.dat~"); //cria novo arquivo e deixa
            antigo para tras
105     fout.open("saida.dat");      //vai para o fim do arquivo
106     fout << "\n0s␣valores␣de␣area␣e␣perimetro␣(em␣pixels)␣sao(foram␣
            ORDENADOS):\n" << endl;
107     fout << "\n" << setw(20) << "AREA(pixel2)" << setw(20) << "PERIMETRO
            (pixel)" << endl;
108     for (int i=0; i<area.size(); i++) {
109         fout << setw(20) << area[i] << setw(20) << perimetro[i] << endl;
110     }
111 }
112
113 void CVetorAreaPerimetro::Conversao() {
114      //arquivo de conversao de unidade
115       string nomearquivo;
116       cout << "\nEntre␣com␣o␣arquivo␣de␣conversao␣(modelo␣especificado)"
              << endl;
117       getline (cin, nomearquivo);
118       ifstream in (nomearquivo.c_str());
119       if (!in) {
120             cout << "\nArquivo␣não␣encontrado" << endl; }
121
122       double ampliacao;
123       double fator;
124       in.ignore (100, '\n'); //ignora a primeira linha
125       while (!in.eof()) {
126           in >> ampliacao;
127           in >> fator;
128           conversao.insert(make_pair(ampliacao, fator));
129           }
130       in.close();
131
132       //conferindo se deu certo
133       ofstream fout;
134       fout.open("saida.dat", ios::app); //vai para o fim do arquivo
135       map<double,double>::const_iterator it;
136       fout << "\nA␣conversao␣utilizada␣segue␣a␣tabela␣seguinte:␣" <<
              endl;
137       fout << "\n" << setw(20) << "AMPLIACAO" << setw(20) << "FATOR" <<
              endl;
138       for(it=conversao.begin(); it!=conversao.end(); it++) {
139             fout << setw(20) << it->first << setw(20) << it->second <<
```

```
                endl; }
140
141       //pegando o valor da ampliacao e armazenando valor do fator de
              conversao
142       double amp;
143       cout << "\nQual␣foi␣a␣ampliacao␣do␣microscopio␣utilizada␣na␣lamina
              ?" << endl;
144       cin >> amp;
145       double fat;
146       // precisa verificar se chegou ao fim do map e não encontrou o
              fator; ocorre quando o usuário seleciona um valor de ampliacao
              que não esta no arquivo. Neste caso pode pedir para usuário
              entrar com o fator.
147       if (conversao.find(amp) != conversao.end()) {
148         map<double,double>::const_iterator iter;
149         iter=conversao.find(amp);
150         fat=iter->second;
151         fout << "\n0␣fator␣de␣conversao␣é:␣" << fat << "␣um/pixel" <<
                endl; }
152       else {
153         cout << "\nQual␣é␣o␣fator␣de␣conversao␣de␣pixel␣para␣micrometros
              ␣do␣microscopio␣utilizado␣para␣essa␣imagem?" << endl;
154         cin >> fat; }
155
156       //converte os valores
157       for (int i=0; i<area.size(); i++) {
158             area[i]=area[i]*fat*fat; //pow(x,2)
159             }
160       for (int i=0; i<perimetro.size(); i++) {
161             perimetro[i]=perimetro[i]*fat;
162             }
163
164       //conferindo
165       //fout.open("saida.dat", ios::app);       //vai para o fim do
              arquivo
166       fout << "\n0s␣valores␣de␣area␣e␣perimetro␣(em␣micrometros)␣sao:\n"
               << endl;
167       fout << "\n" << setw(20) << "AREA(um2)" << setw(20) << "PERIMETRO(
              um)" << endl;
168       for (int i=0; i<area.size(); i++) {
169             fout << setw(20) << area[i] << setw(20) << perimetro[i] <<
                endl;
170       }
171 }
172
173 double CVetorAreaPerimetro::AreaTotal() {
174     areatotal=0.0;
175     for (int i=0; i<area.size(); i++) {
```

```
176         areatotal+=area[i];
177     }
178 
179     ofstream fout;
180     fout.open("saida.dat", ios::app);    //vai para o fim do arquivo
181     fout << "\nA␣soma␣das␣areas␣é:␣" << areatotal << "␣um^2" << endl;
182 
183     return areatotal;
184 }
185 
186 double CVetorAreaPerimetro::PerimetroTotal() {
187      perimetrototal=0.0;
188      for (int i=0; i<perimetro.size(); i++) {
189         perimetrototal+=perimetro[i];
190      }
191 
192      ofstream fout;
193      fout.open("saida.dat", ios::app);  //vai para o fim do arquivo
194      fout << "\nA␣soma␣dos␣perimetros␣é:␣" << perimetrototal << "␣um" <<
             endl;
195 
196      return perimetrototal;
197      }
198 
199 void CVetorAreaPerimetro::Calculos() {
200       ofstream fout;
201       fout.open("saida.dat", ios::app); //vai para o fim do arquivo
202       fout << "\nO␣valor␣da␣superficie␣especifica␣é:␣" << spv << "␣1/um"
              << endl;
203       fout << "\nO␣valor␣da␣superficie␣especifica␣por␣volume␣de␣grão␣é:␣
             " << svgr << "␣1/um"<< endl;
204 
205     }
```

Apresenta-se na listagem 6.3 o arquivo com código da classe `CPermeabilidadeKC`.

Listing 6.3: Arquivo de cabeçalho da classe CPermeabilidadeKC.

```
206 /**
207 @autor Juliana Avila
208 @file CPermeabilidadeKC.h
209 @brief CPermeabilidadeKC: calcula a permeabilidade a partir da area e do
      periodo
210 */
211 
212 #ifndef CPermeabilidadeKC_h
213 #define CPermeabilidadeKC_h
214 
215 #include <cmath>
216 #include <fstream>
```

```
217 #include <vector>
218 #include "CVetorAreaPerimetro.h"
219
220 class CPermeabilidadeKC {
221
222 public:
223         CVetorAreaPerimetro image;
224         double kkc;
225
226 public:
227         //construtores e destrutor
228         //construtor default
229         CPermeabilidadeKC() {}
230
231         //destrutor
232         ~CPermeabilidadeKC(){};
233
234          double PermeabilidadeKC(CVetorAreaPerimetro image) {return kkc
                =(((1.0/(5.0*(image.svgr*image.svgr)))*((image.porosidade*
                image.porosidade*image.porosidade)/((1.0-image.porosidade)
                *(1.0-image.porosidade))))/0.00098717);}
235
236          double operator()(CVetorAreaPerimetro image) {return kkc; }
237
238         void SaidadeCalculo();
239
240 };
241 #endif
```

Apresenta-se na listagem 6.4 o arquivo de implementação da classe `CPermeabilidadeKC`.

Listing 6.4: Arquivo de implementação da classe CPermeabilidadeKC.

```
242 /**
243 @autor Juliana Avila
244 @file CPermeabilidadeKC.cpp
245 @brief CPermeabilidadeKC: calcula a permeabilidade a partir da area e do
        periodo
246 */
247
248 #include "CPermeabilidadeKC.h"
249 #include <iostream>
250
251 using namespace std;
252
253 void CPermeabilidadeKC::SaidadeCalculo() {
254       ofstream fout;
255       fout.open("saida.dat", ios::app); //vai para o fim do arquivo
256       fout << "\n0␣valor␣da␣permeabilidade␣(modelo␣Kozeny-Carman):␣" <<
              kkc << "␣mD" << endl;
```

```
257         cout << "\n0␣valor␣da␣permeabilidade␣(modelo␣Kozeny-Carman):␣" <<
                kkc << "␣mD" << endl;
258         }
```

Apresenta-se na listagem 6.5 o arquivo com código da classe `CHistograma`.

Listing 6.5: Arquivo de cabeçalho da classe CHistograma.

```
259 /**
260 @autor Juliana Avila
261 @file CHistograma.h
262 @brief CHistograma: plota graficos
263 */
264
265 #ifndef CHistograma_h
266 #define CHistograma_h
267
268 #include <iostream>
269 #include <fstream>
270 #include <vector>
271 #include <cmath> //sqrt e M_PI
272 #include "CVetorAreaPerimetro.h"
273 #include "CGnuplot.h"
274 #include "CPermeabilidadeKC.h"
275
276 class CHistograma {
277
278 public:
279          std::vector<double> raio;
280          //double M_PI;
281          CVetorAreaPerimetro image;
282
283 public:
284         //construtor e destrutor
285         //construtor default
286         CHistograma(): raio(0.0) {}
287
288         //construtor de copia
289         //CHistograma(const CHistograma& hist) {raio=hist.raio;}
290         //construtor sobrecarregado
291         //CHistograma(std::vector<double> _raio): raio(_raio) {}
292
293         //destrutor
294         ~CHistograma(){};
295
296         void Raio(CVetorAreaPerimetro& image);
297         void DistribuicaoPoros();
298         void VPorosoAcumulativo(CVetorAreaPerimetro& image);
299
300 };
```

```
301 #endif
```

Apresenta-se na listagem 6.6 o arquivo de implementação da classe `CHistograma`.

Listing 6.6: Arquivo de implementação da classe CHistograma.

```
302 /**
303 @autor Juliana Avila
304 @file CHistograma.cpp
305 @brief CHistograma: plota graficos
306 */
307
308 #include "CHistograma.h"
309 #include "CGnuplot.h"
310 #include <iostream>
311 #include <string>
312 #include <iomanip>
313 #include <fstream>
314
315 using namespace std;
316
317 void CHistograma::Raio(CVetorAreaPerimetro& image) {
318     //preenche o vetor raio com seus valores acessando o vetor area do
          CVetorAreaPerimetro
319       for (int i=0; i<image.area.size(); i++) {
320         raio.push_back(sqrt(image.area[i]/M_PI));
321                                                 }
322       //conferindo se deu certo e mostrando na tela o vetor criado
323       ofstream fout;
324       fout.open("saida.dat", ios::app); //vai para o fim do arquivo
325       fout << "\n0 vetor de raio criado a partir dos dados da area é:"
             << endl;
326       fout << setw(30) << "RAIO(um)" << endl;
327       for (int i=0; i<raio.size(); i++) {
328         fout << setw(30) << raio[i] << endl;
329                                           }
330 }
331
332
333 void CHistograma::DistribuicaoPoros() {
334  //criando histograma
335     vector<int> histograma (5,0);
336     for (int i=0; i<raio.size(); i++) {
337         if      (raio[i]<=1.0)                          histograma[0]+=1;
338         else if (raio[i]>=1.0 && raio[i]<10.0)          histograma[1]+=1;
339         else if (raio[i]>=10.0 && raio[i]<100.0)        histograma[2]+=1;
340         else if (raio[i]>=100.0 && raio[i]<1000.0)      histograma[3]+=1;
341         else                                            histograma[4]+=1;
342     }
343      vector<double> hist_x (5,0);
```

```
344       hist_x[0]=0.1;
345       hist_x[1]=1.0;
346       hist_x[2]=10.0;
347       hist_x[3]=100.0;
348       hist_x[4]=1000.0;
349
350        ofstream hout;
351        hout.open("histograma.dat");       //vai para o fim do arquivo
352        for (int i=0; i<histograma.size(); i++) {
353          hout << setw(10) << histograma[i] << setw(10) << i << "␣=␣" <<
               hist_x[i] << endl;
354                                             }
355        CGnuplot gdp; // Construtor (grafico distribuicao tamanho de poros
             )
356        gdp.set_style("histograms");
357        gdp.XLabel("Raio␣de␣poros␣(um)"); // Rotulo eixo x
358        gdp.YLabel("Frequencia"); // Rotulo eixo y
359        gdp.YAutoscale();
360        gdp << "set␣term␣png\n"; // terminal é arquivo png
361        gdp << "set␣out␣\"frequencia_raio.png\"\n"; // nome do arquivo
362        gdp.plotfile_x("histograma.dat", 1, ""); // 1:xtic(2)
363        gdp << "xtic(2)\n";
364        //gdp.PlotVector(raio, "Distribuicao de tamanho de poros");
365        //cin.get();
366 }
367
368 void CHistograma::VPorosoAcumulativo(CVetorAreaPerimetro& image) {
369        //cria vetor area acumulada
370        vector<double> areaA;
371        areaA.push_back(image.area[0]);
372        for (int i=1; i<image.area.size(); i++) {
373          areaA.push_back(areaA[i-1]+image.area[i]);
374        }
375
376        //cria vetor area acumulada normalizada (%)
377        vector<double> areaAN;
378        for (int i=0; i<areaA.size(); i++) {
379          areaAN.push_back((areaA[i]/image.areatotal)*100.0);
380        }
381
382        //conferindo se deu certo
383        ofstream fout;
384        fout.open("saida.dat", ios::app);
385        fout << "\n0␣vetor␣de␣area␣tratado␣é:" << endl;
386        fout << setw(30) << "\nAREA␣ACUMULADA␣NORMALIZADA(um2)" << endl;
387        for (int i=0; i<areaAN.size(); i++) {
388          fout << setw(30) << areaAN[i] << endl;
389        }
```

```
390      //arquivo com os valores do grafico
391       ofstream raio_area;
392       raio_area.open("raio_area.dat");
393       for (int i=0; i<areaAN.size(); i++)
394         raio_area << setw(30) << raio[i] << setw(30) << areaAN[i] <<
              endl;
395       raio_area.close();
396
397       //plotando com gnuplot
398       CGnuplot gva; // Construtor (grafico volume acumulado)
399       gva.Style("linespoints");
400       gva.XLabel("Raio de poros (um)"); // Rotulo eixo x
401       gva.YLabel("Fracao acumulatica de volume poroso (%)"); // Rotulo
             eixo y
402       gva.set_xlogscale(10);
403       gva.XAutoscale();
404       gva.YRange(0, 100);
405       gva << "set term png\n"; // terminal é arquivo png
406       gva << "set out \"vporoso_acumulado.png\"\n"; // nome do arquivo
407       gva.plotfile_xy("raio_area.dat", 1,2);
408       //gva.PlotVector(raio, areaAN, "Distribuicao acumulativa de
             tamanho de poros1");
409       //cin.get();
410  }
```

Apresenta-se na listagem 6.7 o programa que usa as classes anteriores.

Listing 6.7: Arquivo de implementação da função `main()`.

```
411 /**
412 @autor Juliana Avila
413 @file main.cpp
414 @brief main: implementa as classes
415 */
416
417 #include <iostream>
418 #include "CVetorAreaPerimetro.h"
419 #include "CPermeabilidadeKC.h"
420 #include "CHistograma.h"
421
422 using namespace std;
423
424 int main() {
425        CVetorAreaPerimetro image; //pede o arquivo com os dados de area
             e perimetro
426        image.LeituradeDados(); //preenche e mostra os vetores area e
             perimetro em pixel
427        image.Conversao(); //pede o arquivo de conversao, le e mostra ele
              na tela
428                           //pergunta a ampliacao da imgem, converte e
```

```
                                mostra na tela os vetores de area e
                                perimetro convertidos
429         image.AreaTotal(); //calcula a soma das areas dos poros
430         image.PerimetroTotal(); //calcula a soma dos perimetros dos poros
431         cout << "\nQual o valor de porosidade encontrado no experimento?
                (ImageJ) (frac)" << endl;
432         cin >> image.porosidade;
433         image.Spv(image.perimetrototal, image.areatotal); //calcula
                superficie especifica do poro total
434         image.Svgr(image.spv); //calcula area especifica do poro total
435         image.Calculos();
436
437         CPermeabilidadeKC perme;
438         perme.PermeabilidadeKC(image); //calcula permeabilidade
439         perme.SaidadeCalculo(); //mostra na tela os valores calculados
                acima
440
441         CHistograma hist; //cria o vetor de raio a partir do vetor area
                da classe CVetorAreaPerimetro
442         hist.Raio(image);
443         hist.DistribuicaoPoros(); //plota um histograma com os tamanhos
                do poro
444         hist.VPorosoAcumulativo(image); //plota um grafico de poros pela
                acumulacao do volume (area)
445 }
```

Capítulo 7

Teste

Todo projeto de engenharia passa por uma etapa de testes. Neste capítulo apresentamos alguns testes do *software* desenvolvido. Estes testes devem dar resposta aos diagramas de caso de uso inicialmente apresentados (diagramas de caso de uso geral e específicos).

7.1 Teste 1: Cálculo de permeabilidade e construção de gráficos

Neste teste, será utilizado, como dados de entrada, o arquivo "results.txt" (Figura 9.1) com os dados de área e perímetro dos poros (arquivo exportado do ImageJ) e o arquivo "conversao.txt" (Figura 7.2) criado para obter dados de conversão de pixels para micrômetros que depende do microscópio utilizado para extrair as imagens.

Inicialmente será solicitado o arquivo de entrada, visto na Figura 8.1, e o usuário digita o nome do *.txt de entrada.

O programa irá preencher vetores com informações de área e perímetro de cada poro. Esses valores serão impressos no arquivo de "saida.dat" para inspeção do usuário, visto na Figura 7.1. Em seguida será solicitado o arquivo com as informações da conversão de pixel para micrômetros utilizadas, como visto na Figura 8.2. Esses valores também serão escritos na saída (Figura 7.3).

Os valores de area e perimetro (em pixels) sao:

AREA(pixel2)	PERIMETRO(pixel)
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657
4	5.657

Figura 7.1: Parte arquivo "saida.dat" com as infomações de entrada

Ampliacao	FatorConversao (micrometros/pixel)
1.25	7.937
2.5	4.082
5	2.049
10	1.027
20	0.5107
40	0.2548

Figura 7.2: Arquivo com dados de conversão de unidades

A conversao utilizada segue a tabela seguinte:

AMPLIACAO	FATOR
1.25	7.937
2.5	4.082
5	2.049
10	1.027
20	0.5107
40	0.2548

Figura 7.3: Parte arquivo "saida.dat" com as infomações de conversão de unidade

O usuário será perguntado sobre a ampliação utilizada no microscópio para captura da imagem (Figura 8.3) e o fator de conversão correspondente será mostrado no arquivo de saída (Figura 7.4). Dispondo desse, os vetores de área e perímetro serão convertidos para a unidade de μm^2 e μm, respectivamente. Os novos valores serão mostrados no arquivo de saída de maneira análoga (Figura 7.5).

```
O fator de conversao é: 1.027 um/pixel
```

Figura 7.4: Fator de conversão de unidade

```
Os valores de area e perimetro (em micrometros) sao:

     AREA(um2)        PERIMETRO(um)
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
       4.05364              5.80974
```

Figura 7.5: Parte arquivo "saida.dat" com valores de área e perímetro convertidos

A soma total da área e perímetro será calculada para uso posterior. Esses valores são expostos na saída criada (Figura 7.6).

```
A soma das areas é: 257256 um^2

A soma dos perimetros é: 102327 um
```

Figura 7.6: Parte arquivo "saida.dat" com somatório dos vetores

A seguir, a porosidade da imagem será solicitada (Figura 8.5). Com essas informações, a permeabilidade pode ser estimada. Os valores calculados serão dispostos no arquivo de saída (Figura 7.7) e o valor de permeabilidade escrito no escopo do programa também (Figura 8.6).

```
O valor da superficie especifica é: 0.506448 1/um

O valor da superficie especifica por volume de grão é:
0.119337 1/um

O valor da permeabilidade (modelo Kozeny-Carman): 150.633 mD
```

Figura 7.7: Parte arquivo "saida.dat" com valores das etapas e final da permeabilidade

Para construir os gráficos é necessário um vetor com informação do raio de cada poro. Esse vetor é calculado a partir do vetor de área e mostrado no arquivo de saída (Figura 7.8).

```
O vetor de raio criado a partir dos dados da area é:
                RAIO(um)
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
                 1.13592
```

Figura 7.8: Parte arquivo "saida.dat" com os raios do poro

Dois arquivos são criados com informação necessária para elaborar os gráficos. O "raio_area.dat" (Figura 7.9) contém valores de raio e área acumulada e normalizada, que representam o gráfico de volume poroso acumulado, observado na Figura 7.10.

1.13592	0.00157572
1.13592	0.00315144
1.13592	0.00472716
1.13592	0.00630289
1.13592	0.00787861
1.13592	0.00945433
1.13592	0.01103
1.13592	0.0126058
1.13592	0.0141815
1.13592	0.0157572
1.13592	0.0173329
1.13592	0.0189087
1.13592	0.0204844
1.13592	0.0220601
1.13592	0.0236358
1.13592	0.0252115
1.13592	0.0267873
1.13592	0.028363
1.13592	0.0299387
1.13592	0.0315144
1.13592	0.0330901
1.13592	0.0346659
1.13592	0.0362416

Figura 7.9: Arquivo "raio_area.dat" gerado

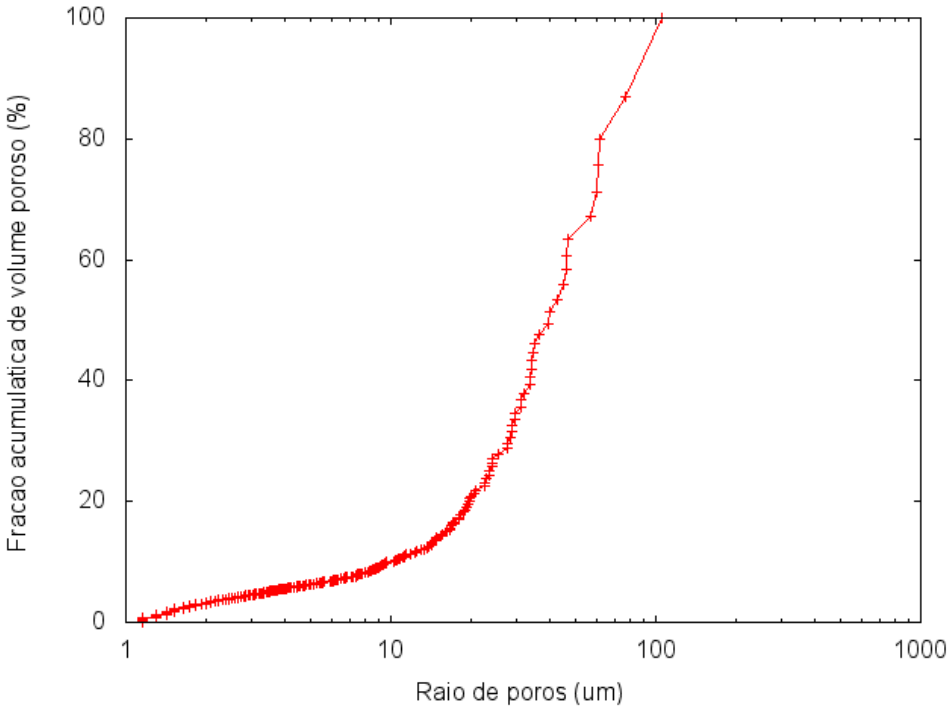

Figura 7.10: Gráfico de Fração acumulativa do volume poroso gerado pelo *software*

O outro é o "histograma.dat" (Figura 7.11), necessário para plotar a frequência de poros, visto na Figura 7.12. Os gráficos são salvos no formato *.png pelo programa.

```
    0   0.1
 1755     1
   80    10
    1   100
    0  1000
```

Figura 7.11: Arquivo "histograma.dat" gerado

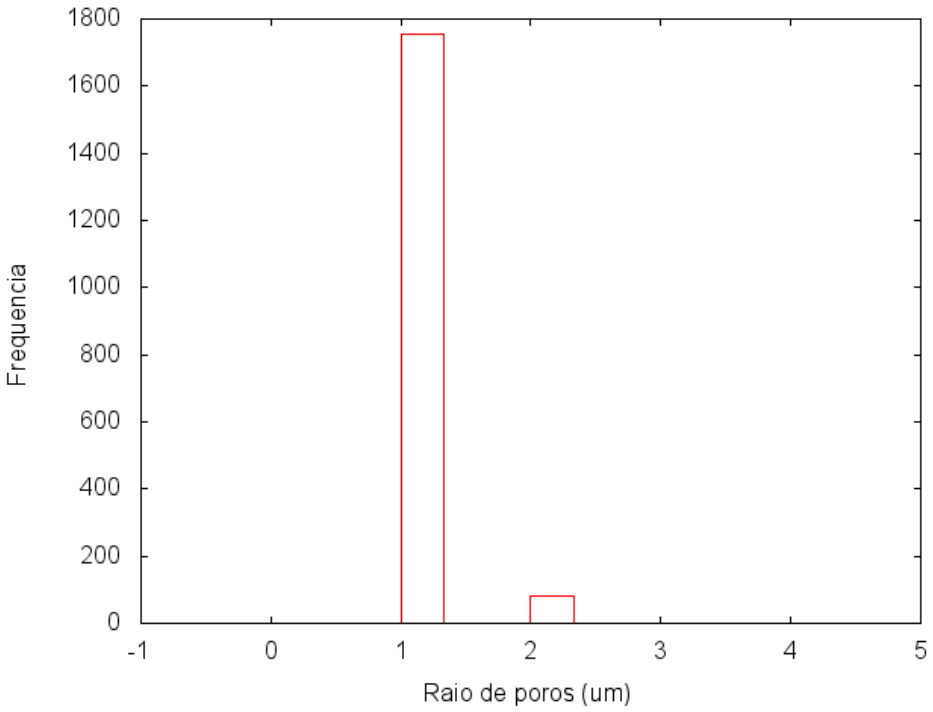

Figura 7.12: Gráfico de Frequência do tamanho de poro gerado pelo *software*

Assim, o *software* é finalizado.

7.2 Teste 2: Arquivo com dados de entrada não existente

Quando o usuário entra com um nome de arquivo inexistente ou sem a extenção do mesmo, o s*oftware* não encontra os dados de entrada e encerra o programa, como é visto na Figura 7.13.

```
Entre com o arquivo de dados exportados do ImageJ
arquivo

Falha acesso ao arquivo. Encerrando!

Process returned 0 (0x0)   execution time : 5.108 s
Press any key to continue.
```

Figura 7.13: Falha de acesso

Capítulo 8

Documentação

Todo projeto de engenharia precisa ser bem documentado. Neste sentido, apresenta-se neste capítulo a documentação de uso do *software* PropriedadesFisicas. Esta documentação tem o formato de uma apostila que explica passo a passo como usar o *software*.

8.1 Documentação do usuário

Descreve-se aqui o manual do usuário, um guia que explica, passo a passo a forma de instalação e uso do *software* desenvolvido.

8.1.1 Como instalar o software

Para instalar o *software* execute o seguinte passo a passo:

- Copie o diretório com os código para alguma pasta de destino;
- Acesse essa pasta de destino com o terminal e realize compilação e *linkagem.* Para isso é necessário que a máquina tenha algum compilador instalado (sugestão *g++*).
- O programa também pode ser aberto em algum ambiente para programação que utilize linguagem em C++ (utilizada nesse código).

8.1.2 Como rodar o software

Para rodar o *software*:

- Executar o arquivo *.out ou *.exe no terminal (Comando LINUX: *./a.out* ou comando WINDOWS: *PropriedadesFisicas*).
- Ou pode ser rodado em alguem ambiente de programação.

O programa é divido em quatro passos principais.

O primeiro solicita o arquivo de entrada *.txt com os dados dos poros (Figura 8.1).

```
Entre com o arquivo de dados exportados do ImageJ
results.txt
```

Figura 8.1: Passo 1 do programa

Em seguida é solicitado o arquivo de conversão de unidades com extensão *.txt (Figura 8.2) para definir qual fator de conversão a ser utilizado. Para isso também deve ser informado, no terceiro passo, a ampliação do microscópio utilizada para capturar a imagem (Figura 8.3).

```
Entre com o arquivo de dados exportados do ImageJ
results.txt
'mv' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

Entre com o arquivo de conversao (modelo especificado)
conversao.txt_
```

Figura 8.2: Passo 2 do programa

```
Entre com o arquivo de dados exportados do ImageJ
results.txt
'mv' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

Entre com o arquivo de conversao (modelo especificado)
conversao.txt

Qual foi a ampliacao do microscopio utilizada na lamina?
10
```

Figura 8.3: Passo 3 do programa

Caso essa ampliação não seja encontrada no banco de dados fornecido, será perguntado ao usuário diretamente qual é o valor do fator de conversão utilizado para aquela imagem (Figura 8.4).

```
Entre com o arquivo de dados exportados do ImageJ
results.txt
'mv' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

Entre com o arquivo de conversao (modelo especificado)
conversao.txt

Qual foi a ampliacao do microscopio utilizada na lamina?
17

Qual Ú o fator de conversao de pixel para micrometros do microscopio utilizado p
ara essa imagem?
1.027
```

Figura 8.4: Passo extra caso o valor da ampliação esteja fora do banco de dados

No quarto passo (Figura 8.5), a informação sobre a porosidade da amostra será requerida a fim de calcular a permeabilidade.

```
Entre com o arquivo de dados exportados do ImageJ
results.txt
'mv' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

Entre com o arquivo de conversao (modelo especificado)
conversao.txt

Qual foi a ampliacao do microscopio utilizada na lamina?
10

Qual o valor de porosidade encontrado no experimento? (ImageJ) (frac)
0.1907_
```

Figura 8.5: Passo 4 do programa

Por fim, a permeabilidade será informada na tela (Figura 8.6) e serão gerados 5 (cinco) arquivos de saída:

- saída.dat: com todas informações utilizadas nas etapas de cálculo e todas as variáveis calculadas.
- raio_area.dat: com os vetores de raio e área acumulada e normalizada da imagem, para gerar o gráfico vporoso_acumulado.png.
- histograma.dat: com informações sobre a frequência de raio dos poros em determinados intervalos especificados no arquivo, para gerar o gráfico frequencia_raio.png.
- vporoso_acumulado.png: gráfico gerado de fração acumulativa do volume poroso em porcentagem.

- frequencia_raio.png: gráfico gerado de frequência de raio de poros.

```
Entre com o arquivo de dados exportados do ImageJ
results.txt
'mv' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

Entre com o arquivo de conversao (modelo especificado)
conversao.txt

Qual foi a ampliacao do microscopio utilizada na lamina?
10

Qual o valor de porosidade encontrado no experimento? (ImageJ) (frac)
0.1907

O valor da permeabilidade (modelo Kozeny-Carman): 150.633 mD

Process returned 0 (0x0)   execution time : 20.533 s
Press any key to continue.
```

Figura 8.6: Passo final da interação do software com o usuário informando valor de permeabilidade

Veja no Capítulo 7 - Teste, exemplos de uso do *software*.

8.2 Documentação para desenvolvedor

Apresenta-se nesta seção a documentação para o desenvolvedor, isto é, informações para usuários que queiram modificar, aperfeiçoar ou ampliar este *software*.

8.2.1 Dependências

Para compilar o *software* é necessário atender as seguintes dependências:

- Instalar o compilador g++ da GNU disponível em http://gcc.gnu.org. Para instalar no GNU/Linux use o comando `yum install gcc`.
- Biblioteca CGnuplot; os arquivos para acesso a biblioteca CGnuplot devem estar no diretório com os códigos do *software*;
- O *software* `gnuplot`, disponível no endereço http://www.gnuplot.info/, deve estar instalado. É possível que haja necessidade de setar o caminho para execução do `gnuplot`.
- Obter um arquivo com dados necessários explicado no Apêndice 9.
- Necessário ter instalado *software* que forneça informações de imagens petrofíscas. *Software* sugerido: ImageJ disponível para download em imagej.nih.gov/ij/ com macro JPor encontrado em http://www.geoanalysis.org/jPOR.html.

8.2.2 Documentação usando o software Doxygen

A documentação do código do software também foi feita usando o software `doxygen` que gera a documentação do desenvolvedor no formato html. Ele lê os arquivos com os códigos (*.h e *.cpp) e gera uma documentação muito útil e de fácil navegação no formato html.

A Figura 8.7 exibe a tela do doxygen que permite a listar as classes do programa. Na Figura 8.8 é apresentada a listagem dos arquivos que compõe o programa. E na Figura 8.9 é vista as informações de uma das classes do programa, CVetorAreaPerimetro.

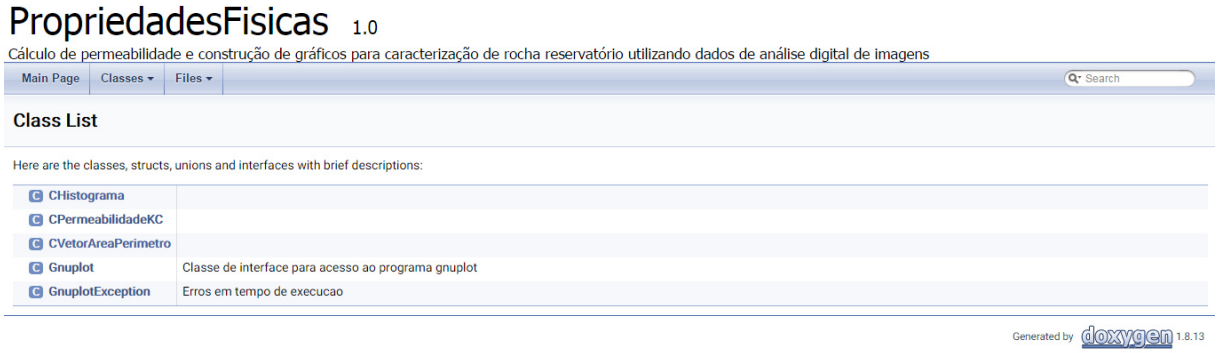

Figura 8.7: Documentação doxygen mostrando as classes do programa

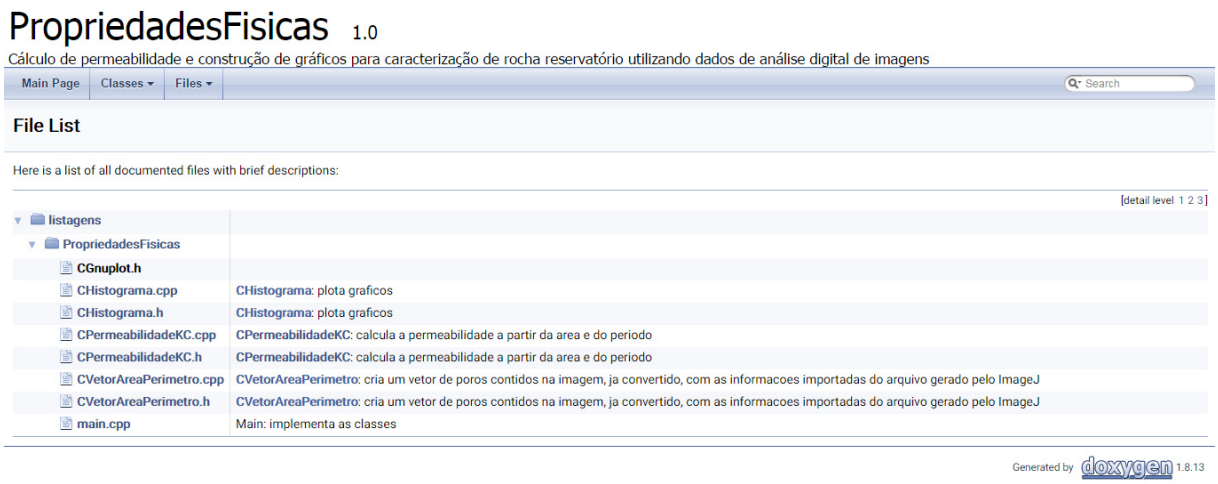

Figura 8.8: Documentação doxygen mostrando os arquivos do programa

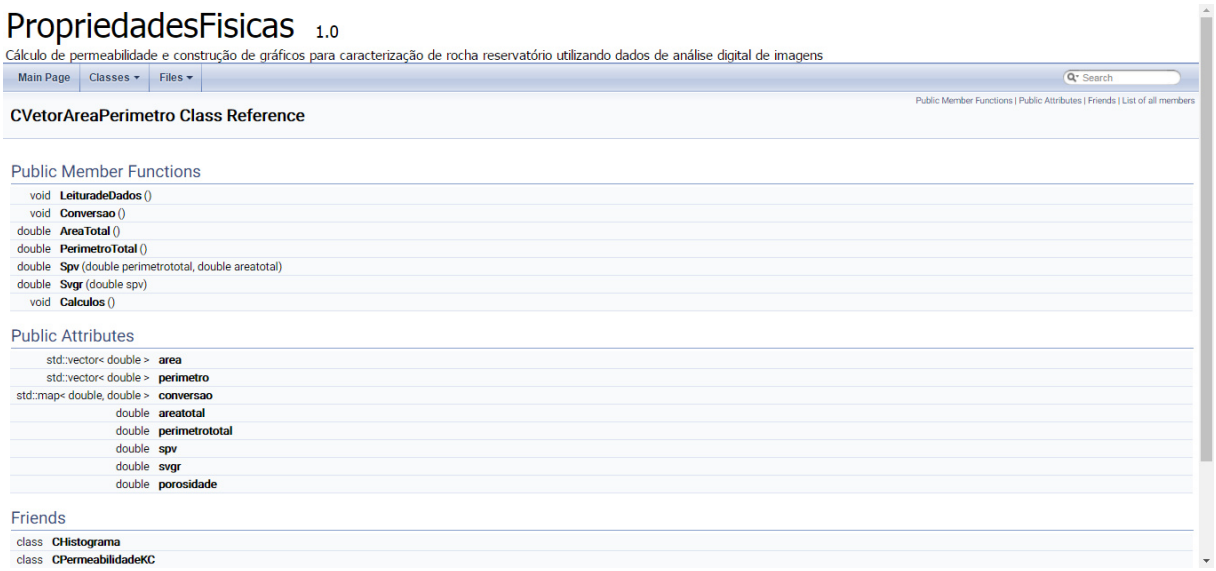

Figura 8.9: Documentação doxygen mostrando informações da classe CVetorAreaPerimetro

Referências Bibliográficas

[Blaha and Rumbaugh, 2006] Blaha, M. and Rumbaugh, J. (2006). *Modelagem e Projetos Baseados em Objetos com UML 2*. Campus, Rio de Janeiro. 16

[Cunha et al., 2012] Cunha, A. R., Moreira, A. C., Kronbauer, D. P., Mantovani, I. F., and Fernandes, C. P. (2012). Determinação de propriedades petrofísicas de rochas via simulação. um caminho interdisciplinar. *Revista Brasileira de Ensino de Física*, 34:4315. 1, 8

[Rasband, 2014] Rasband, W. (2014). Imagej - image processing and analysis in java @ONLINE. 1

[Rumbaugh et al., 1994] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., and Lorensen, W. (1994). *Modelagem e Projetos Baseados em Objetos*. Edit. Campus, Rio de Janeiro. 16

[Tiab and Donaldson, 2015] Tiab, D. and Donaldson, E. C. (2015). *Theory and Practice of Measuring Reservoir Rock and Fluid Transport Properties*. Elsevier, USA. 7, 8

[Ávila, 2017] Ávila, J. R. (2017). Tutorial para estimar prorpiedades petrofísicas das rochas através de análise digital de imagens por microscopia ótica utilizando imagej. Technical report, UENF. 49

Capítulo 9

Arquivos de entrada

Descreve-se neste apêndice como gerar arquivos compatíveis com o *software*.

9.1 Arquivo de entrada com dados de poros

O arquivo de entrada deve ter um formato único que será descrito nessa seção. Ele pode ser gerado por qualquer tratamento de imagens de rocha, desde que contenha informações sobre área, perímetro e número de referência de cada poro. A ordem é indicada na Figura 9.1, a separação das colunas podem ser com espaço ou tabulação (*default* do programa). Os dados utilizados serão, a partir da segunda linha, as colunas dois e três.

	Area	Perim.
1	383	185.765
2	9	10.485
3	4	8.485
4	10675	2865.386
5	18	36.527
6	4	8.485
7	6450	1236.511
8	5	11.314
9	2237	521.737
10	35	32.042
11	15	32.284
12	4	11.314
13	199	109.539
14	11	22.385
15	4	5.657
16	376	146.267
17	11	25.799
18	4	7.657
19	6	9.899
20	8	16.971
21	12	22.385
22	4	7.657
23	4	9.657
24	5	11.899
25	4	7.071
26	4	9.657
27	15	24.971
28	6	16.142
29	1536	654.673
30	8	15.314
31	853	279.546
32	16	32.870
33	9	16.142
34	1167	403.512
35	13	20.385
36	4	8.485
37	4	7.657
38	5	14.142
39	6	13.314
40	11	18.142
41	6	12.485

Figura 9.1: Parte do aquivo de entrada (results.txt)

9.1.1 Gerando entrada a partir do ImageJ

Aqui será descrito um tutorial para gerar as informações a partir do *software* livre ImageJ.

O primeiro passo é abrir a imagem a ser utilizada no ImageJ utilizando o *plug-in* JPor, visto na Figura 9.2.

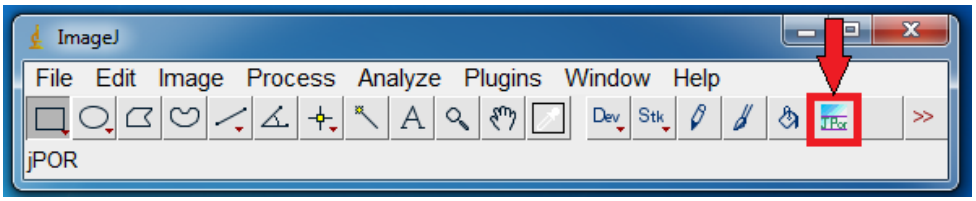

Figura 9.2: *Software* ImageJ com macro JPor

Calcule a porosidade utlizando esse ícone Figura 9.2 e a guarde para utilizá-la posteriormente no programa aqui desenvolvido. Para mais informações sobre esse cálculo leia [Ávila, 2017].

Com a imagem binarizada, faça a análise de partículas. Vá ao caminho *Analyze* -> *Set Measurements* para escolher as opções de medidas. Marque somente área e perímetro. Se necessário, redirecione à imagem que está aberta, como visto na Figura 9.3.

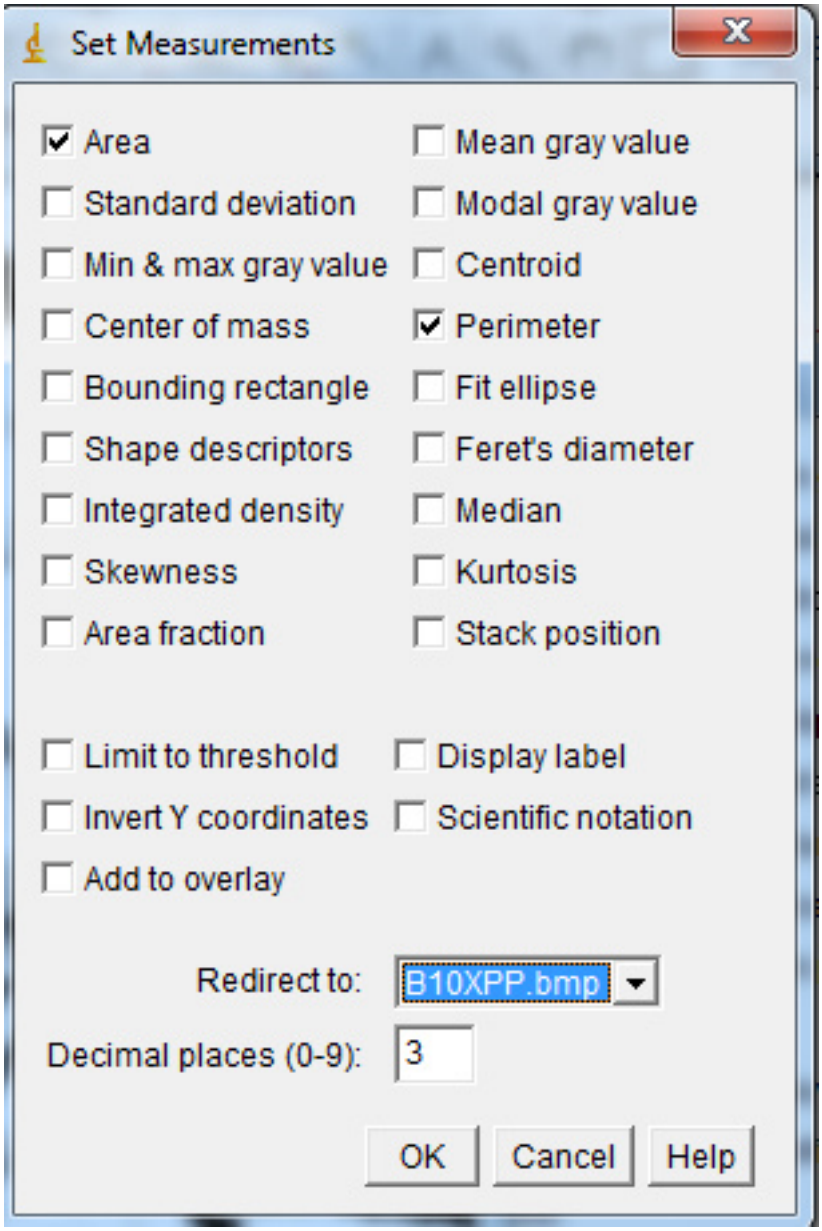

Figura 9.3: Informar medidas a serem realizadas no ImageJ

Para realizar as medidas vá em *Analyze* -> *Analyze Particles*. Escolha o intervalo de análise entre 4 (quatro) pixels até o máximo e marque para mostrar os resultados, como visto na Figura 9.4.

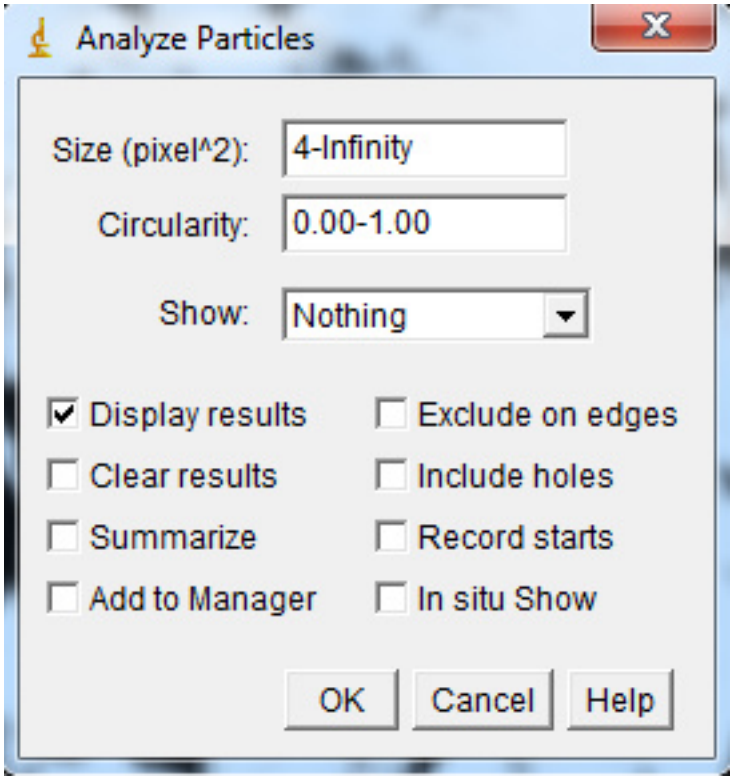

Figura 9.4: Análise das partículas da imagems

O arquivo gerado deve ser salvo em *.txt no diretório do programa, para uso de entrada.

9.2 Arquivo de entrada com dados de conversão

Aqui será descrito um tutorial para gerar as informações de conversão de pixel para unidade de comprimento.

Cada microscópio tem uma conversão para cada ampliação utilizada. Ela pode facilmente ser medida comparando a escala real da lâmina com a virtual capturada pela imagem. No software desenvolvido nesse projeto, os dados para conversão são descritos em um arquivo com extensão *.txt. O modelo deve ser igual ao da Figura 9.5, com separações com espaço ou tabulação (*default* do programa).

```
Ampliacao       FatorConversao (micrometros/pixel)
1.25    7.937
2.5     4.082
5       2.049
10      1.027
20      0.5107
40      0.2548
```

Figura 9.5: Aquivo de entrada de conversão (conversao.txt)

Índice Remissivo