

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

Versão 2 (Atualizado):
AUGUSTO CAIO ROTTE
FELLIP SILVEIRA DE ASSIS MATHIAS

Prof. André Duarte Bueno

MACAÉ - RJ
Dezembro - 2023

Sumário

1	Introdução	1
1.1	Escopo do problema	1
1.2	Objetivos	2
2	Especificação	4
2.1	Nome do sistema e produto	4
2.2	Especificação	4
2.2.1	Requisitos funcionais	5
2.2.2	Requisitos não funcionais	6
2.3	Casos de uso	7
2.3.1	Diagrama de caso de uso geral	9
2.3.2	Diagrama de caso de uso específico	10
3	Elaboração	12
3.1	Análise de domínio	12
3.2	Formulação teórica	12
3.3	Identificação de pacotes – assuntos	15
3.4	Diagrama de pacotes – assuntos	17
4	AOO – Análise Orientada a Objeto	18
4.1	Diagramas de classes	18
4.1.1	Dicionário de classes	19
4.2	Diagrama de sequência – eventos e mensagens	20
4.2.1	Diagrama de sequência geral	20
4.2.2	Diagrama de sequência específico	22
4.3	Diagrama de comunicação – colaboração	22
4.4	Diagrama de máquina de estado	23
4.5	Diagrama de atividades	23
5	Projeto	24
5.1	Projeto do sistema	24
5.2	Projeto orientado a objeto – POO	25

5.3	Diagrama de componentes	26
5.4	Diagrama de implantação	27
6	Implementação	29
6.1	Código fonte	29
7	Teste	41
7.1	Teste 1: Cálculo de permeabilidade e construção de gráficos	41
7.2	Teste 2: Arquivo com dados de entrada não existente	46
8	Documentação	48
8.1	Documentação do usuário	48
8.1.1	Como instalar o software	48
8.1.2	Como rodar o software	48
8.2	Documentação para desenvolvedor	51
8.2.1	Dependências	51
8.2.2	Documentação usando o software Doxygen	52
9	Arquivos de entrada	56
9.1	Arquivo de entrada com dados de poros	56
9.1.1	Gerando entrada a partir do ImageJ	58
9.2	Arquivo de entrada com dados de conversão	60

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 microscópica 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].

Este conjunto de análises e simulações compõem um projeto de engenharia que utiliza o paradigma da orientação a objetos.

Seguem os dados usados no Projeto: Analise Porosidade:

Realiza a análise da porosidade em amostras de lâminas petrográficas de rochas sedimentares, obtendo dados por meio da análise digital de imagens com o software ImageJ. Os resultados são utilizados para calcular a porosidade e contribuir para a caracterização do reservatório. Parametros Morfometricos:

Configura e conduz análises morfométricas nas imagens obtidas, extraindo dados como área e perímetro de poros. Esses parâmetros são fundamentais para a determinação da distribuição de tamanho de poros e para estimar a superfície de poro específica. Propriedades Rocha (homogeneidade e heterogeneidade):

Realiza análises de homogeneidade e heterogeneidade das propriedades da rocha. Isso inclui características como cor, tamanho e tipo. Os resultados dessas análises contribuem para um entendimento mais abrangente da composição da rocha. Método RbSr:

Utiliza o método RbSr para análise isotópica de rochas, obtendo dados como data de coleta, idade e relação isotópica. Essas informações são cruciais para compreender a evolução geológica e características específicas das amostras. Volume Representativo do

Elemento (REV):

Identifica e caracteriza elementos representativos do volume da rocha, essenciais para simulações futuras. Essa análise contribui para a compreensão das propriedades macroscópicas da rocha. Simulação Fluxo de Fluidos (líquido e gasoso):

Configura e conduz simulações de fluxo de fluidos, tanto líquidos quanto gasosos, utilizando as propriedades previamente analisadas. As simulações incluem a configuração de viscosidade, temperatura, compressibilidade, entre outros parâmetros relevantes.. É de grande valia determinar essas características para realizar um bom estudo do reservatório que se espera produzir. Cálculo da Área do Vetor do Perímetro:

Realiza correlações para encontrar a porosidade, a permeabilidade, a distribuição de tamanho de poros na rocha e estimativa de superfície de poro específica.

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 integrado para caracterização avançada de rochas sedimentares: O objetivo principal do projeto é criar uma ferramenta de software robusta e orientada a objetos capaz de conduzir análises aprofundadas em amostras de rochas sedimentares. O software visa integrar múltiplas classes e funcionalidades para fornecer uma caracterização completa dos reservatórios.

- Objetivos específicos:

– Realizar análises de porosidade: Desenvolver métodos específicos para conduzir análises de porosidade em amostras de lâminas petrográficas, utilizando o software ImageJ como plataforma de análise digital de imagens.

- Configurar e executar análises morfométricas para obtenção de parâmetros essenciais: Desenvolver funcionalidades na classe Parametros Morfometricos para configurar e conduzir análises morfométricas, extraindo dados cruciais como área e perímetro de poros, fundamentais para a caracterização das amostras.
- Realizar análises de homogeneidade e heterogeneidade das propriedades da rocha: Implementar métodos na classe Propriedades Rocha para realizar análises detalhadas das características homogêneas e heterogêneas, contribuindo para um entendimento mais abrangente da composição da rocha.
- Utilizar o Método RbSr para análise isotópica de rochas: Desenvolver funcionalidades na classe Metodo RbSr para configurar e conduzir análises isotópicas, obtendo dados como data de coleta, idade e relação isotópica, essenciais para compreender a evolução geológica das amostras.
- Identificar e caracterizar elementos representativos para simulações futuras: Implementar funcionalidades na classe Volume Representativo do Elemento (REV) para identificar e caracterizar elementos representativos do volume da rocha, contribuindo para simulações futuras.
- Configurar e conduzir simulações de fluxo de fluidos (líquido e gasoso): Desenvolver métodos na classe SimulacaoFluxoFluidos para configurar e conduzir simulações de fluxo, utilizando as propriedades analisadas, como viscosidade, temperatura e compressibilidade.
- Realizar correlações para determinar porosidade, permeabilidade e distribuição de tamanho de poros: Implementar funcionalidades na classe CVetorAreaPerimetro para realizar correlações essenciais para determinar porosidade, permeabilidade e distribuição de tamanho de poros nas rochas sedimentares.
- Processar dados gerados no ImageJ e transformá-los em informações cruciais: Integrar a classe CVetorAreaPerimetro ao fluxo de processamento do software, garantindo que os dados gerados no ImageJ sejam processados e transformados em informações valiosas sobre as rochas analisadas.

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 Físicas
Componentes principais	Determinação de propriedades físicas de rochas sedimentares a partir de dados área do perímetro, fluxo de fluidos, método RbSr, propriedades da rocha, porosidade, volume representativo do elemento e parâmetros morfométricos 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 tem como papel criar um software orientado a objetos capaz de realizar análises avançadas em amostras de rochas sedimentares, visando uma caracterização completa dos reservatórios petrolíferos. Cada componente do projeto desempenha um papel específico, contribuindo para uma compreensão abrangente das propriedades das rochas.

CVetorAreaPerimetro: Essencial para análises morfométricas, calcula a área e o perímetro dos poros, fornecendo dados cruciais para correlacionar características geométricas com propriedades físicas.

Analise Porosidade: Conduz análises de porosidade utilizando a técnica de Análise Digital de Imagens (ADI) com o software ImageJ, fornecendo dados para o cálculo da

porosidade.

Parametros Morfométricos: Configura e conduz análises morfométricas, extraindo dados como área e perímetro de poros, fundamentais para determinar a distribuição de tamanho de poros e estimar a superfície de poro específica.

Propriedades Rocha (homogeneidade e heterogeneidade): Realiza análises detalhadas das características homogêneas e heterogêneas da rocha, contribuindo para uma compreensão mais abrangente de sua composição.

Metodo RbSr: Utiliza o método RbSr para análise isotópica de rochas, obtendo dados cruciais, como data de coleta, idade e relação isotópica, para compreender a evolução geológica das amostras.

Volume Representativo do Elemento (REV): Identifica e caracteriza elementos representativos do volume da rocha, essenciais para simulações futuras e compreensão das propriedades macroscópicas.

SimulacaoFluxoFluidos (líquido e gasoso): Configura e conduz simulações de fluxo de fluidos, utilizando propriedades previamente analisadas, incluindo viscosidade, temperatura e compressibilidade.

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 licença GPL, podendo ser livremente distribuído e copiado.

2.2.1 Requisitos funcionais

Apresenta-se a seguir os requisitos funcionais.

RF-01	Calcular a área total dos poros na amostra.
--------------	---

RF-02	Calcular o perímetro total dos poros na amostra.
--------------	--

RF-03	Iniciar e conduzir a análise da porosidade em amostras de lâminas petrográficas.
--------------	--

RF-04	Processar os resultados da análise para calcular a porosidade da amostra.
--------------	---

RF-05	Configurar os parâmetros para análises morfométricas.
--------------	---

RF-06	Executar a análise morfométrica, extraindo dados como área e perímetro de poros.
--------------	--

RF-07	Realizar análises de homogeneidade das propriedades da rocha.
--------------	---

RF-08	Realizar análises de heterogeneidade das propriedades da rocha.
--------------	---

RF-09	Configurar os parâmetros para análise isotópica usando o método RbSr.
RF-10	Executar a análise isotópica, obtendo dados como data de coleta, idade e relação isotópica.
RF-11	Identificar elementos representativos para análises futuras.
RF-12	Caracterizar elementos identificados, contribuindo para a compreensão das propriedades macroscópicas.
RF-13	Configurar parâmetros para simulação de fluxo de fluidos líquidos.
RF-14	Configurar parâmetros para simulação de fluxo de fluidos gassosos.
RF-15	Conduzir a simulação de fluxo, incluindo viscosidade, temperatura e compressibilidade.

2.2.2 Requisitos não funcionais

RNF-01	O software deve ser capaz de processar e analisar grandes conjuntos de dados de imagem de forma eficiente, garantindo tempos de resposta rápidos para o usuário.
RNF-02	A execução das simulações de fluxo de fluidos deve ser otimizada para lidar com diferentes escalas de complexidade, mantendo a eficiência computacional.
RNF-03	A interface do usuário deve ser intuitiva, proporcionando uma experiência de usuário amigável e facilitando o uso por profissionais da indústria de petróleo.
RNF-04	O software deve fornecer feedback claro sobre o progresso das análises, simulações e resultados obtidos.
RNF-05	O software deve ser robusto e resistente a falhas, assegurando que as análises e simulações possam ser concluídas mesmo em condições adversas.

RNF-06	Em caso de interrupção inesperada, o software deve ser capaz de recuperar o estado da análise ou simulação em que estava antes da falha.
RNF-07	O acesso ao software deve ser protegido por autenticação, garantindo que apenas usuários autorizados possam realizar análises e simulações.
RNF-08	As informações sensíveis, como dados de amostras, devem ser armazenadas de forma segura e estar sujeitas a políticas de privacidade rigorosas.
RNF-09	O software deve ser compatível com diferentes sistemas operacionais, garantindo sua utilização em uma variedade de ambientes de trabalho.
RNF-10	A interoperabilidade com outros softwares amplamente utilizados na indústria de petróleo, como o ImageJ, deve ser assegurada.
RNF-11	O código-fonte do software deve ser estruturado e documentado de forma clara, facilitando a manutenção e futuras atualizações.
RNF-12	Módulos específicos do software devem ser facilmente substituíveis para permitir a incorporação de avanços tecnológicos.
RNF-13	O software deve ser projetado para otimizar o consumo de recursos computacionais, contribuindo para uma pegada ambiental reduzida durante sua execução.
RNF-14	O software deve permitir a fácil integração com outros sistemas utilizados na indústria de petróleo, possibilitando uma abordagem de trabalho mais holística.

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:	<ol style="list-style-type: none"> 1. Selecionar amostra para análise. 2. Configurar parâmetros de análise. 3. Executar análise de porosidade. 4. Selecionar amostra para análise morfométrica. 5. Configurar parâmetros de análise. 6. Executar análise morfométrica. 7. Selecionar amostra para análise de propriedades. 8. Executar análise de homogeneidade. 9. Executar análise de heterogeneidade. 10. Selecionar amostra para análise isotópica. 11. Configurar parâmetros do método RbSr. 12. Executar análise isotópica. 13. Identificar elementos representativos. 14. Caracterizar elementos identificados. 15. Configurar parâmetros da simulação. 16. Executar simulação de fluxo líquido. 17. Configurar parâmetros da simulação. 18. Executar simulação de fluxo gasoso.
Cenários alternativos:	<ol style="list-style-type: none"> 1. Se a amostra não estiver adequada para análise, o sistema notifica o usuário. 2. Se os parâmetros de análise não forem adequados, o usuário pode ajustá-los. 3. Se a análise não for conclusiva, o usuário pode repetir a análise ou verificar outras propriedades. 4. Se os resultados não forem satisfatórios, o usuário pode revisar os parâmetros ou escolher outra amostra. 5. Se a identificação não for precisa, o usuário pode ajustar critérios de seleção. 6. Se os resultados não forem satisfatórios, o usuário pode ajustar parâmetros. 7. Se os resultados não forem satisfatórios, o usuário pode ajustar parâmetros.

Tabela 2.2: Exemplo de caso de uso específico

Nome do caso de uso:	Específico
Resumo/descrição:	Determinação de cada etapa do programa.
Etapas:	<ol style="list-style-type: none"> 1. Selecionar amostra para análise. 2. Configurar parâmetros de análise. 3. Executar análise de porosidade. 4. Selecionar amostra para análise morfométrica. 5. Configurar parâmetros de análise. 6. Executar análise morfométrica. 7. Selecionar amostra para análise de propriedades. 8. Executar análise de homogeneidade. 9. Executar análise de heterogeneidade. 10. Selecionar amostra para análise isotópica. 11. Configurar parâmetros do método RbSr. 12. Executar análise isotópica. 13. Identificar elementos representativos. 14. Caracterizar elementos identificados. 15. Configurar parâmetros da simulação. 16. Executar simulação de fluxo líquido. 17. Configurar parâmetros da simulação. 18. Executar simulação de fluxo gasoso.
Cenários alternativos:	<ol style="list-style-type: none"> 1. Se a amostra não estiver adequada para análise, o sistema notifica o usuário. 2. Se os parâmetros de análise não forem adequados, o usuário pode ajustá-los. 3. Se a análise não for conclusiva, o usuário pode repetir a análise ou verificar outras propriedades. 4. Se os resultados não forem satisfatórios, o usuário pode revisar os parâmetros ou escolher outra amostra. 5. Se a identificação não for precisa, o usuário pode ajustar critérios de seleção. 6. Se os resultados não forem satisfatórios, o usuário pode ajustar parâmetros.

2.3.1 Diagrama de caso de uso geral

O diagrama de caso de uso geral da Figura 2.1 mostra o usuário gerando as análise de porosidade, análises morfométricas, avaliação de propriedades da rocha, análises isotópicas (método RbSr), identificação de elementos representativos e simulações de fluxo de fluidos. Essas funcionalidades visam proporcionar uma visão abrangente das propriedades das rochas, desde a microestrutura até as características isotópicas e de fluxo de fluidos.



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

2.3.2 Diagrama de caso de uso específico

Nesse caso do diagrama de caso de uso os usuários podem conduzir análises de porosidade, análises morfométricas, avaliação de propriedades da rocha, análises isotópicas pelo método RbSr, identificação de elementos representativos e simulações de fluxo de fluidos (líquidos e gasosos). Cada módulo oferece funcionalidades distintas, proporcionando uma abordagem abrangente para a caracterização de rochas sedimentares, desde análises microscópicas até simulações de comportamento de fluidos.. Esse processo é visto na Figura 2.2.



Figura 2.2: Diagrama de caso de uso específico – Cálculo 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;
- Estudo da permeabilidade e porosidade de amostras de lâminas petrográficas de rochas sedimentares.

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*.

$$S_{pv} = \frac{4L_p}{\pi A_p} \quad (3.1)$$

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

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

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 aproximação do diâmetro dos poros (d), e assim uma distribuição do tamanho dos poros.

$$d = 2\sqrt{\frac{A_p}{\pi}} \quad (3.4)$$

Para os demais módulos do software, como Análise de Porosidade, Parâmetros Morfométricos, Propriedades da Rocha (Homogeneidade e Heterogeneidade), Método RbSr, Volume Representativo do Elemento (REV) e Simulação de Fluxo de Fluidos (Líquidos e Gasosos), serão aplicadas técnicas e análises específicas para atingir os objetivos propostos em cada módulo, proporcionando uma abordagem holística na caracterização de rochas sedimentares.

Para o módulo de Análise de Porosidade, a abordagem se concentra na utilização de técnicas avançadas de análise digital de imagens, visando extrair informações precisas sobre a porosidade das amostras de lâminas petrográficas. Isso inclui a determinação da distribuição de tamanho de poros, permitindo uma caracterização mais detalhada dos espaços vazios na rocha.

No módulo de Parâmetros Morfométricos, a ênfase está na análise detalhada da forma e geometria dos poros. A obtenção de dados como área e perímetro de cada poro permite a avaliação da morfologia das amostras, contribuindo para uma compreensão mais profunda da estrutura da rocha.

O módulo de Propriedades da Rocha (Homogeneidade e Heterogeneidade) realiza análises específicas para avaliar a uniformidade e variação nas características físicas das rochas. A distinção entre homogeneidade e heterogeneidade proporciona insights valiosos para a compreensão da diversidade na composição das amostras.

No Método RbSr, a análise isotópica desempenha um papel crucial na determinação da idade e na relação isotópica das rochas. Essas informações são essenciais para reconstruir a história geológica das amostras, fornecendo dados valiosos para estudos de evolução temporal.

O módulo de Volume Representativo do Elemento (REV) identifica e caracteriza elementos representativos do volume da rocha, contribuindo para uma compreensão mais abrangente das propriedades macroscópicas. Essa análise é vital para garantir que as simulações subsequentes se aproximem da realidade do reservatório.

Para o módulo de Simulação de Fluxo de Fluidos (Líquidos e Gasosos), as técnicas envolvem a configuração de parâmetros como viscosidade, temperatura e compressibilidade, permitindo simulações precisas do comportamento dos fluidos nas amostras analisadas. Essas simulações são cruciais para prever o desempenho do reservatório em condições diversas.

Em conjunto, esses módulos abrangem uma abordagem holística na caracterização de rochas sedimentares, oferecendo uma gama completa de análises, desde a escala microscópica até a macroscópica, para atender às demandas da indústria de exploração de petróleo.

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

CVetorPerimetro:

- Correlações para determinar porosidade, permeabilidade e distribuição de tamanho de poros.
- Utiliza dados da análise digital de imagens (ADI) com o software ImageJ.
- Processa informações cruciais para caracterização do reservatório.

AnalisePorosidade:

- Análise específica da porosidade em lâminas petrográficas.
- Utiliza dados provenientes da ADI com o ImageJ.
- Contribui para determinação da porosidade e caracterização do reservatório.

ParametrosMorfometricos:

- Configura e conduz análises morfométricas em imagens de lâminas petrográficas.
- Extrai dados como área e perímetro de poros.
- Fundamentais para determinar distribuição de tamanho de poros e estimar superfície de poro específica.

PropriedadesRocha(Homogeneidade/Heterogeneidade):

- Realiza análises de homogeneidade e heterogeneidade.
- Inclui características como cor, tamanho e tipo.
- Contribui para entendimento abrangente da composição da rocha.

Metodo-RbSr:

- Utiliza método RbSr para análise isotópica de rochas.
- Obtém dados como data de coleta, idade e relação isotópica.
- Crucial para compreensão da evolução geológica e características das amostras.

VolumeRepresentativoElemento(REV):

- Identifica e caracteriza elementos representativos do volume da rocha.
- Essencial para simulações futuras.
- Contribui para compreensão das propriedades macroscópicas da rocha.

SimulacaoFluxoFluidos(Líquido/Gasoso):

- Configura e conduz simulações de fluxo de fluidos.
- Abrange líquidos e gasosos.
- Utiliza propriedades previamente analisadas.
- Contribui para compreensão do comportamento dos fluidos em diferentes cenários.

3.4 Diagrama de pacotes – assuntos

A representação dos pacotes se encontra na Figura 3.2.

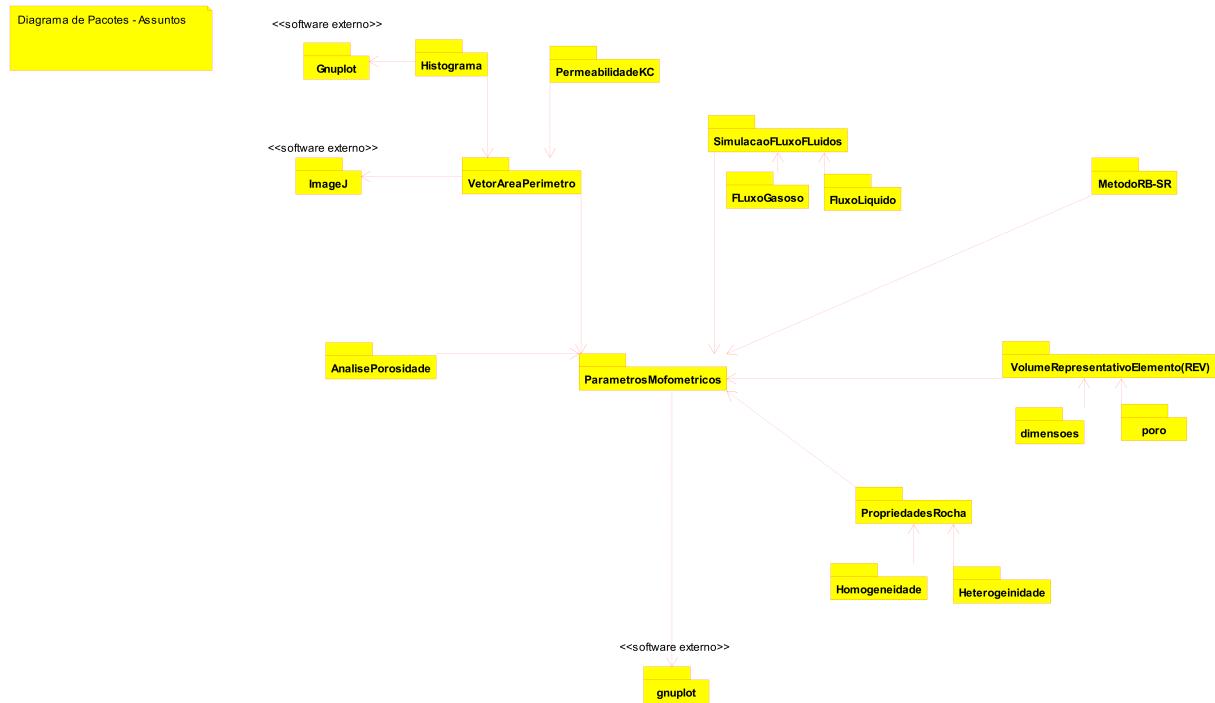


Figura 3.2: Diagrama de Pacotes

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 relações 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

- CVetorPerimetro: Módulo responsável por realizar correlações para determinar porosidade, permeabilidade e distribuição de tamanho de poros em rochas sedimentares. Utiliza dados obtidos por análise digital de imagens (ADI) com o software ImageJ, processando informações cruciais para a caracterização do reservatório.
- Atributos: are: <int> = 0, perimetro: vetor<double> = 0.0, porosidade: double = 0.0
- AnalisePorosidade: Especializado na análise da porosidade em amostras de lâminas petrográficas de rochas sedimentares. Utiliza dados provenientes da ADI com o ImageJ, contribuindo para a determinação da porosidade e, por conseguinte, para a caracterização do reservatório.
- Atributos: amostra: AmostraRocha, metodo: String , resultados: Map<String, Double>
- ParametrosMorfometricos: Configura e conduz análises morfométricas em imagens de lâminas petrográficas, extraiendo dados como área e perímetro de poros. Esses parâmetros são fundamentais para a determinação da distribuição de tamanho de poros e para estimar a superfície de poro específica.
- Atributos: amostra: AmostraRocha, metodo: String, resultados: Map<String, Double>
- PropriedadesRocha: Realiza análises de homogeneidade e heterogeneidade das propriedades da rocha, incluindo características como cor, tamanho e tipo. Os resultados dessas análises contribuem para um entendimento mais abrangente da composição da rocha.
- Atributos: nome: String, tipo: String, cor: String, descricao: String, tamanho: Double
- MetodoRbSr: Utiliza o método RbSr para análise isotópica de rochas, obtendo dados como data de coleta, idade e relação isotópica. Essas informações são cruciais para compreender a evolução geológica e características específicas das amostras.
- Atributos: dataColeta: Data, idade: Double, relacao87Sr86Sr: Double
- VolumeRepresentativoElemento (REV): Identifica e caracteriza elementos representativos do volume da rocha, essenciais para simulações futuras. Essa análise contribui para a compreensão das propriedades macroscópicas da rocha.

- Atributos: nome: String, tipoElemento: String, dimensoes: Dimensoes, porosidadeMedia: Double, permeabilidadeMedia: Double, distribuicaoPoros: List<Poro>, caracteristicasQuimicas: Map<String, Double>
- SimulacaoFluxoFluidos (Líquido e Gasoso): Configura e conduz simulações de fluxo de fluidos, abrangendo tanto líquidos quanto gasosos. Utiliza as propriedades previamente analisadas para simular condições diversas, contribuindo para a compreensão do comportamento dos fluidos em diferentes cenários.
- Atributos: propriedadesRocha: AmostraRocha

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. Segue o Dicionário:

Usuario:

Descrição: Representa um usuário do sistema que interage com os diversos módulos para realizar operações específicas.

Atributos: Nenhum atributo específico listado.

VetorAreaPerimetro:

Descrição: Classe responsável por calcular a área e o perímetro de poros em uma amostra rochosa.

Métodos: calcularArea(): double calcularPerimetro(): double

SimulacaoFluxoFluidos:

Descrição: Realiza simulações de fluxo de fluidos (líquidos e gasosos) utilizando as propriedades da amostra rochosa.

Métodos: configurarViscosidade(viscosidade: double): void configurarCompressibilidade(compressibilidade: double): void configurarTemperatura(temperatura: double): void executarSimulacao(): void

MetodoRBSR:

Descrição: Utiliza o método RbSr para análise isotópica de rochas.

Métodos: configurarDataColeta(data: Data): void configurarIdade(idade: double): void configurarRelacao87Sr86Sr(relacao: double): void

PropriedadesRocha:

Descrição: Representa as propriedades físicas e químicas de uma rocha, sendo utilizada em diversas análises.

Métodos: obterNome(): String obterTipo(): String obterCor(): String obterDescricao(): String obterTamanho(): double

AnalisePorosidade:

Descrição: Realiza a análise da porosidade em amostras de lâminas petrográficas de rochas sedimentares.

Métodos: configurarAmostra(amostra: AmostraRocha): void configurarMetodo(metodo: String): void adicionarResultado(nome: String, valor: double): void obterAmostra(): AmostraRocha obterMetodo(): String obterResultados(): Map<String, Double>

ParametrosMorfometricos:

Descrição: Realiza análises morfométricas em imagens de lâminas petrográficas, extrai dados como área e perímetro de poros.

Métodos: configurarAmostra(amostra: AmostraRocha): void configurarMetodo(metodo: String): void adicionarResultado(nome: String, valor: double): void obterAmostra(): AmostraRocha obterMetodo(): String obterResultados(): Map<String, Double>

VolumeRepresentativoElemento:

Descrição: Identifica e caracteriza elementos representativos do volume da rocha, essenciais para simulações futuras.

Métodos: obterNome(): String obterTipoElemento(): String obterDimensoes(): Dimensoes obterPorosidadeMedia(): double obterPermeabilidadeMedia(): double obterDistribuicaoPoros(): List<Poro> obterCaracteristicasQuimicas(): Map<String, Double>



Figura 4.2: Diagrama de sequência

4.2.2 Diagrama de sequência específico

Veja o diagrama de sequência específico na Figura 4.3.

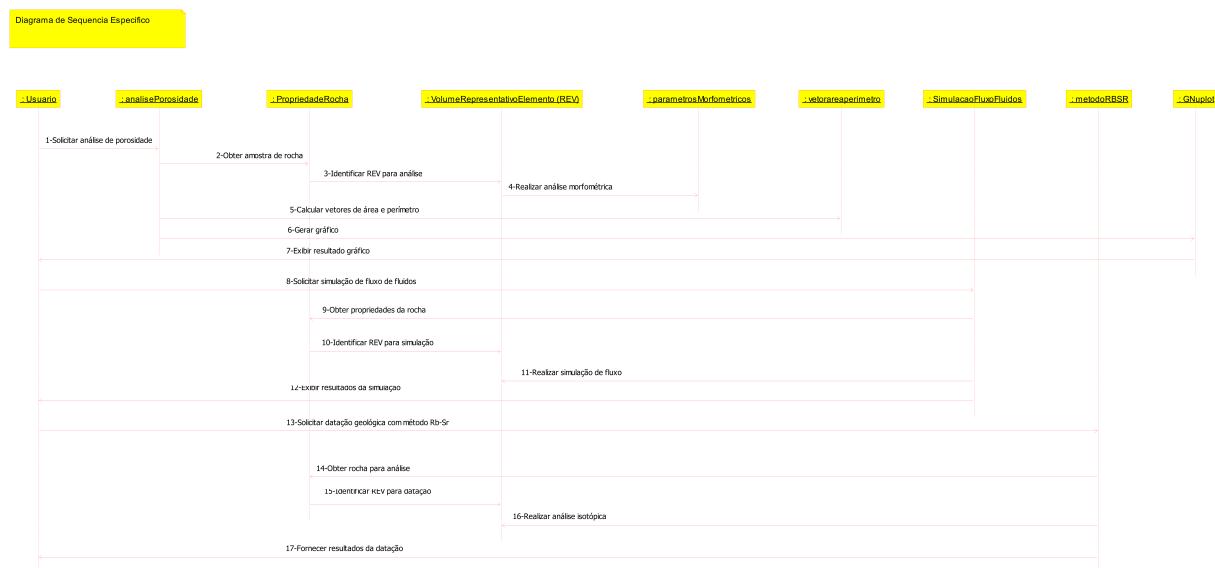


Figura 4.3: Diagrama de sequência específico

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.

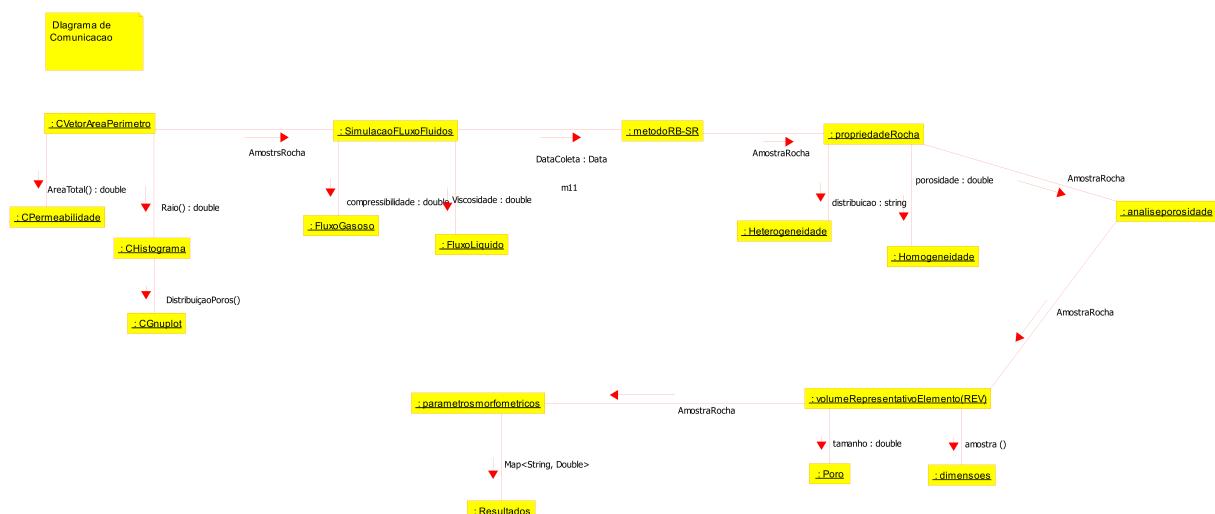


Figura 4.4: Diagrama de comunicação

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.

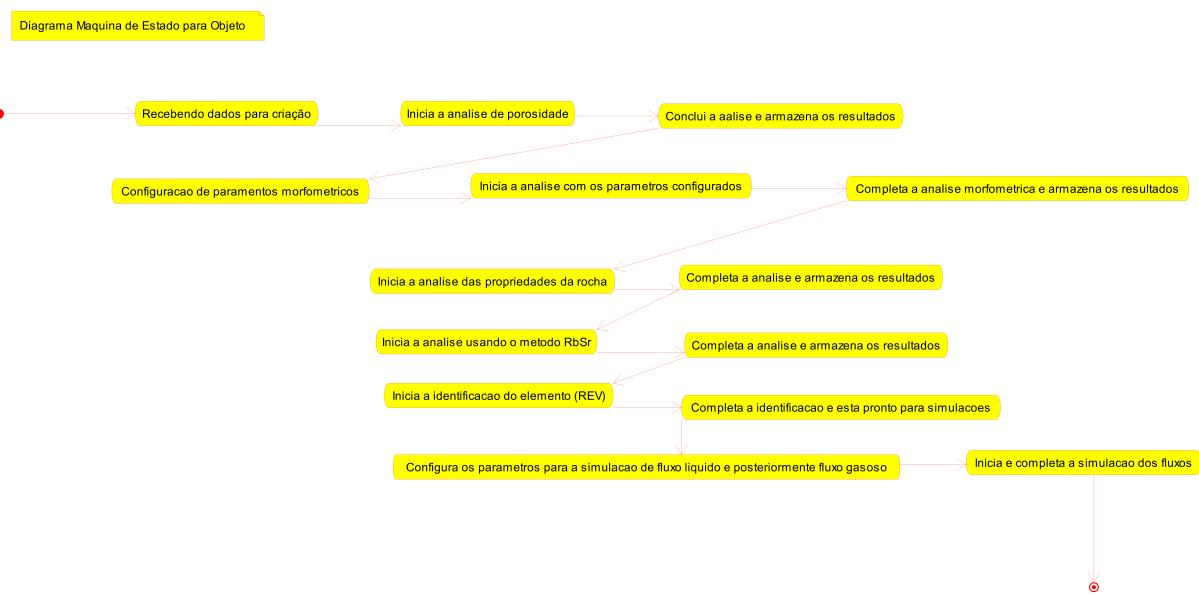


Figura 4.5: Diagrama de máquina de estado do objeto

4.5 Diagrama de atividades

Veja na Figura 4.6 o diagrama de atividades correspondente a uma atividade específica do diagrama de máquina de estado.

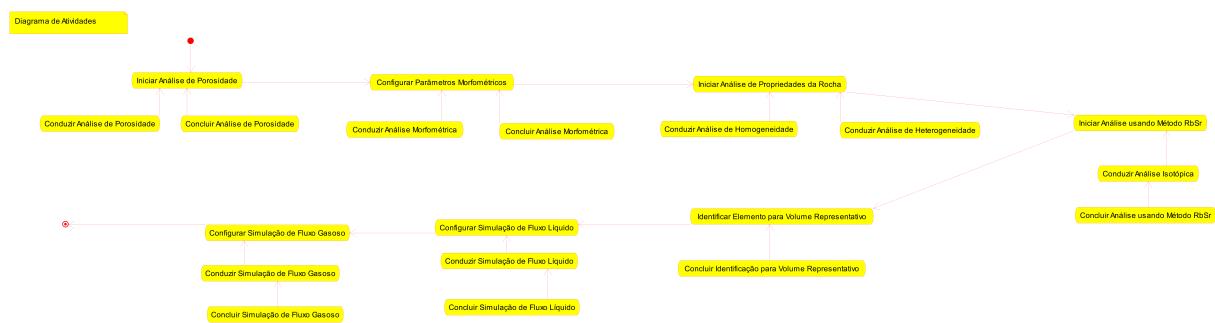


Figura 4.6: Diagrama de atividades

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 utilização 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 seguintes 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.

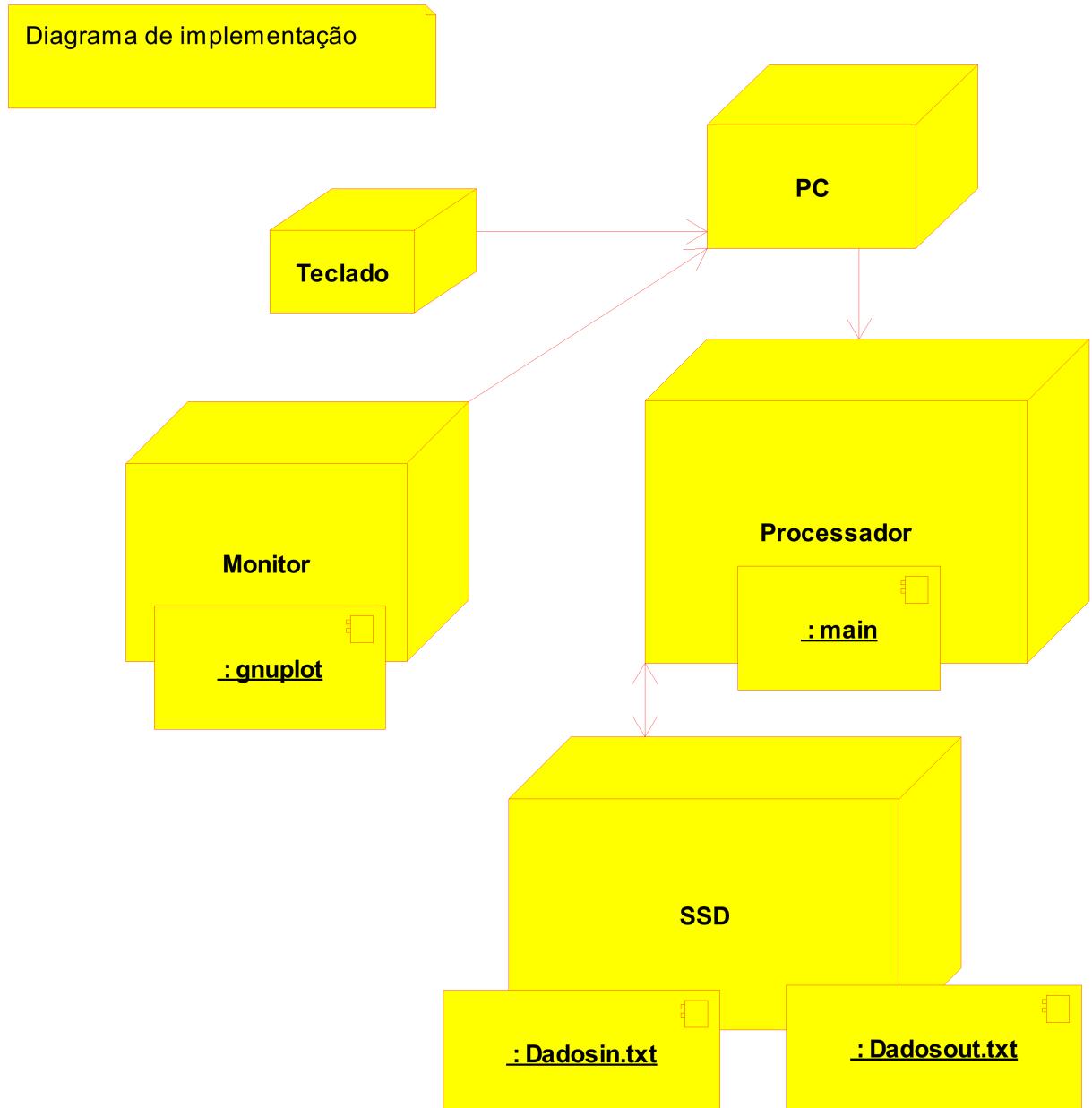


Figura 5.2: Diagrama de implantação

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 @author Juliana Avila
3 @file CVetorAreaPerimetro.h
4 @brief CVetorAreaPerimetro: cria um vetor de poros contidos na imagem,
5        ja convertido, com as informacoes importadas do arquivo gerado pelo
6        ImageJ
7 */
8
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),
33         perimetrototal(0.0), spv(0.0), svgr(0.0) {}
34
35     //construtor de copia
36     //CVetorAreaPerimetro(const CVetorAreaPerimetro& image) {
37         //area=image.area; perimetro=image.
38         //perimetro; conversao=image.conversao;
39         //areatotal=image.areatotal;
40         //perimetrototal=image.perimetrototal;}
41
42     //construtor sobrecarregado
43     //CImagenBinarizada(std::vector<int> _area, std::vector<double>
44         //_perimetro, map<double,double> _conversao, double _so, double
45         //_areatotal, double _perimetrototal):
46         //area(_area), perimetro(_perimetro),
47         //conversao(_conversao), areatotal(
48         //_areatotal), perimetrototal(
49         //_perimetrototal) {}
50
51     //destrutor
52     ~CVetorAreaPerimetro() {};
53
54     void LeituraDados();
55     void Conversao(); //usar arquivo para conversao do tipo
56         especificado
57     double AreaTotal();
58     double PerimetroTotal();
59
60     double Spv(double perimetrototal, double areatotal) {return spv
61         =((4.0*perimetrototal)/(M_PI*areatotal));}
62
63     double Svgr(double spv) {return svgr=(spv*(porosidade/(1.0-
64         porosidade)));}
65
66     void Calculos();
67
68     friend class CHistograma;
69     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 @author 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>
73using namespace std;
74
75void CVetorAreaPerimetro::LeituraDados() {
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 no 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
105    //antigo para trás
106    fout.open("saida.dat"); //vai para o fim do arquivo
107    fout << "\nOs valores de area e perimetro (em pixels) são (foram
108    //ordenados):\n" << endl;
109    fout << "\n" << setw(20) << "AREA(pixel2)" << setw(20) << "PERIMETRO
110    (pixel)" << endl;
111    for (int i=0; i<area.size(); i++) {
112        fout << setw(20) << area[i] << setw(20) << perimetro[i] << endl;
113    }
114
115 void CVetorAreaPerimetro::Conversao() {
116     //arquivo de conversão de unidade
117     string nomearquivo;
118     cout << "\nEntre com o arquivo de conversão (modelo especificado)"
119     << endl;
120     getline (cin, nomearquivo);
121     ifstream in (nomearquivo.c_str());
122     if (!in) {
123         cout << "\nArquivo não encontrado" << endl; }
124
125     double ampliacao;
126     double fator;
127     in.ignore (100, '\n'); //ignora a primeira linha
128     while (!in.eof()) {
129         in >> ampliacao;
130         in >> fator;
131         conversao.insert(make_pair(ampliacao, fator));
132     }
133     in.close();
134
135     //conferindo se deu certo
136     ofstream fout;
137     fout.open("saida.dat", ios::app); //vai para o fim do arquivo
138     map<double,double>::const_iterator it;
139     fout << "\nA conversão utilizada segue a tabela seguinte:" <<
140     endl;
141     fout << "\n" << setw(20) << "AMPLIACAO" << setw(20) << "FATOR" <<
142     endl;
143     for(it=conversao.begin(); it!=conversao.end(); it++) {
144         fout << setw(20) << it->first << setw(20) << it->second <<
145         endl;
146 }
```

```

        endl; }

140
141     // pegando o valor da ampliação e armazenando valor do fator de
142     // conversão
143     double amp;
144     cout << "\nQual foi a ampliação do microscópio utilizada na lâmina
145     ?" << endl;
146     cin >> amp;
147     double fat;
148     // precisa verificar se chegou ao fim do map e não encontrou o
149     // fator; ocorre quando o usuário seleciona um valor de ampliação
150     // que não está no arquivo. Neste caso pode pedir para usuário
151     // entrar com o fator.
152     if (conversao.find(amp) != conversao.end()) {
153         map<double, double>::const_iterator iter;
154         iter=conversao.find(amp);
155         fat=iter->second;
156         fout << "\nO fator de conversão é:" << fat << " um/pixel" <<
157         endl; }
158     else {
159         cout << "\nQual é o fator de conversão de pixel para micrometros
160         do microscópio utilizado para essa imagem?" << endl;
161         cin >> fat; }

162
163     // converte os valores
164     for (int i=0; i<area.size(); i++) {
165         area[i]=area[i]*fat*fat; //pow(x,2)
166         }
167     for (int i=0; i<perimetro.size(); i++) {
168         perimetro[i]=perimetro[i]*fat;
169         }

170
171     // conferindo
172     //fout.open("saída.dat", ios::app);           // vai para o fim do
173     // arquivo
174     fout << "\nOs valores de área e perímetro (em micrometros) são:\n"
175     << endl;
176     fout << "\n" << setw(20) << "ÁREA(um²)" << setw(20) << "PERÍMETRO(
177     um)" << endl;
178     for (int i=0; i<area.size(); i++) {
179         fout << setw(20) << area[i] << setw(20) << perimetro[i] <<
180         endl;
181     }
182
183 double CVetorAreaPerimetro::AreaTotal() {
184     areatotal=0.0;
185     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" <<
195             endl;
196
197     return perimetrototal;
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"
203             << endl;
204     fout << "\nO valor da superficie especifica por volume de grão é:"
205             " << svgr << " 1/um" << endl;
206 }
```

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

Listing 6.3: Arquivo de cabeçalho da classe CPermeabilidadeKC.

```

206 /**
207 @author Juliana Avila
208 @file CPermeabilidadeKC.h
209 @brief CPermeabilidadeKC: calcula a permeabilidade a partir da area e do
210 periodo
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
235         =(((1.0/(5.0*(image.svgr*image.svgr)))*((image.porosidade*
236             image.porosidade*image.porosidade)/((1.0-image.porosidade)
237             *(1.0-image.porosidade))))/0.00098717);}
238
239     double operator()(CVetorAreaPerimetro image) {return kkc; }
240
241 void SaidadeCalculo();
242 };
243 #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 @author Juliana Avila
244 @file CPermeabilidadeKC.cpp
245 @brief CPermeabilidadeKC: calcula a permeabilidade a partir da area e do
246 periodo
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 << "\nO valor da permeabilidade (modelo Kozeny-Carman): "
257         << kkc << " mD" << endl;

```

```

257     cout << "\n0 valor da permeabilidade (modelo Kozeny-Carman): " <<
258         kkc << " mD" << endl;
259 }
```

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

Listing 6.5: Arquivo de cabeçalho da classe CHistograma.

```

259 /**
260 @author 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 sobreescarregado
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 @author 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
319     //CVetorAreaPerimetro
320     for (int i=0; i<image.area.size(); i++) {
321         raio.push_back(sqrt(image.area[i]/M_PI));
322     }
323     //conferindo se deu certo e mostrando na tela o vetor criado
324     ofstream fout;
325     fout.open("saida.dat", ios::app); //vai para o fim do arquivo
326     fout << "\nO vetor de raio criado a partir dos dados da area é:"
327     << endl;
328     fout << setw(30) << "RAIO(um)" << endl;
329     for (int i=0; i<raio.size(); i++) {
330         fout << setw(30) << raio[i] << endl;
331     }
332 }
333
334 void CHistograma::DistribuicaoPoros() {
335     vector<int> histograma (5,0);
336     for (int i=0; i<raio.size(); i++) {
337         if (raio[i]<=1.0) histograma[0]++;
338         else if (raio[i]>=1.0 && raio[i]<10.0) histograma[1]++;
339         else if (raio[i]>=10.0 && raio[i]<100.0) histograma[2]++;
340         else if (raio[i]>=100.0 && raio[i]<1000.0) histograma[3]++;
341         else histograma[4]++;
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 << " = " <<
354         hist_x[i] << endl;
355     }
356     CGnuplot gdp; // Construtor (grafico distribuicao tamanho de poros
357     )
358     gdp.set_style("histograms");
359     gdp.XLabel("Raio de poros (um)"); // Rotulo eixo x
360     gdp.YLabel("Frequencia"); // Rotulo eixo y
361     gdp.YAutoscale();
362     gdp << "set term png\n"; // terminal é arquivo png
363     gdp << "set out \"frequencia_raio.png\"\n"; // nome do arquivo
364     gdp.plotfile_x("histograma.dat", 1, ""); // 1:xtic(2)
365     gdp << "xtic(2)\n";
366     //gdp.PlotVector(raio, "Distribuicao de tamanho de poros");
367     //cin.get();
368 }
369
370 void CHistograma::VPorosoAcumulativo(CVetorAreaPerimetro& image) {
371     //cria vetor area acumulada
372     vector<double> areaA;
373     areaA.push_back(image.area[0]);
374     for (int i=1; i<image.area.size(); i++) {
375         areaA.push_back(areaA[i-1]+image.area[i]);
376     }
377
378     //cria vetor area acumulada normalizada (%)
379     vector<double> areaAN;
380     for (int i=0; i<areaA.size(); i++) {
381         areaAN.push_back((areaA[i]/image.reatotal)*100.0);
382     }
383
384     //conferindo se deu certo
385     ofstream fout;
386     fout.open("saida.dat", ios::app);
387     fout << "\nO vetor de area tratado é:" << endl;
388     fout << setw(30) << "\nAREA ACUMULADA NORMALIZADA (um2)" << endl;
389     for (int i=0; i<areaAN.size(); i++) {
390         fout << setw(30) << areaAN[i] << endl;
391     }

```

```

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] <<
395             endl;
396     raio_area.close();
397
398     //plotando com gnuplot
399     CGnuplot gva; // Construtor (grafico volume acumulado)
400     gva.Style("linespoints");
401     gva.XLabel("Raio de poros (um)"); // Rotulo eixo x
402     gva.YLabel("Fracao acumulativa de volume poroso (%)"); // Rotulo
403         eixo y
404     gva.set_xlogscale(10);
405     gva.XAutoscale();
406     gva.YRange(0, 100);
407     gva << "set term png\n"; // terminal é arquivo png
408     gva << "set out \"vporoso_acumulado.png\"\n"; // nome do arquivo
409     gva.plotfile_xy("raio_area.dat", 1,2);
410     //gva.PlotVector(raio, areaAN, "Distribuicao acumulativa de
411         tamano de poros");
412     //cin.get();
413 }
```

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

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

```

411 /**
412 @author 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
426         e perimetro
427     image.LeituraDados(); //preenche e mostra os vetores area e
428         perimetro em pixel
429     image.Conversao(); //pede o arquivo de conversao, le e mostra ele
430         na tela
431             //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 encontrada no experimento?"_
432         (ImageJ)_(frac) << endl;
433     cin >> image.porosidade;
434     image.Spv(image.perimetrototal, image.reatotal); //calcula
435         superficie especifica do poro total
436     image.Svgr(image.spv); //calcula area especifica do poro total
437     image.Calculos();
438
439     CPermeabilidadeKC perme;
440     perme.PermeabilidadeKC(image); //calcula permeabilidade
441     perme.SaidadeCalculo(); //mostra na tela os valores calculados
442         acima
443
444     CHistograma hist; //cria o vetor de raio a partir do vetor area
445         da classe CVetorAreaPerimetro
446     hist.Raio(image);
447     hist.DistribuicaoPoros(); //plota um histograma com os tamanhos
448         do poro
449     hist.VPorosoAcumulativo(image); //plota um grafico de poros pela
450         acumulacao do volume (area)
451 }
```

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 “saída.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 (pixel)	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
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 (um ²)	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
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 um²

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  
1.13592  
1.13592  
1.13592  
1.13592  
1.13592  
1.13592
```

Figura 7.8: Parte arquivo “saída.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 extensão do mesmo, o *software* 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 algum ambiente de programação.

O programa é dividido 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 é 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</frac>
```

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 `sudo 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 petrofísicas. *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.

PropriedadesFisicas 1.0

Cálculo de permeabilidade e construção de gráficos para caracterização de rocha reservatório utilizando dados de análise digital de imagens

Class	Description
<code>CHistograma</code>	
<code>CPermeabilidadeKC</code>	
<code>CVetorAreaPerimetro</code>	
<code>Gnuplot</code>	Classe de interface para acesso ao programa gnuplot
<code>GnuplotException</code>	Erros em tempo de execucao

Generated by [doxygen](#) 1.8.13

Figura 8.7: Documentação doxygen mostrando as classes do programa

PropriedadesFisicas 1.0

Cálculo de permeabilidade e construção de gráficos para caracterização de rocha reservatório utilizando dados de análise digital de imagens

File	Description
<code>listagens</code>	
<code>PropriedadesFisicas</code>	
<code>CGnuplot.h</code>	
<code>CHistograma.cpp</code>	<code>CHistograma</code> : plota graficos
<code>CHistograma.h</code>	<code>CHistograma</code> : plota graficos
<code>CPermeabilidadeKC.cpp</code>	<code>CPermeabilidadeKC</code> : calcula a permeabilidade a partir da area e do periodo
<code>CPermeabilidadeKC.h</code>	<code>CPermeabilidadeKC</code> : calcula a permeabilidade a partir da area e do periodo
<code>CVetorAreaPerimetro.cpp</code>	<code>CVetorAreaPerimetro</code> : cria um vetor de poros contidos na imagem, ja convertido, com as informacoes importadas do arquivo gerado pelo ImageJ
<code>CVetorAreaPerimetro.h</code>	<code>CVetorAreaPerimetro</code> : cria um vetor de poros contidos na imagem, ja convertido, com as informacoes importadas do arquivo gerado pelo ImageJ
<code>main.cpp</code>	<code>Main</code> : implementa as classes

Generated by [doxygen](#) 1.8.13

Figura 8.8: Documentação doxygen mostrando os arquivos do programa

PropriedadesFisicas 1.0

Cálculo de permeabilidade e construção de gráficos para caracterização de rocha reservatório utilizando dados de análise digital de imagens

Main Page Classes ▾ Files ▾ Search Public Member Functions | Public Attributes | Friends | List of all members

CVetorAreaPerimetro Class Reference

Public Member Functions

```
void LeituraDados()
void Conversao()
double AreaTotal()
double PerimetroTotal()
double Spv(double perimetrototal, double areatotal)
double Svgr(double spv)
void Calculos()
```

Public Attributes

```
std::vector<double> area
std::vector<double> perimetro
std::map<double, double> conversao
    double areatotal
    double perimetrototal
    double spv
    double svgr
    double porosidade
```

Friends

```
class CHistograma
class CPermeabilidadeKC
```

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. 24
- [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, 13
- [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 Lohrensen, W. (1994). *Modelagem e Projetos Baseados em Objetos*. Edit. Campus, Rio de Janeiro. 24
- [Tiab and Donaldson, 2015] Tiab, D. and Donaldson, E. C. (2015). *Theory and Practice of Measuring Reservoir Rock and Fluid Transport Properties*. Elsevier, USA. 12, 13
- [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. 58

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.

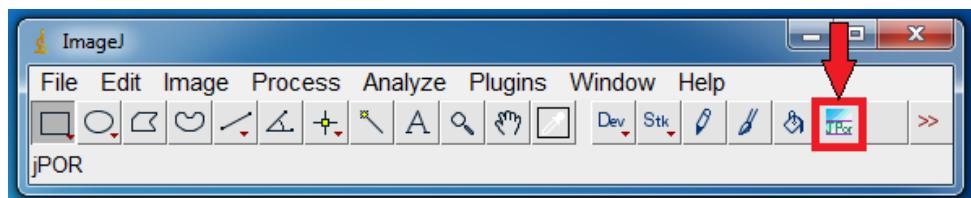


Figura 9.2: *Software* ImageJ com macro JPor

Calcule a porosidade utilizando 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.

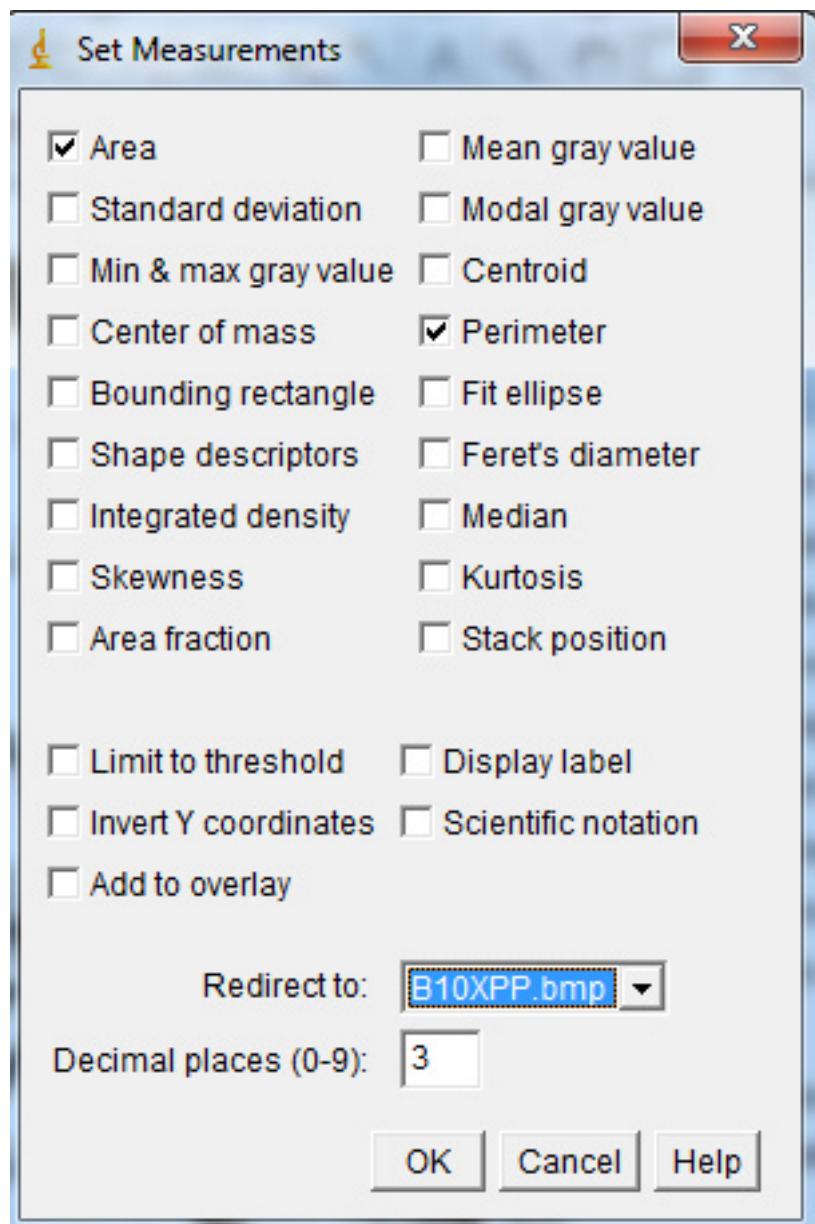


Figura 9.3: Informar medidas a serem realizadas no ImageJ

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.

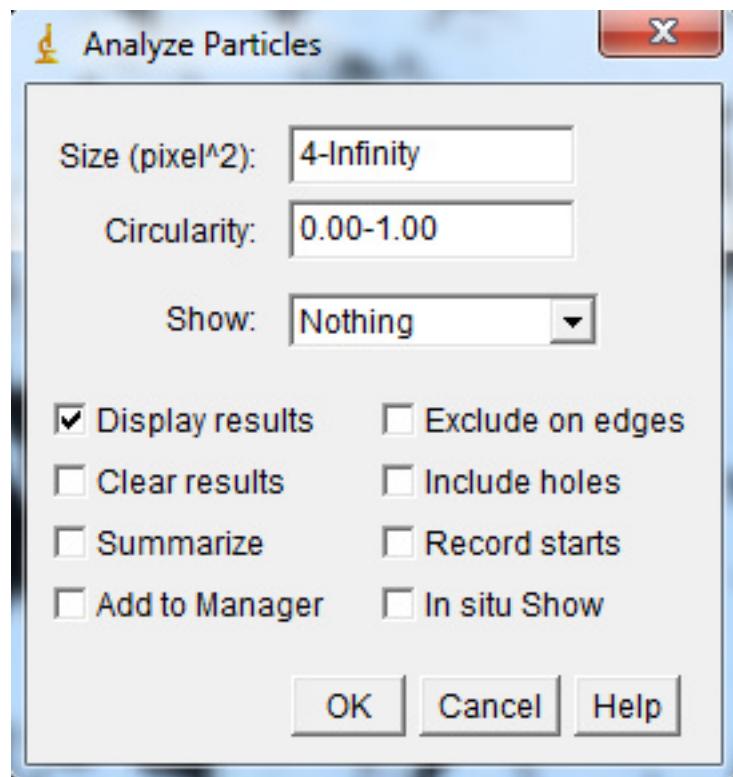


Figura 9.4: Análise das partículas da imagems

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: Arquivo de entrada de conversão (conversao.txt)