Relatório de Processamento de Imagens com HPC usando CellProfiler

8 de novembro de 2024

Resumo

Este relatório detalha o processo de configuração e execução do pipeline de análise de viabilidade celular utilizando o HPC Marvin e o software CellProfiler. As instruções incluem opções para rodar o pipeline tanto no ambiente HPC Marvin quanto no computador pessoal do usuário, com ajustes necessários para ambos os casos.

1 Introdução

O processamento de imagens em biologia celular permite extrair dados quantitativos essenciais. Neste projeto, utilizamos o HPC Marvin e o CellProfiler para analisar imagens de viabilidade celular. O objetivo deste relatório é fornecer instruções detalhadas para configurar, clonar e executar o pipeline de análise.

2 Requisitos do Projeto

Para reproduzir esta análise, são necessários:

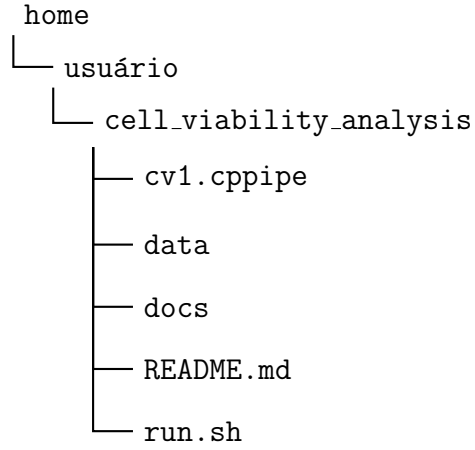
- Acesso ao HPC Marvin (se for rodar remotamente).
- Computador pessoal com o CellProfiler instalado (se for rodar localmente). Caso precise de ajuda para instalar o CellProfiler em seu computador, consulte o manual disponível em: `https://github.com/cnpem/lnbio-bioimage-analysis/blob/main/cellprofiler/INSTALLATION.md`.
- Conhecimento básico em manipulação de arquivos no Linux.

- Familiaridade com o SLURM para execução de jobs em HPC (para rodar no Marvin).

3 Estrutura do Projeto

Para analisar a viabilidade celular e outros aspectos relevantes em imagens biológicas utilizando o aplicativo CellProfiler. O projeto foi desenvolvido com base em conceitos fundamentais de processamento e análise de imagens, abordando a importância de formatos como TIFF, PNG e JPEG, e suas características específicas. Utilizando esses conhecimentos, buscamos automatizar a segmentação e contagem de células viáveis.

A estrutura de diretórios do projeto após a clonagem é apresentada na árvore de arquivos a seguir:



Nesta seção, descrevemos o propósito de cada arquivo e diretório na estrutura do projeto:

- `cv1.cppipe`: Este arquivo contém a definição do projeto do CellProfiler. Ele especifica os módulos, configurações e parâmetros necessários para realizar a análise de viabilidade celular.

- `data`: Este diretório é utilizado para armazenar as imagens `.tiff` que serão processadas pelo CellProfiler. É fundamental que as imagens sejam colocadas nesta pasta para que o pipeline funcione corretamente.

- `docs`: Este diretório que tem documentação , guias relacionadas ao projeto

- `README.md`: Este arquivo fornece uma visão geral do projeto, incluindo instruções sobre como configurar e executar o pipeline. É um ponto

de partida importante para novos usuários e para aqueles que precisam entender a estrutura do projeto rapidamente.

- `run.sh`: Este script shell contém os comandos necessários para executar o pipeline do CellProfiler usando o SLURM. Ele define a alocação de recursos e executa a análise com base nas imagens e no projeto especificado.

4 Acessando o HPC Marvin via PowerShell

Para rodar o pipeline no HPC Marvin, é necessário primeiro acessar o servidor remoto usando o SSH pelo PowerShell no Windows. Siga os passos abaixo para realizar o acesso:

1. Abra o PowerShell no seu computador com Windows. Você pode encontrar o PowerShell digitando "PowerShell"na barra de busca do sistema.

2. No PowerShell, use o seguinte comando para acessar o HPC Marvin:

   ```
   ssh seu_usuario@marvin.cnpem.br
   ```

   Substitua `seu_usuario` pelo seu nome de usuário no HPC Marvin. Após inserir o comando, será solicitado que você insira sua senha. Digite sua senha e pressione `Enter`.

Após esses passos, você estará dentro do ambiente do HPC Marvin e poderá prosseguir com as próximas etapas do processamento de imagens.

5 Clonando o Diretório de Projeto

Para configurar o ambiente inicial, é necessário clonar o repositório que contém os arquivos necessários do GitHub para a pasta principal do usuário no HPC Marvin ou a pasta de que deseja rodar em seu computador. As instruções são apresentadas abaixo:

```
# Voltar ao diretório home no hpc
cd ~
```

```
# Clone o repositório
git clone https://github.com/cnpem/ILUM-20241479.git
↪  cell_viability_analysis
```

O comando acima cria uma pasta chamada `cell_viability_analysis` no diretório principal do usuário. Este diretório conterá todos os arquivos necessários para a execução do pipeline, incluindo o script `run.sh`, que automatiza a execução do job no SLURM.

5.1 Clonando o Repositório no Windows

Se você estiver utilizando o Windows, siga os passos abaixo para clonar o repositório no seu computador pessoal:

1. Baixe e instale o **Git** no seu computador, caso ainda não tenha. O Git pode ser baixado em `https://git-scm.com/download/win`.
2. Após a instalação, abra o **Git Bash**, que foi instalado junto com o Git.
3. Navegue até o diretório onde deseja clonar o repositório. Por exemplo, para clonar na área de trabalho, use o comando:

```
cd ~/Desktop
```

4. Clone o repositório com o comando:

```
git clone https://github.com/cnpem/ILUM-20241479.git
↪  cell_viability_analysis
```

5. O comando criará uma pasta chamada `cell_viability_analysis` no diretório especificado. Agora, você pode acessar essa pasta para trabalhar com os arquivos do repositório.

6 Acessando o Diretório `cell_viability_analysis`

Para acessar a pasta `cell_viability_analysis` onde o projeto foi clonado, siga os passos abaixo:

6.1 Acessando no HPC Marvin

No terminal do HPC Marvin, execute o seguinte comando:

```
# Acesse o diretório do projeto
cd ~/cell_viability_analysis
```

Dentro dessa pasta, você terá acesso a todos os arquivos necessários para a análise, incluindo o pipeline e o script `run.sh`.

6.2 Acessando no Windows

Se você clonou o repositório no seu computador com Windows, siga os passos abaixo para acessar o diretório:

1. Abra o **Git Bash**, que você já instalou durante o processo de clonagem.
2. Navegue até a pasta onde o repositório foi clonado. Se você clonou na área de trabalho, use o seguinte comando para navegar até lá:

```
cd ~/Desktop/cell_viability_analysis
```

3. Agora você está dentro do diretório `cell_viability_analysis` e pode acessar todos os arquivos necessários para a análise.

7 Acessando o Diretório `cell_viability_analysis`

Para acessar a pasta `cell_viability_analysis` onde o projeto foi clonado, siga os passos abaixo:

7.1 Acessando no HPC Marvin

No terminal do HPC Marvin, execute o seguinte comando:

```
# Acesse o diretório do projeto
cd ~/cell_viability_analysis
```

Dentro dessa pasta, você terá acesso a todos os arquivos necessários para a análise, incluindo o pipeline e o script `run.sh`.

7.2 Acessando no Windows

Se você clonou o repositório no seu computador com Windows, siga os passos abaixo para acessar o diretório:

1. Abra o **Git Bash**, que você já instalou durante o processo de clonagem.
2. Navegue até a pasta onde o repositório foi clonado. Se você clonou na área de trabalho, use o seguinte comando para navegar até lá:

```
cd ~/Desktop/cell_viability_analysis
```

3. Agora você está dentro do diretório `cell_viability_analysis` e pode acessar todos os arquivos necessários para a análise.

8 Configuração do Job SLURM

O arquivo `run.sh` contém as configurações para o SLURM, que incluem alocação de memória, CPUs e GPUs. Abaixo, mostramos o conteúdo do arquivo e explicamos cada uma das linhas:

lst:run

```
#!/bin/sh
#SBATCH --job-name=cell_viability_benchmarking
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=4
#SBATCH --partition=short-gpu-small
#SBATCH --gres=gpu:1g.5gb:3
#SBATCH --mem-per-cpu=4G

usage() {
    echo "Run benchmarking for cell viability pipelines."
    echo "Usage: $0 -m [marvin|local] -p [plugins_directory]"

    echo "Options:"
    echo "  -h    Show this help message."
    echo "  -m [marvin|local]    Mode to run the pipeline.
    ↪  \"marvin\" for HPC Marvin, \"local\" for local machine."
    echo "  -p [plugins_directory]    Path to the plugins
    ↪  directory. Required when mode is local."
    echo " plugins_directory is the path to the \"active_plugins\"
    ↪  directory from the repository:
    ↪  https://github.com/CellProfiler/CellProfiler-plugins.git."
    exit 1
}

# Parse command line arguments
while getopts "m:p:h" opt; do
    case $opt in
    m) mode=$OPTARG ;;
    p) plugins_directory=$OPTARG ;;
    h) usage ;;
    *) usage ;;
    esac
done

# Check if mode is set
if [ -z "$mode" ]; then
    echo "Mode not specified."
```

```
    usage
fi

run_pipeline() {
    local pipeline=$1
    local output_dir=$2
    local extra_args=$3

    echo "> Running $pipeline pipeline"
    singularity run --nv
    ↪  /opt/images/cellprofiler/cellprofiler-4_2_6.sif -c -r -p
    ↪  pipelines/$pipeline -i data -o results/$output_dir
    ↪  --images-per-batch=12 $extra_args
}

case "$mode" in
marvin)
    echo "[==> Processing plate on HPC Marvin"
    run_pipeline "cell_viability_conventional.cppipe"
    ↪  "conventional"
    run_pipeline "cell_viability_cellpose.cppipe" "cellpose"
    ;;
local)
    if [ -z "$plugins_directory" ]; then
        echo "Plugins directory must be specified with -p when
        ↪  mode is local."
        exit 1
    fi
    echo "[==> Processing plate on local machine"
    run_pipeline "cell_viability_conventional.cppipe"
    ↪  "conventional" "--plugins-directory=${plugins_directory}"
    run_pipeline "cell_viability_cellpose.cppipe" "cellpose"
    ↪  "--plugins-directory=${plugins_directory}"
    ;;
*)
    echo "Invalid mode: $mode"
    usage
    ;;
esac

# Data mining and visualization
echo "[==> Data mining and visualization"
if [ -f "benchmarking.py" ]; then
```

```
    python benchmarking.py
else
    echo "benchmarking.py not found. Skipping data mining and
    ↪ visualization."
fi
```

- `#SBATCH --job-name=cell_viability_benchmarking`: Define o nome do job.
- `#SBATCH --ntasks=1`: Especifica o número de tarefas a serem executadas.
- `#SBATCH --cpus-per-task=4`: Aloca 4 CPUs para a tarefa.
- `#SBATCH --partition=short-gpu-small`: Define a partição com GPU.
- `#SBATCH --gres=gpu:1g.5gb:3`: Aloca 3 GPUs de 5GB.
- `#SBATCH --mem-per-cpu=4G`: Define a memória por CPU.
- A função `usage()`: Exibe informações sobre como usar o script e os parâmetros esperados.
- O `getopts`: Permite que o script receba parâmetros de linha de comando, como o modo de execução (`marvin` ou `local`) e o diretório de plugins quando executado localmente.
- O `run_pipeline()`: Executa o pipeline com o comando `singularity` utilizando a imagem do CellProfiler.
- O script verifica o modo (`marvin` ou `local`) e executa o pipeline correspondente.
- Após o processamento, o script executa a mineração de dados e visualização, caso o script `benchmarking.py` esteja presente.

9 Executando o Job SLURM

Após configurar o caminho das imagens, execute o pipeline e a mineração de dados usando o script `run.sh`:

- **Rodando na máquina local**

```
sbash run.sh -m local -p
↪ /path/to/CellProfiler-plugins/active_plugins
```

- **Rodando na máquina HPC Marvin**

```
sbash run.sh -m marvin
```

A saída será salva no diretório `results`, contendo os seguintes arquivos:

- `summary.csv`: [...]

10 Monitoramento e Verificação

Para verificar o andamento do seu job no SLURM, você pode utilizar o comando `squeue`. Este comando mostra uma lista dos jobs ativos na fila de execução, permitindo que você acompanhe o status do seu job em tempo real.

Verificando o status do job:

Execute o seguinte comando para ver o status dos seus jobs ativos no SLURM:

```
squeue -u seu_usuario
```

Onde `seu_usuario` é o seu nome de usuário no sistema. Esse comando exibirá uma lista com informações sobre todos os jobs em execução ou aguardando na fila. As colunas mais importantes incluem:

- **JOBID**: O identificador único do seu job.
- **PARTITION**: A partição onde o job está sendo executado.
- **NAME**: O nome do job, que foi definido no comando `#SBATCH --job-name`.
- **USER**: O usuário que submeteu o job.
- **ST**: O estado atual do job (ex: `R` para executando, `PD` para aguardando).
- **TIME**: O tempo de execução do job.
- **NODES**: O número de nós alocados para o job.

- **NODELIST(REASON)**: Lista dos nós onde o job está alocado ou a razão pela qual o job está aguardando.

Se o job estiver em execução, o estado será exibido como `R` (em execução). Se o job estiver aguardando na fila, o estado será `PD` (pendente).

Visualizando o histórico do job:

Caso queira visualizar mais detalhes sobre um job específico, como o tempo de execução ou o status de recursos, você pode utilizar o comando `scontrol`:

```
scontrol show job JOBID
```

Substitua `JOBID` pelo identificador do seu job. Esse comando fornecerá informações mais detalhadas sobre o andamento e os recursos alocados para o job.

Além disso, sempre verifique os arquivos de saída gerados pelo SLURM para garantir que o pipeline tenha sido executado corretamente. Os arquivos de log geralmente contêm mensagens úteis sobre o andamento do job.

11 Executando o Pipeline `cppipe` no Windows

Para rodar o pipeline do CellProfiler (`cppipe`) no Windows, é necessário configurar o ambiente de execução e garantir que todos os componentes estejam instalados corretamente. Aqui estão as etapas detalhadas para executar o pipeline de maneira eficaz.

11.1 Requisitos

Antes de executar o pipeline, verifique se você tem os seguintes pré-requisitos instalados:

- **CellProfiler**: Você precisa do CellProfiler instalado em sua máquina local. Caso ainda não tenha o CellProfiler instalado, siga as instruções disponíveis na [documentação oficial do CellProfiler](https://cellprofiler.org).
- **Plugins do CellProfiler**: O pipeline `cppipe` pode requerer plugins adicionais, como os do repositório `CellProfiler-plugins`. Faça o download e configure o diretório `active_plugins` conforme necessário.
- **Python**: O CellProfiler pode exigir Python para rodar certos scripts, então é recomendável ter o Python instalado (preferencialmente uma versão compatível com o CellProfiler).

11.2 Rodando o Pipeline no Windows

Após a instalação do CellProfiler e dos plugins, você pode executar o pipeline `cppipe` no seu sistema local.

1. **Configuração do ambiente**: Certifique-se de que todas as dependências estejam corretamente configuradas no seu sistema. O CellProfiler no Windows pode ser executado diretamente a partir do terminal de comando (cmd) ou do PowerShell.

2. **Executando o Pipeline**: A seguir está um exemplo de como rodar o pipeline no Windows:

Abra o prompt de comando (cmd) ou PowerShell e execute o seguinte comando:

```
cellprofiler -p caminho/do/pipeline.cppipe -i
↪ caminho/para/imagens -o caminho/para/resultados
↪ --images-per-batch 12
```

Onde:

- `-p caminho/do/pipeline.cppipe`: Especifica o caminho para o arquivo `cppipe` que define o pipeline.
- `-i caminho/para/imagens`: Define o diretório de entrada, onde estão as imagens a serem processadas.
- `-o caminho/para/resultados`: Define o diretório de saída, onde os resultados serão salvos.
- `--images-per-batch 12`: Define o número de imagens processadas por vez. Ajuste conforme necessário.

Este comando executará o pipeline no CellProfiler, processando as imagens no diretório de entrada e salvando os resultados no diretório de saída especificado.

12 Alterando o `cppipe` para Ajustar a Análise

Dependendo da análise que você está realizando, pode ser necessário ajustar o arquivo `cppipe` para configurar corretamente os módulos e as etapas do pipeline. O arquivo `cppipe` é um arquivo de configuração do CellProfiler que define a sequência de módulos a serem executados durante a análise de imagens, bem como os parâmetros específicos de cada módulo.

12.1 Entendendo o Arquivo `cppipe`

O arquivo `cppipe` contém uma série de módulos que são executados sequencialmente, cada um realizando uma tarefa específica, como segmentação, medição ou visualização das imagens. Cada módulo tem parâmetros que podem ser ajustados para controlar o comportamento da análise, e as alterações no arquivo `cppipe` são necessárias quando você deseja adaptar o pipeline para diferentes tipos de imagens ou análises.

Por exemplo, se você estiver analisando células em imagens de alta resolução, pode ser necessário ajustar os parâmetros do módulo de segmentação para lidar com a resolução maior, ou se estiver analisando imagens de fluorescência, você pode precisar modificar os parâmetros para considerar o contraste ou a intensidade das células.

12.2 Editando o `cppipe` Usando o CellProfiler

O CellProfiler oferece uma interface gráfica e uma interface de linha de comando para criar e editar arquivos `cppipe`. Abaixo, mostramos como você pode editar o arquivo `cppipe` de maneira prática usando o CellProfiler.

12.2.1 Usando a Interface Gráfica do CellProfiler

A maneira mais fácil de editar o arquivo `cppipe` é através da interface gráfica do CellProfiler. Siga as etapas abaixo para abrir e modificar o pipeline:

1. **Abrir o CellProfiler**: Inicie o CellProfiler em sua máquina local ou no ambiente de trabalho.

2. **Carregar o Pipeline Existente**: No menu inicial do CellProfiler, clique em "Open Pipeline"e selecione o arquivo `cppipe` que você deseja editar.

3. **Modificar os Módulos e Parâmetros**: O CellProfiler exibirá todos os módulos do pipeline na interface. Você pode clicar em qualquer módulo para visualizar e modificar seus parâmetros. Por exemplo: - Se o módulo de segmentação não estiver funcionando bem, você pode ajustar o valor de `Thresholding` ou `Rescaling`. - Se você precisar adicionar um novo módulo de análise (como medição de intensidade), clique em "Add Module"e escolha o módulo apropriado.

4. **Salvar o Novo Pipeline**: Após fazer as alterações necessárias, clique em "Save Pipeline As"e salve o arquivo modificado com um novo nome para evitar sobrescrever o arquivo original.

5. **Executar o Pipeline**: Após editar o arquivo `cppipe`, você pode executá-lo diretamente no CellProfiler ou através da linha de comando, como

mostrado na seção anterior.

12.3 Alterações Comuns no `cppipe`

Dependendo do tipo de análise e das características das imagens, você pode precisar realizar várias alterações no arquivo `cppipe`, como:

- **Alterar o Método de Segmentação**: Se as imagens contêm objetos com características diferentes, como células de tamanhos variados ou alta intensidade de fluorescência, você pode precisar alterar o método de segmentação (por exemplo, de Otsu para Adaptive).
- **Adicionar ou Remover Módulos**: Se o seu objetivo é medir características específicas, como a intensidade média das células, você pode adicionar módulos como "MeasureObjectIntensity". Para análises mais simples, você pode remover módulos desnecessários.
- **Ajustar Parâmetros de Tamanho de Objeto**: Se as células variam de tamanho nas imagens, você pode precisar ajustar os limites de tamanho de objeto no módulo de segmentação.
- **Modificar Parâmetros de Medição**: Se você deseja medir aspectos específicos das células (como a área, a intensidade ou a forma), altere os parâmetros dos módulos de medição (por exemplo, "MeasureObjectAreaShape").

12.4 Testando o Pipeline Após Alterações

Após alterar o arquivo `cppipe`, é fundamental testar o pipeline com um conjunto de imagens para verificar se as alterações foram bem-sucedidas. Você pode rodar o pipeline em um pequeno subconjunto de imagens para garantir que todos os módulos estão funcionando corretamente e que as medições estão sendo realizadas como esperado.

Sempre que possível, execute o pipeline em um subconjunto representativo das imagens antes de rodá-lo em todo o conjunto de dados. Isso ajuda a identificar rapidamente qualquer problema nas configurações do pipeline.