UNIVERSIDADE DE SÃO PAULO
Instituto de Física de São Carlos
Trabalho de Conclusão de Curso

Extração de características em escoamentos via Decomposição Ortogonal Própria

João Victor Dell Agli Floriano

Trabalho de conclusão de curso apresentado ao Instituto de Física de São Carlos da Universidade de São Paulo para obtenção do título de Bacharel em Física Computacional. Orientador: Prof. Dr. Afonso Paiva - Instituto de Ciências Matemáticas e de Computação

São Carlos - SP
2023

Resumo

Este trabalho de conclusão de curso traz um estudo sobre o uso de um dos métodos de Decomposição Ortogonal Própria existentes, a SVD, como ferramenta para a análise de escoamentos obtidos por simulações de fluidos. Aplicou-se o algoritmo em um conjunto correspondente a uma simulação de 1499 frames de um campo de densidade de um escoamento, que possibilitou o estudo do comportamento de suas estruturas espaciais com relação à dimensão temporal. Também foi possível, a partir das matrizes retornadas pelo método, realizar uma redução significativa de sua complexidade, chegando a 82%.
Palavras-chave: POD. SVD. Escoamentos. Simulação de Fluidos.

Sumário

1 Introdução

A Decomposição Ortogonal Própria, DOP, ou POD em inglês, é um termo guarda-chuva que descreve um conjunto de métodos de análise de dados baseado na sua fatoração em conjuntos de autovalores e autovetores. Esse conjunto, composto pela Análise de Componentes Principais (PCA), a Decomposição de Karnuhen-Loève (KLD) e a Decomposição em Valores Singulares (SVD) (1), nem sempre foi ou é entendido como tal, apesar da extrema similaridade de seus métodos, visto que cada um deles foi independentemente desenvolvido ao longo dos séculos XIX e XX, sendo o SVD o mais antigo deles, tendo sido estabelecido para matrizes reais em 1870 por Beltrami e Jordan (1).

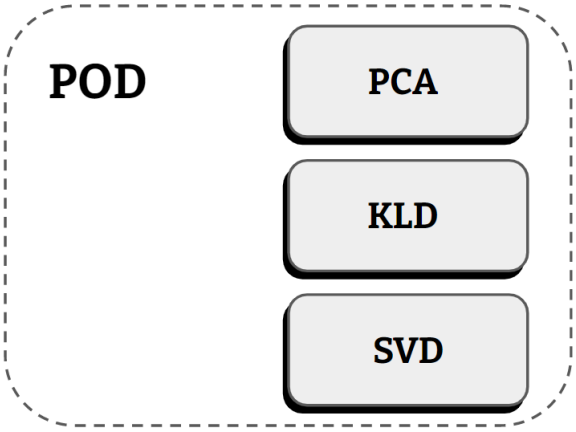

Figura 1: PCA, KLD e SVD como três métodos da POD.
Fonte: Elaborada pelo autor.

A utilidade dos métodos de POD se estende a inúmeras áreas das ciências no geral, sendo o PCA o mais famoso dos três, com uso que vai desde, por exemplo, estudo de amostras fósseis (2), até o estudo da variação de estados de temperatura e pressão de dados meteorológicos (2), sendo aplicado na redução da dimensionalidade de dados de muitas dimensões. Em 1991, Matthew Turk e Alex Pentland introduziram um método de identificação de rostos baseado na decomposição de um conjunto de faces nas que ficaram bem conhecidas como "Eigenfaces" (3), um espaço de autovetores usados como base para comparação de projeções entre rostos do conjunto original e rostos os quais se desejava identificação.

Apesar de ser aplicado em várias áreas, é em dinâmica de fluidos computacional que as PODs encontram grande utilidade, sendo inclusive usualmente associadas a essa área. Aqui, as PODs são aplicadas na decomposição de complexos campos de velocidade e densidade, sejam de simulações, sejam de dados reais, em conjuntos de autofunções determinísticas, podendo inclusive ser usadas para desacoplar a dimensão espacial da temporal de um conjunto de dados (4). Baseando-se nessa última aplicação, este projeto teve como objetivo estudar a aplicação de um dos métodos de POD, a SVD, em alguns conjuntos de dados de simulações de fluidos.

2 Materiais e Métodos

2.1 A decomposição SVD

A Decomposição em Valores Singulares, ou SVD, é, como descrito anteriormente, um método antigo de fatoração de matrizes. Dada uma matriz M real, de tamanho $n \times m$, sua decomposição SVD terá o seguinte formato:

$$M = U\Sigma V^T \tag{1}$$

A qual:

- U: Matriz $n \times n$ a qual suas colunas são os autovetores de MM^T, também chamados de *vetores singulares à esquerda*
- V: Matriz $m \times m$ a qual suas colunas os autovetores de M^TM, também chamados de *vetores singulares à direita*
- Σ: Matriz $n \times m$ a qual suas entradas diagonais são as raízes quadradas dos autovalores não nulos de M^TM ou MM^T, também chamadas de *valores singulares*.

Essas três submatrizes, por sua vez, possuem inúmeras interpretações, geralmente dependendo do contexto a qual a decomposição foi aplicada. Uma interpretação comum, no contexto o qual M é uma transformação linear, é que essas três submatrizes são uma fatoração da transformação original em três operações elementares: uma rotação no subespaço de V, um dimensionamento por Σ, e uma rotação no subespaço de U.

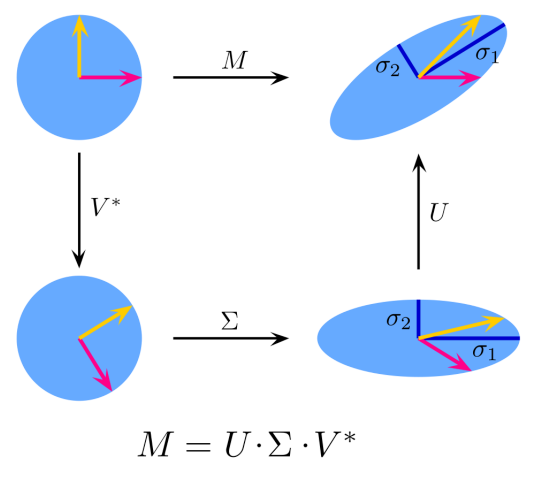

Figura 2: Sub-transformações de M.
Fonte: https://commons.wikimedia.org/wiki/File:Singular-Value-Decomposition.svg

No contexto deste trabalho, a decomposição SVD será aplicada à amostras de campos, sendo eles escalares ou vetoriais, da forma $\mathbf{F}(\mathbf{x}, t)$ representando o escoamento de um fluido em um intervalo de tempo e domínio espacial específico, chamado de *snapshot*. Considerando um conjunto de dados de um escoamento no domínio $[0, x] \times [0, y] \times [0, t]$, discretizados em um vídeo de dimensões (n, m, f), sendo n e m a quantidades de elementos em x e y, e f o número

de frames em t, para aplicarmos a decomposição nesse conjunto, devemos primeiro condensar as dimensões espaciais discretizadas de x e y em um vetor de apenas uma dimensão. Tendo cada frame t_i condensado em um vetor de tamanho $n \times m$, podemos empilhar esses vetores coluna seguidamente em uma matriz M, que terá enfim um tamanho (h, f), sendo $h = n \times m$.

$$M = \begin{bmatrix} \vdots & \vdots & & \vdots \\ t_0 & t_1 & \dots & t_f \\ \vdots & \vdots & & \vdots \end{bmatrix} \tag{2}$$

Com o nosso conjunto de dados no formato de uma matriz comum, a decomposição pode enfim ser feita. As três submatrizes terão o seguinte formato:

$$M = U(\mathbf{x}) \cdot \Sigma \cdot V^T(t) \tag{3}$$

Em particular, a SVD desempenhará um papel além de apenas fatorar essa matriz de dados: ela desacoplará as dimensões temporal e espacial (4), as quais serão representadas pelas matrizes U e V^T respectivamente. Nesse caso, a matriz U será composta pelos f autovetores coluna espaciais, e a matriz V^T será composta pelos f autovetores coluna transpostos temporais:

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ t_0 & t_1 & \dots & t_f \\ \vdots & \vdots & & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ u_1 & u_2 & \dots & u_h \\ \vdots & \vdots & & \vdots \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & 0 & ... & 0 \\ 0 & \sigma_2 & ... & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & ... & \sigma_f \\ \vdots & \vdots & & \vdots \\ 0 & 0 & ... & 0 \end{bmatrix} \cdot \begin{bmatrix} \dots & v_1^T & \dots \\ \dots & v_2^T & \dots \\ & \vdots & \\ \dots & v_f^T & \dots \end{bmatrix} \tag{4}$$

Se traduzindo em uma combinação linear dos produtos dos vários elementos de U e V^T, com os valores singulares de Σ como coeficientes:

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ t_0 & t_1 & \dots & t_f \\ \vdots & \vdots & & \vdots \end{bmatrix} = \sigma_1 \begin{bmatrix} & \dots & v_1^T & \dots \\ \vdots & & & \\ u_1 & & & \\ \vdots & & & \end{bmatrix} + \dots + \sigma_f \begin{bmatrix} & \dots & v_f^T & \dots \\ \vdots & & & \\ u_f & & & \\ \vdots & & & \end{bmatrix} \tag{5}$$

Em relação a essas matrizes, é comum que surja uma dúvida importante: Se a matriz Σ é composta pela raiz quadrada dos autovalores de $M^T M$, os quais são os mesmos autovalores de MM^T, e considerando que essas duas matrizes podem possuir postos diferentes se M for uma matriz retangular, como esses autovalores são os mesmos se uma tem mais autovalores que a outra? A resposta para isso é simples: Apenas os autovalores até o menor posto entre as duas matrizes são relevantes, o resto dos autovalores são nulos.

Isso pode inclusive ser verificado, resguardados os limites numéricos do contexto o qual o método está sendo aplicado, pela progressão do valor dos autovalores, do mais relevante ao menos relevante, que usualmente segue uma progressão semelhante a $\frac{1}{x}$, como no exemplo da figura 3, retirado da decomposição de um conjunto de dados específico.

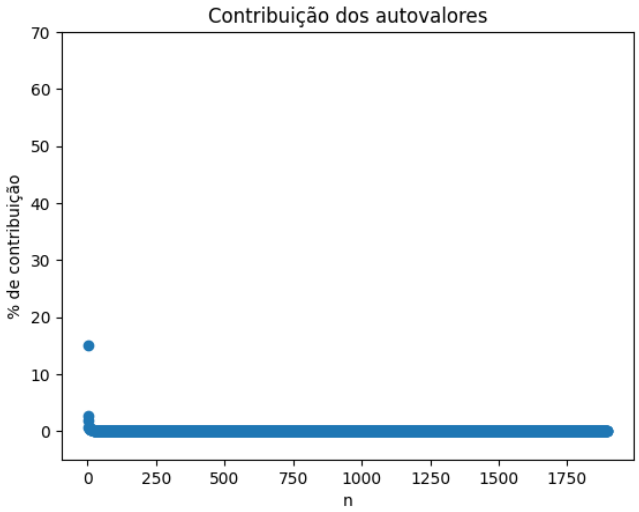

Figura 3: Progressão do autovalores.
Fonte: Elaborada pelo autor.

Como pode ser verificado acima, a quantidade de autovalores próximos de zero é muito grande. Essa característica é crucial para a utilidade desses métodos de POD, pois a partir dela pode-se construir um subespaço de complexidade muito menor se comparado ao subespaço original.

No exemplo da figura 3, se realizarmos uma soma dos autovalores até que essa soma totalize o valor de 99%, chegamos ao seguinte resultado presente na figura 4.

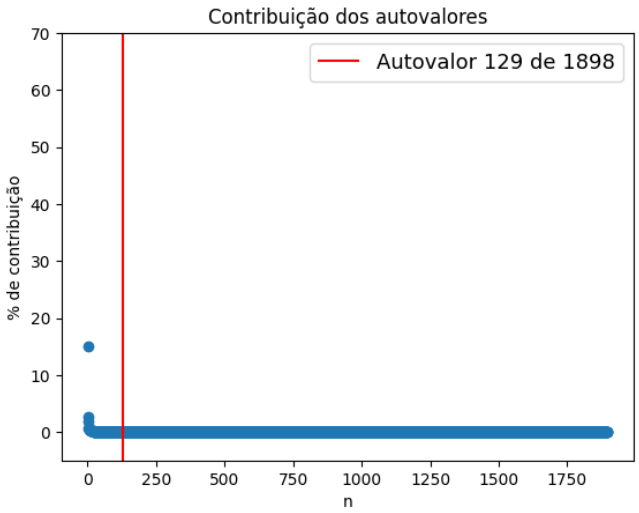

Figura 4: Autovalor limite a qual a soma dos anteriores corresponde a 99% dos dados.
Fonte: Elaborada pelo autor.

Apenas 129 autovalores correspondem a 99% dos dados. Essa quantidade é aproximadamente $6,79\%$ do total. Esse dado constitui uma das principais aplicações das POD, que é a redução da complexidade do problema, pois sabendo-se que essa quantidade pequena de autovalores corresponde a esse quase total valor dos dados, seleciona-se apenas essa quantidade para análise e reconstrução dos mesmos.

$$\begin{bmatrix} \vdots & \vdots & & \vdots \\ t_0 & t_1 & \dots & t_f \\ \vdots & \vdots & & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ u_1 & u_2 & \dots & u_n \\ \vdots & \vdots & & \vdots \end{bmatrix} \cdot \begin{bmatrix} \sigma_1 & 0 & ... & 0 & \dots & 0 \\ 0 & \sigma_2 & ... & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \\ 0 & 0 & ... & \sigma_n & \dots & 0 \end{bmatrix} \cdot \begin{bmatrix} \dots & v_1^T & \dots \\ \dots & v_2^T & \dots \\ & \vdots & \\ \dots & v_f^T & \dots \end{bmatrix} \tag{6}$$

Sendo n, nesse caso, a quantidade de autovalores desejada para reconstrução do espaço.

2.2 PCA

2.2.1 A teoria

O método PCA é outra abordagem interessante da decomposição ortogonal, sendo equivalente ao SVD, porém vindo de um contexto diferente. Apesar de ter sido inventada independentemente por Hardold Hotelling em 1933 (5), o qual cunhou seu nome, o método foi originalmente derivado por Karl Pearson (6) em 1901, como um análogo do Teorema dos Eixos Principais, que descreve um conjunto de linhas ortogonais que cruzam um elipsoide ou hiperboloide como sendo seus "eixos principais".

Assim como o teorema descrito acima, a ideia do método PCA é encontrar os "componentes principais" de um conjunto de dados abstrato, sendo estes as direções, ortogonais entre si, as quais os dados tem a maior projeção. Essa ideia pode ser traduzida geometricamente no ato de procurar as retas principais que passam "pelo meio" da nuvem de dados.

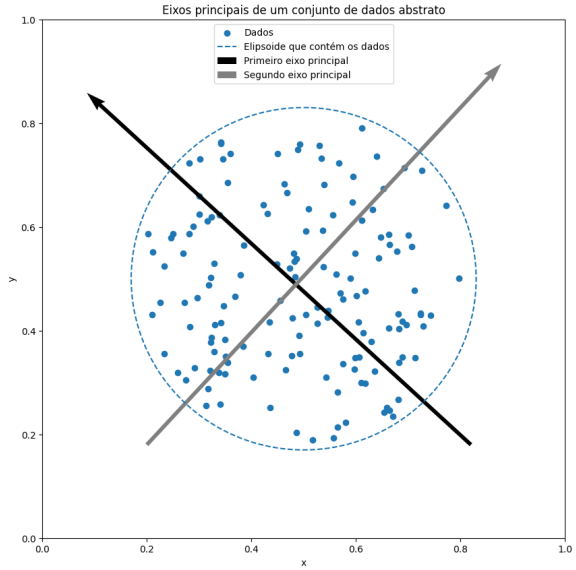

Figura 5: Eixos principais de um conjunto de dados.
Fonte: Elaborada pelo autor.

Assim como na SVD, sabendo as direções com a maior projeção a calculados pelo método, é possível calcular também qual a porcentagem de sua contribuição nos dados originais, sendo ela a contribuição do componente de interesse dividido pela soma da contribuição de todos os componentes. Sabendo essas porcentagens, conseguimos ordenar as maiores contribuições, nos permitindo definir quais são os componentes mais e menos significativos, que por sua vez

também permitem uma filtragem semelhante ao apresentado anteriormente, que reconstrói os dados em menos dimensões.

Um passo a mais existente nas aplicações de PCA, além da filtragem, é da rotação do conjunto de dados na direção das componentes principais, para que os eixos de referência coincidam com os mesmos, e a análise dos dados seja facilitada.

2.2.2 A dedução do método PCA 1D

Usemos como exemplo o caso da projeção em uma dimensão. Supondo que queiramos projetar um conjunto de dados de p dimensões em uma reta só que passe a partir da origem "pelo meio" exato desses dados, representada pelo vetor unitário $\vec{w}$. A reta que melhor se encaixará nesse casos será a reta a qual as distâncias entre ela e cada ponto será a mínima (que também significa que será a reta a qual a projeção dos dados na mesma será máxima). Ou seja, temos que dar um jeito de minimizar a média das distâncias ponto-reta de todos os dados (também conhecida como Erro Quadrático Médio):

$$min(\ MSE(\vec{w})\ ) = min(\ \frac{1}{n}\sum_{i=1}^{n} ||\vec{x}_i - (\vec{x}_i \cdot \vec{w})\vec{w}||^2\ ) \tag{7}$$

A partir dessa minimização descobriremos qual deve ser a direção da reta que cumpre esses requisitos. Abrindo o termo interno do somatório acima, temos:

$$||\vec{x}_i - (\vec{x}_i \cdot \vec{w})\vec{w}||^2 = \vec{x}_i \cdot \vec{x}_i - (\vec{x}_i \cdot \vec{w})^2 \tag{8}$$

Que, no MSE, é:

$$MSE(\vec{w}) = \frac{1}{n}(\ \sum_{i=1}^{n} \vec{x}_i \cdot \vec{x}_i - (\vec{x}_i \cdot \vec{w})^2\ ) = \frac{1}{n}(\ \sum_{i=1}^{n} ||\vec{x}_i||^2 - \sum_{i=1}^{n} (\vec{x}_i \cdot \vec{w})^2\ ) \tag{9}$$

Para que o MSE seja minimizado, a somatória das projeções na reta deve ser maximizada. Reescrevendo a mesma como:

$$\frac{1}{n}\sum_{i=1}^{n} (\vec{x}_i \cdot \vec{w})^2 = (\frac{1}{n}\sum_{i=1}^{n} \vec{x}_i \cdot \vec{w})^2 + Var[\ \vec{x}_i \cdot \vec{w}\ ] \tag{10}$$

Como a média da soma dos dados é zero, então a média das projeções também será zero, portanto, só sobra a variância a ser maximizada. *Sendo assim, encontrar a reta que passa "pelo meio" dos dados significa encontrar a reta a qual há a maximização da variância da projeção dos dados nessa reta.*

Basta agora maximizarmos a variância das projeções para encontrar a reta. Para facilitar na visualização, façamos os cálculos na forma matricial, empilhando os dados $\vec{x}_i$ em uma matriz $\mathbf{x}$ de dimensões $n \times p$. A variância, como visto acima, pode ser escrita então como:

$$Var[\ \vec{x}_i \cdot \vec{w}\ ] = \sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (\vec{x}_i \cdot \vec{w})^2 \tag{11}$$

$$= \frac{1}{n}(\mathbf{xw})^T(\mathbf{xw}) = \frac{1}{n}\mathbf{w}^T\mathbf{x}^T\mathbf{xw} \tag{12}$$

$$= \mathbf{w}^T\frac{\mathbf{x}^T\mathbf{x}}{n}\mathbf{w} = \mathbf{w}^T\mathbf{vw} \tag{13}$$

Para que a maximização, precisamos garantir que ela resultará em um vetor $\vec{w}$ tal que ele será unitário, pois essa é uma das condições definidoras do nosso problema. Para que essa garantia seja feita, introduziremos o uso de um *multiplicador de Lagrange* λ associado à uma restrição $\mathbf{w}^T\mathbf{w} = 1$ que junto com o problema original constituirá um novo objeto, que aí sim será maximizado:

$$L(\mathbf{w}, \lambda) = \mathbf{w}^T\mathbf{v}\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1) \tag{14}$$

Derivando parcialmente:

$$\frac{\partial L}{\partial \lambda} = \mathbf{w}^T\mathbf{w} - 1 \tag{15}$$

$$\frac{\partial L}{\partial \mathbf{w}} = 2\mathbf{v}\mathbf{w} - 2\lambda\mathbf{w} \tag{16}$$

Igualando as duas derivadas a zero para encontrar os pontos extremos, temos:

$$\mathbf{w}^T\mathbf{w} = 1 \tag{17}$$

$$\mathbf{v}\mathbf{w} = \lambda\mathbf{w} \tag{18}$$

Que nos mostra uma coisa muito importante. A reta que define a variância máxima das projeções é a reta definida por um autovetor da matriz de covariância $\mathbf{v}$, no caso, aquele associado ao maior autovalor. Portanto, basta encontrar os autovalores e autovetores de $\mathbf{v}$ que calcularemos o componente principal desejado dos dados.

A componente principal calculada resolverá o caso para projeção em uma dimensão, porém esses resultados não se restringem a apenas isso. Sendo a matriz $\mathbf{v}$ uma matriz de covariância, ela é *simétrica e positiva*, nos dizendo que todos seus autovetores serão *ortogonais* com *autovalores positivos*. Cada autovetor será uma componente principal dos dados originais.

2.3 Aspectos Computacionais

Como visto anteriormente, alguns passos devem ser seguidos para se obter uma implementação de SVD funcional:

1. Calcular M^TM.
2. Calcular MM^T.
3. Calcular os autovalores e autovetores de 1.
4. Calcular os autovalores e autovetores de 2.
5. Ordenar os autovalores e autovetores calculados por maior relevância e dispô-los em U, Σ e V^T

Poderia ter-se implementado cada passo numericamente com seus devidos métodos já bem conhecidos, porém isso envolveria tempo gasto de maneira desnecessária, podendo inclusive resultar em uma aplicação lenta, visto que as devidas otimizações estado-da-arte muito provavelmente não seria usadas. Como o objetivo do projeto era de se estudar a aplicação de

PODs, optou-se por usar uma implementação já pronta, presente na biblioteca *numpy*, dentro do módulo de álgebra linear *linalg*.

```
# Como importar svd
from numpy.linalg import svd
```

Seu uso se da da seguinte maneira:

```
# Como importar svd
U, Sigma, Vt = svd(M)
```

Coloca-se uma matriz M, a qual se deseja fazer a decomposição SVD, e recebe-se as três matrizes oriundas da decomposição. Um passo a mais no seu uso que acelera os cálculos das três matrizes, visto que muitas vezes estaremos lidando com matrizes de tamanho considerável, é de atualizar a variável *full_matrices*, que por padrão é *True*, para *False*. Essa atualização fará com que a função evite calcular todos os autovalores e autovetores da decomposição, ou seja, irá calcular apenas aqueles até o número de autovalores não-nulos, definidos pelo menor posto da matriz original.

```
# Como importar svd
U, Sigma, Vt = svd(M, full_matrices = False)
```

Além da biblioteca *numpy*, também foram usadas nesse projeto as bibliotecas *pandas* e *matplotlib.pyplot*, para importação e plotação dos resultados, respectivamente.

```
# Importando pandas e matplotlib.pyplot
import pandas as pd
import matplotlib.pyplot as plt
```

Usando-se dessas bibliotecas, pode-se enfim desenvolver um procedimento para tratar os dados desejados. O procedimento segue a simples ordem:

1. Recebe-se o conjunto de imagens o qual deseja-se analisar em um array.
2. Achata-se as imagens em vetores coluna unidimensionais, empilhando-os em uma matriz.
3. Aplica-se a decomposição SVD.

Um exemplo de programa que realiza esses passos é:

```
# Importacao dos dados com pandas
rawdata = pd.read_csv("/caminho/para/os/dados.csv")

# Conversao dos dados crus em um np.array
data = np.array(rawdata)

# Decomposicao SVD pelo numpy
U, S, Vt = svd(data_d, full_matrices = False)
```

A partir da decomposição, pode-se escolher que direção tomar com as informações recebidas nas três submatrizes. Pode-se tentar, por exemplo, reconstruir uma imagem de fora do conjunto original a partir dos autovetores obtidos.

Sendo I uma imagem desconhecida e $E = V^T$ a matriz do auto-espaço calculado do conjunto de imagens, a reconstrução dessa imagem é feita de seguinte maneira:

1. Projetamos a imagem desconhecida no auto-espaço gerado de forma a obter um vetor com cada entrada sendo os coeficientes de projeção:

$$\Omega = IE^T \tag{19}$$

2. Multiplicamos esses coeficientes por cada autovetor do auto-espaço, gerando assim a combinação linear dos mesmos:

$$I' = \Omega E \tag{20}$$

No caso acima, quanto mais autovetores escolhermos, melhor será a reconstrução, apesar de não necessariamente perfeita. Pode-se também tentar identificar uma imagem presente no conjunto por meio dos autovetores. Para realizar essa comparação, devemos primeiro calcular o vetor com os coeficientes da projeção da imagem desconhecida no auto-espaço calculado. O vetor, da forma:

$$\Omega = \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_k \end{bmatrix} \tag{21}$$

Deverá ser subtraído dos vetores das projeções de cada imagem original do conjunto, da maneira:

$$D_i = ||\Omega - \Omega_i|| \tag{22}$$

A menor distância corresponderá à imagem mais parecida, porém não necessariamente corresponderá à uma imagem necessariamente igual. Podemos ainda definir uma tolerância, para qual imagens com distância abaixo de certa tolerância T corresponderão à mesma pessoa da imagem desconhecida, porém para os fins desse projeto, isso não será feito.

Caso desejemos fazer uma filtragem por autovalores relevantes, e por exemplo, reconstruir o conjunto com menos elementos, uma aplicação que pode economizar espaço de memória, deveremos seguir os passos seguintes:

1. Calcula-se os autovalores elevando os elementos de Σ ao quadrado.
2. Seleciona-se a quantidade de autovalores significativos desejados.
3. Reconstrói-se os dados a partir desse número inferior de autovalores.

```
# Filtragem dos autovalores
s = 0
m = 0
percent = 0.99
while(s <= percent):
  s = np.sum(contribs[:m])
  m += 1
```

Para a reconstrução, usou-se duas funções auxiliares. A primeira, *normalize*, renormaliza a imagem dentro de um intervalo especificado. A segunda *to_image*, converte uma imagem para o intervalo de um inteiro sem sinal, que é o intervalo de $[0, 255]$, intervalo usado para a atual codificação de imagens:

```
#Funcoes uteis
def normalize(img : np.ndarray, min, max):
  '''Function that converts an image to the given desired range.
     '''
  return ((img - np.min(img))/(np.max(img) - np.min(img)))*(max -
      min) + min

def to_image(img):
  return normalize(img, 0, 255).astype(np.uint8)
```

Que enfim são aliadas na reconstrução do conjunto:

```
# Reconstrucao do conjunto
def rebuild_frame(U, Sigma, Vt, frame, slice):
  proj2 = np.matrix(U[frame, :]) * np.matrix(np.diag(Sigma)[:, :
     slice])

  return to_image(proj2 * np.matrix(Vt[:slice, :]))
```

2.4 O Clássico Problema das Eigenfaces

Como teste inicial desse algoritmo, resolveu-se um clássico problema, proposto inicialmente por Sirovich e Kirby em 1987 (7), e posteriormente aprimorado por Turk e Pentland em 1991. A partir do desejo de se encontrar bases de dimensão baixas de imagens de rostos, Sirovich e Kirby haviam proposto o uso de PCA para a formação de uma base a partir de conjunto pré-coletado dos mesmos, base a partir da qual se reconstruiria esse mesmo conjunto por meio de combinações lineares de seus elementos, como mostrando anteriormente. Turk e Pentland teriam enfim, em 1991, apresentado uma proposta consistente de detecção de rostos por meio dessa técnica, que ficou então conhecido como o "Método de Eigenfaces para reconhecimento de rostos". O método consiste, de maneira resumida, em:

1. Escolher um conjunto de rostos que será base para essa identificação.
2. Realizar a decomposição ortogonal desse conjunto, obtendo-se a matriz com os autovetores dos rostos, a base dos "auto-rostos".
3. Classificar cada rosto do conjunto original projetando-o na base de autovetores. O vetor resultante da projeção nessa matriz será a "identificação" desse rosto.
4. Obtendo-se essa base com as identificações, basta agora calcular o vetor da projeção do rosto que deseja se identificar. Com essa identificação, realiza-se uma comparação, que pode ser um simples cálculo da distância entre esses dois vetores, entre a identificação calculada e as identificações de cada rosto da base. A melhor comparação corresponderá ao rosto a qual desejava-se identificar.

Para testar esse método, usou-se como base de dados uma amostra de imagens de rostos disponibilizada, na época, no site da Universidade de Boston. Essa base de dados continha fotos de rostos de 38 pessoas diferentes, cada uma com, em média, 64 fotos de ângulos de iluminação. Para testar a possibilidade de reconstrução de rostos de fora do conjunto, escolheu-se apenas 30 dessas 38 pessoas para os cálculos.

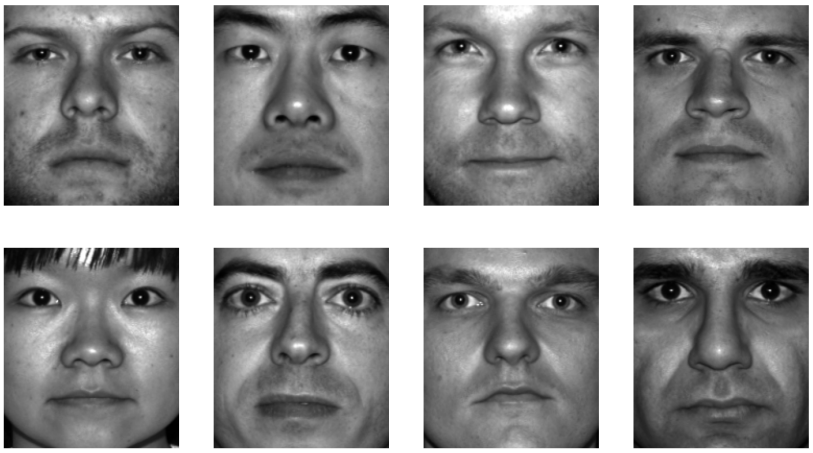

Figura 6: Alguns dos rostos disponíveis no conjunto de dados.
Fonte: Universidade de Boston.

Figura 7: Cada rosto possui sub-fotos de ângulos de iluminação diferentes.
Fonte: Universidade de Boston.

Aplicando a SVD nesse conjunto, obtemos a matriz com os "auto-rostos", cada um disposto como uma coluna da matriz.

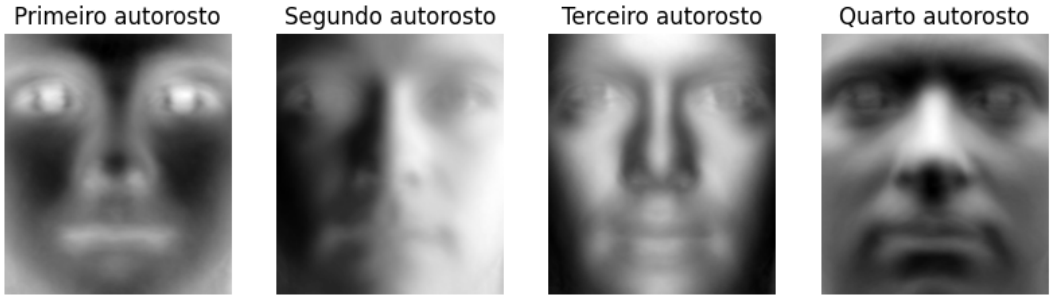

Figura 8: Primeiros quatro auto-rostos.
Fonte: Elaborado pelo autor.

Podemos então testar se conseguimos reconstruir uma imagem fora do conjunto original apenas com esses auto-rostos.

Figura 9: Rosto de fora do conjunto.
Fonte: Universidade de Boston.

Dado o rosto 9 de fora do conjunto, calculando sua reconstrução pelo método demonstrado anteriormente, obtemos o resultado representado pela figura 10.

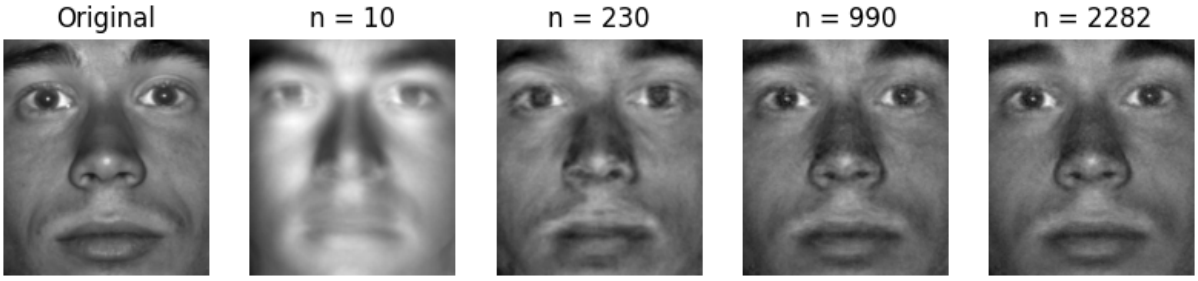

Figura 10: Reconstrução com n autovetores.
Fonte: Universidade de Boston.

Testemos agora o método de identificação de rostos, projetando um rosto de fora do conjunto. Para algumas imagens escolhidas, obteve-se os seguintes resultados, representados pelas imagens 11 e 12.

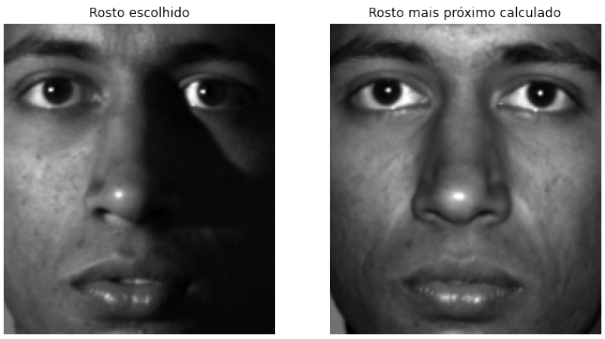

Figura 11: Exemplo de identificação que deu certo.
Fonte: Elaborado pelo autor.

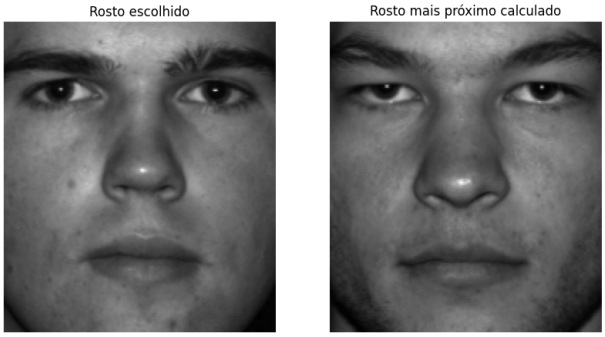

Figura 12: Exemplo de identificação que deu errado.
Fonte: Elaborado pelo autor.

É possível verificar pelo segundo exemplo que nem todo rosto será corretamente identificado. Se rodarmos esse algoritmo para todo o conjunto de rostos, poderemos verificar quais foram identificados corretamente.

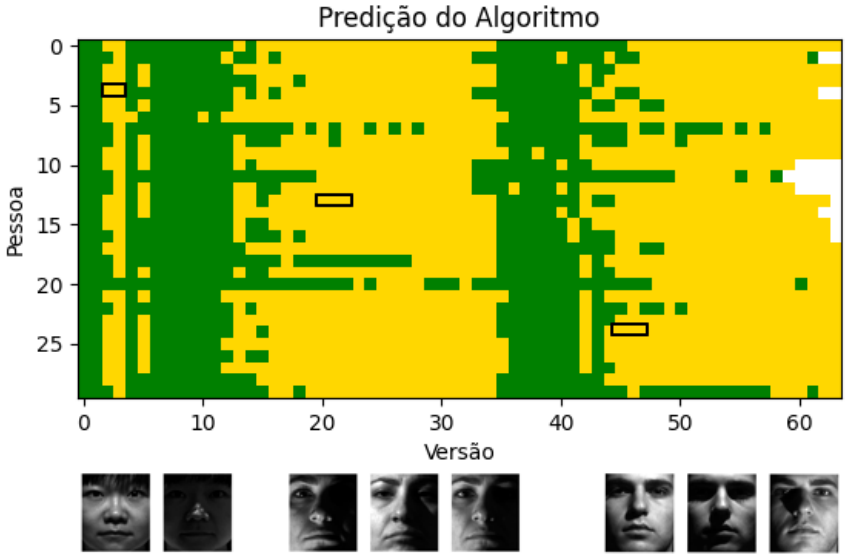

Figura 13: Acertos e erros do algoritmo.
Fonte: Elaborado pelo autor.

Na figura 13, verde corresponde aos rostos corretamente identificados, e amarelo os identificados erroneamente. Pode-se verificar assim as áreas do conjunto as quais o algoritmo falha, que por sua vez correspondem a ângulos de iluminação os quais produzem quantidade significativa de sombra sobre o mesmo.

3 Resultados

3.1 A Aplicação em Escoamentos

Para demonstrar a aplicação das PODs em escoamentos, usou-se um conjunto de dados gerado independentemente. Esse conjunto constitui em uma simulação do escoamento de um fluido, representado por frames de um campo escalar correspondente à sua densidade, a qual é armazenada em cada pixel da imagem. Essa simulação possui 1500 frames de dimensão $(100, 100)$. É importante pontuar, antes de continuar com esse estudo, que todas as imagens da simulação aqui mostradas terão seu valor normalizado no intervalo $[0.0, 1.0]$, mas não necessariamente correspondem a valores dentro desse intervalo.

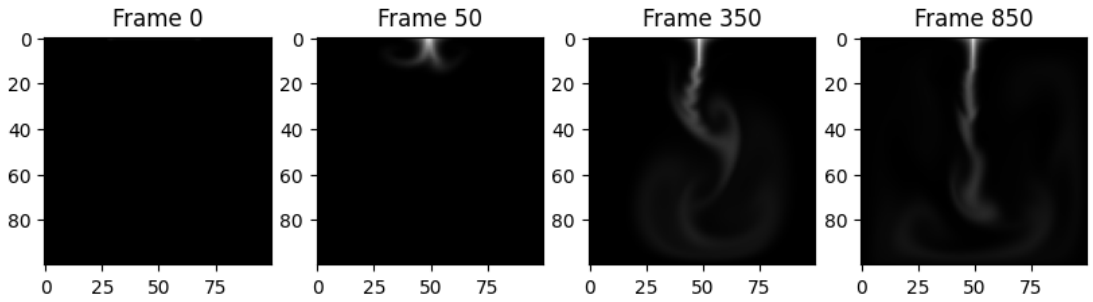

Figura 14: Simulação do escoamento de um Fluido.
Fonte: Elaborado pelo autor.

Primeiro, achatou-se os frames em vetores de 10000 entradas, e os empilhou em colunas, rendendo uma matriz $(10000, 1499)$. Aplicando a decomposição SVD nesse conjunto, obteve-se as três submatrizes, com as seguintes dimensões:

- U : $(10000, 1499)$
- Σ : $(1499, 1499)$
- V^T : $(1499, 1499)$

Como descrito anteriormente, U corresponde aos autovetores espaciais, Σ aos autovalores, e V^T aos autovetores temporais. Seus autovalores normalizados tem as seguintes contribuições, representadas na figura 15.

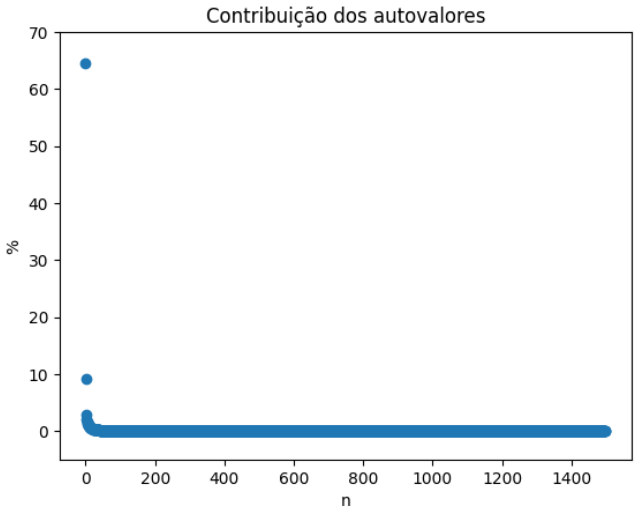

Figura 15: Porcentagem da contribuição dos autovalores.
Fonte: Elaborado pelo autor.

Cada um desses autovalores corresponde aos seguintes autovetores, em ordem de significância.

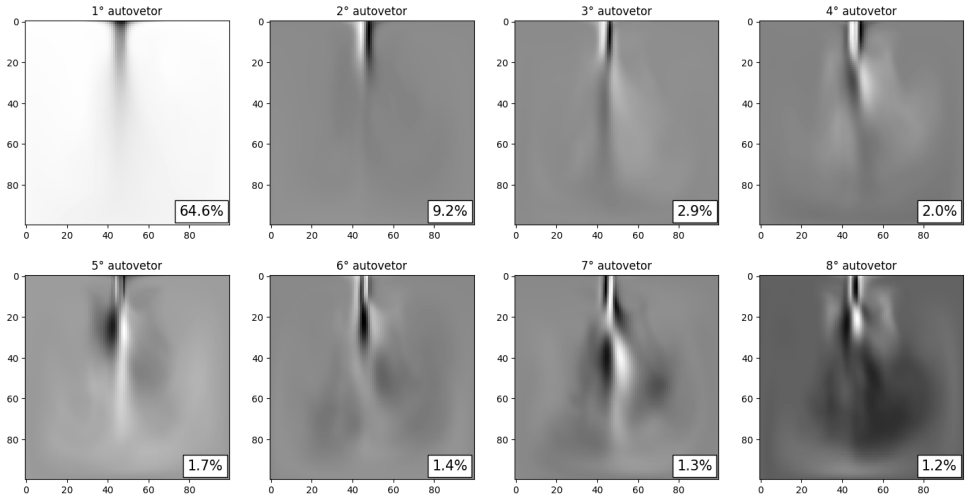

Figura 16: Primeiros e mais relevantes autovetores e suas contribuições.
Fonte: Elaborado pelo autor.

Analisando o conjunto, representado por alguns modos iniciais em 16, nota-se uma característica peculiar: O modo de maior densidade, correspondente ao primeiro autovetor, possui valores pequenos no meio, aonde a estrutura da fumaça mais se concentrava, e valores grandes fora dessa região. Fora de contexto, esse resultado não deveria fazer sentido, pois se cada autovetor acima corresponde aos modos de maior densidade, e a densidade deve possuir valores positivos, por que o modo de maior densidade possui essa inversão? Essa dúvida bem motivada é respondida analisando o primeiro modo conjuntamente com seu respectivo autovetor temporal.

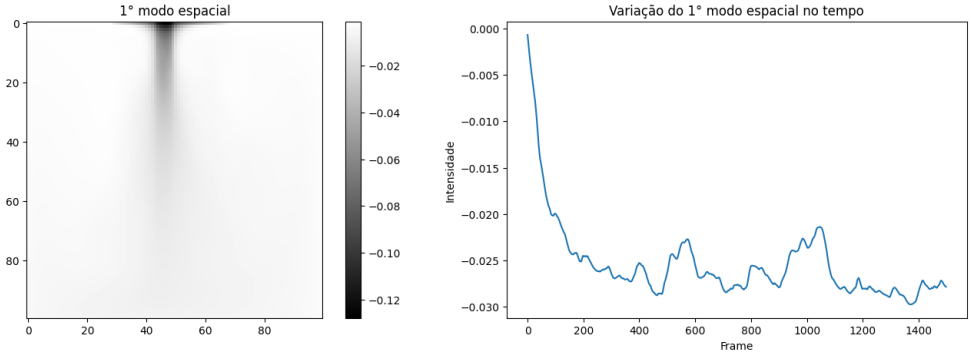

Figura 17: Comparação entre o primeiro modo espacial e sua "variação" no tempo.
Fonte: Elaborado pelo autor.

Essa comparação levanta um ponto importante desta análise a qual deve ser sempre praticado: o estudo dos modos espaciais deve vir sempre acompanhado dos modos temporais. No exemplo da figura 17, verificamos que a variação temporal do primeiro e mais "energético" modo tem valores negativos ao longo de toda a simulação, que ao serem multiplicados pelo modo espacial, invertem seu valor.

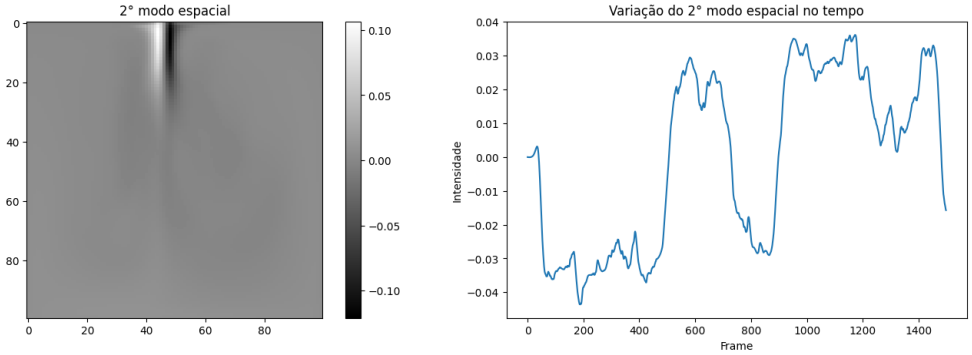

Figura 18: Comparação entre o segundo modo espacial e sua variação no tempo.
Fonte: Elaborado pelo autor.

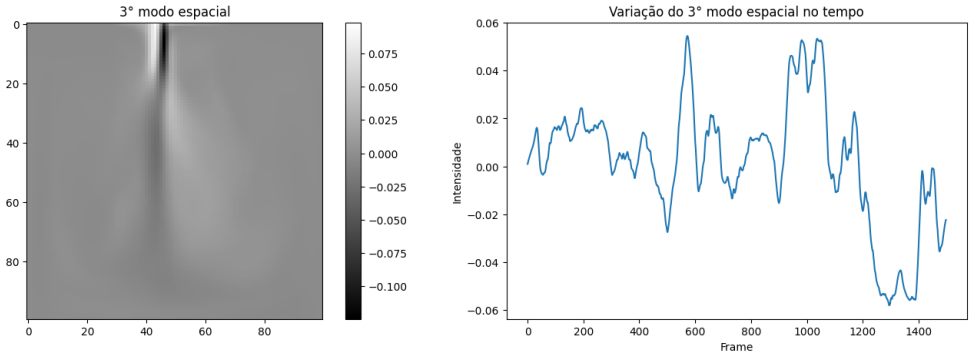

Figura 19: Comparação entre o terceiro modo espacial e sua variação no tempo.
Fonte: Elaborado pelo autor.

Se realizarmos o procedimento de filtragem procurando pela quantidade de autovalores correspondente a 99%, obtemos o resultado representado pela figura 20

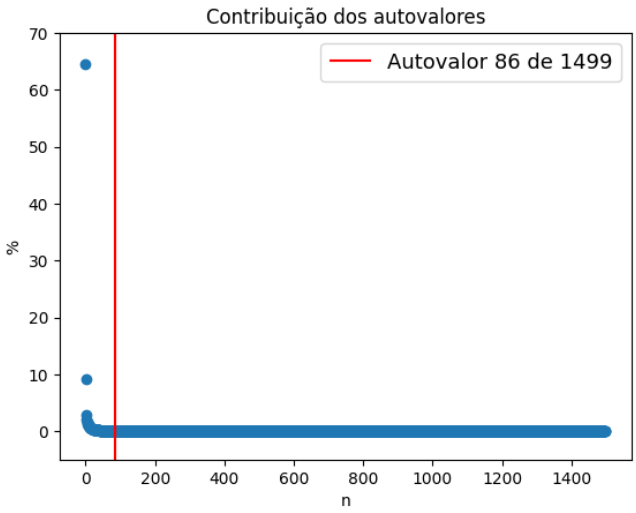

Figura 20: Autovalor limite correspondente a 99% do conjunto.
Fonte: Elaborado pelo autor.

Pelo que foi calculado, 86 autovalores correspondem a 99% do conjunto. Se reconstruirmos um frame qualquer com apenas essa quantidade de autovalores, obtemos os seguintes frames na figura 21.

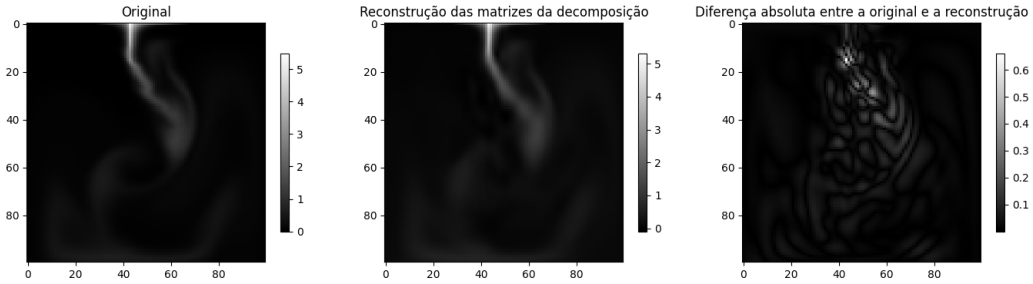

Figura 21: Reconstrução de um frame qualquer com 86 autovalores.
Fonte: Elaborado pelo autor.

É possível verificar por 21 que a reconstrução conseguiu obter, de fato, uma representação muito parecida com o frame original. Isso pode ser respaldado pela ordem da diferença absoluta entre o original e a reconstrução, que é pequena se comparada aos valores de densidade do frame.

Usando o exemplo acima para calcular a eficiência de compressão desse algoritmo, considerando que se tratava de uma matriz de tamanho $(10000, 1499)$ a qual selecionou-se apenas 86 autovalores, o necessário para reconstruir o conjunto será:

- U $(10000, 86)$
- Σ (86)
- V^T $(1499, 1499)$

Somando essa quantidade, obtemos 3.107.087 números a serem guardados, que se comparados à quantidade original, 17.238.500, representa uma redução de $81,97\%$ no espaço armazenado.

4 Conclusão

Os métodos de POD configuram uma importante e poderosa ferramenta de análise de dados. Sua característica de aplicação *a posteriori*, ou seja, diretamente a um conjunto de dados já obtido, configura uma interessante versatilidade, pois para o método, pouca ou nenhuma diferença fazem as restrições e regras analíticas inerentes ao comportamento analisado, as quais podem inclusive serem reveladas a partir da análise dos resultados. As áreas as quais esse método pode ser aplicado são inúmeras, limitadas apenas à imaginação de quem está o empregando, que deverá demonstrar empenho em lidar com significados variados que podem emergir de seus resultados.

Essa versatilidade já poderia ter sido identificada no exemplo das Eigenfaces, pois o método demonstra não só como pode encontrar uma base em baixas dimensões para um conjunto de rostos, mas como pode lidar com novos dados, conseguindo reconstruir um rosto de fora do conjunto apenas com as imagens limitadas fornecidas. Em física, mais especificamente em CFD, essa ferramenta pode servir para uma análise eficiente dos comportamentos espaciais e sua relação temporal, visto que uma Decomposição Ortogonal Própria pode desacoplar essas dimensões com comportamentos antes complicados de se analisar por vias tradicionais. Além das aplicações físicas, do ponto de vista computacional, esse método demonstra interessante capacidade em reduzir a complexidade e dimensionalidade dos dados a qual ele foi empregado, correspondendo à intenção original que deu origem ao mesmo, pois como visto anteriormente, ele pode representar uma redução de até 82% no tamanho de armazenamento de uma simulação, um resultado significativo levando em consideração a quantidade de espaço e processamento empregado em uma rotina de análise do mesmo.

É importante destacar que, apesar de suas características importantes para qualquer área que lide com análise de dados, o método possui sim, limitações. Como verificado no exemplo das Eigenfaces, mudanças não muito expressivas nos ângulos de limitação [13] podem render identificações errôneas de rostos. No exemplo do simples escoamento, apesar de pequenas as diferenças entre o conjunto original e sua reconstrução [21], essas podem representar alguma característica importante do comportamento do escoamento, e devem ser descartadas com cuidado. Para lidar com essas limitações, caberá ao indivíduo a qual o está o aplicando cuidado e garantia de que o mesmo se apropriou o bastante do contexto da aplicação para evitar cair no encantamento cego pelas vantagens as quais esse método apresenta.

Referências

1. Y.C. LIANG, H.P. LEE, S.P. LIM, W.Z. LIN, K.H. LEE, and C.G. WU. Proper orthogonal decomposition and its applications—part i: Theory. *Journal of Sound and Vibration*, 252(3):527–544, 2002.

2. Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

3. Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587. IEEE Computer Society, 1991.

4. Jin-Chun Wang, Xin Fu, Guo-Ping Huang, Shu-Li Hong, and Yuan-Chi Zou. Application of the proper orthogonal decomposition method in analyzing active separation control with periodic vibration wall. *International Journal of Turbo & Jet-Engines*, 36(2):175–184, 2019.

5. Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

6. Harold Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936.

7. L. Sirovich and M. Kirby. Low-dimensional procedure for the characterization of human faces. *J. Opt. Soc. Am. A*, 4(3):519–524, Mar 1987.