UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Reconstrução de curvas por meio de características robustas extraídas de imagens

Relatório científico final de iniciação científica referente ao processo FAPESP nº 2020/07224-5 para o período entre 01/03/2021 e 30/09/2021.

Bolsista: André Luís Mendes Fakhoury

Orientador: João do Espirito Santo Batista Neto

Brasil
2021

Sumário

1 Introdução

Este documento tem como objetivo apresentar as atividades realizadas pelo bolsista André Luís Mendes Fakhoury no período de março de 2021 a setembro de 2021, referente ao projeto de Iniciação Científica com processo FAPESP de nº 2020/07224-5. O trabalho, intitulado "Reconstrução de curvas por meio de características robustas extraídas de imagens", é parte de umas das linhas de pesquisa do projeto temático FAPESP de nº 2019/07316-0, que visa a reconstrução de faces humanas a partir de informações reduzidas do domínio.

1.1 O projeto e plano de trabalho

O projeto de Iniciação Científica em questão está situado sob o contexto do mapeamento de características robustas entre espaços bidimensionais e tridimensionais. Este processo pode ser simplificado com a redução de informações a serem analisadas, como a representação de curvas a partir de um conjunto reduzido de pontos.

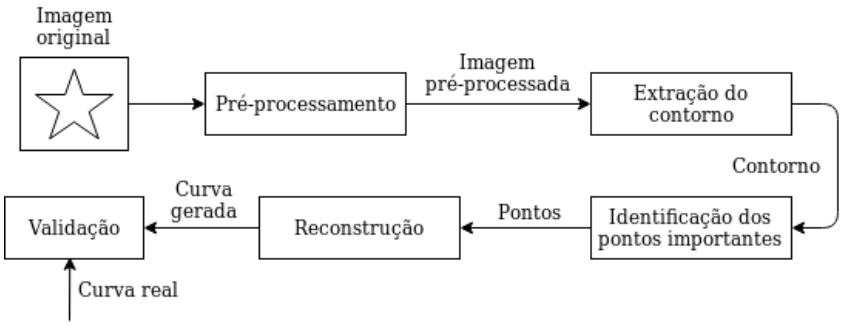
Para isso, serão analisadas principalmente curvas discretas extraídas de imagens. A partir destas curvas, pode-se analisar suas respectivas características robustas (também denominadas pontos importantes), que serão posteriormente responsáveis por reconstruir a curva original. Com isso, os seguintes objetivos específicos foram definidos para o projeto:

- Pré-processar as imagens com eliminação de ruídos, binarização e consequente segmentação;
- Extrair atributos de formas das imagens (contorno e cálculo da curvatura);
- Extrair características robustas em $\mathbb{R}^2$ para reconstrução de curvas com alta precisão a partir de imagens;
- Reconstruir as formas a partir das características robustas, por meio de curvas poligonais e operadores Laplacianos como sugerido por Sorkine (2006);
- Aferir a qualidade da reconstrução a partir da curva original.

Para atingí-los, as seguintes atividades foram desenvolvidas: estudo das técnicas de reconstrução de curvas; pré-processamento de imagens; extração dos pontos importantes; implementação da reconstrução de curvas; avaliação e testes; combinação dos algoritmos.

O diagrama visto na figura 1 descreve as principais etapas de desenvolvimento do projeto, e estas atividades foram dispostas no seguinte cronograma da tabela 1 (atualizado conforme descrito no último relatório).

Figura 1 – Diagrama de bloco das etapas de desenvolvimento



Fonte: Elaborada pelo autor.

Atividades	Meses de trabalho					
	1º e 2º	3º e 4º	5º e 6º	7º e 8º	9º e 10º	11º e 12º
Estudo das técnicas de reconstrução de curvas	•	•				
Pré-processamento		•	•			
Implementação da reconstrução de curvas		•	•			
Redação do relatório parcial			•			
Extração dos pontos importantes			•	•		
Combinação dos algoritmos				•		
Avaliação e testes					•	•
Desenvolvimento do relatório final						•

Tabela 1 – Cronograma de atividades para os 12 meses de trabalho.

As seguintes seções deste relatório descrevem com mais detalhes as atividades realizadas neste período e as considerações finais do aluno bolsista.

2 Atividades realizadas

Nesta seção serão relatadas as atividades desenvolvidas pelo aluno bolsista nesta segunda metade do projeto. A seção 2.1 descreve o processo de extração dos pontos importantes da curva discreta obtida da imagem. A seção 2.2 contém o que foi desenvolvido na etapa de combinação dos algoritmos, com a respectiva fonte para os códigos desenvolvidos. Por último, a etapa 2.3 ilustra os resultados obtidos, nos diferentes tipos de imagens e malhas utilizados para os testes.

2.1 Extração dos pontos importantes

A extração de pontos importantes consistem em, a partir de um conjunto finito de n pontos, que representa uma curva discreta, computar os índices dos pontos âncora que serão utilizados pelo algoritmo de reconstrução. A quantidade de pontos a serem escolhidos

também é enviada por parâmetro.

Alguns métodos foram considerados para a seleção dos pontos importantes. Foram eles:

- **Pontos linearmente espaçados:** os pontos são escolhidos de modo que os índices sejam igualmente espaçados entre si.
- **Escolha aleatória de pontos:** os pontos são escolhidos de modo aleatório, apenas tomando cuidado para que não sejam muito próximos um do outro.
- **Escolha de pontos pela curvatura:** inicialmente é calculada a curvatura discreta da curva em cada ponto, e aqueles com maior valor absoluto são escolhidos.

Destes, serão analisados resultados encontrados utilizando pontos linearmente espaçados e pela curvatura. A escolha de pontos igualmente espaçados pode ser realizada facilmente utilizando a função `linspace` em linguagens como Python e MATLAB. Já para a escolha de pontos pela curvatura é previamente necessária a análise da curva discreta. Para isso, será revisada a teoria da análise de curvatura discreta.

A curvatura de uma curva regular parametrizada por uma aplicação $t \to (x(t), y(t))$, onde $x(t)$ e $y(t)$ são funções de classe C^2, é dada por (OLIVEIRA; MARROQUIM, 2016):

$$\kappa(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'(t)^2 + y'(t)^2)^{3/2}}. \tag{2.1}$$

No projeto, as curvas não são contínuas, mas sim representadas por um conjunto discreto de pontos (x, y) - que representam a posição de cada pixel. Por isso, as derivadas mostradas na equação acima devem ser calculadas de maneira discreta. Uma das maneiras de se calcular isto é partir de métodos espectrais de Fourier (BRETHERTON, 2019).

Seja u_j uma aproximação discreta da função $u(x)$, com n pontos de amostra $x_j \in h, 2h, \ldots, ih, \ldots, 2\pi - h, 2\pi$, onde $h = 2\pi/n$. Para o caso discreto, pode-se aplicar a versão computacionalmente otimizada da transformada de Fourier (FFT) em u_j, tal que $FFT(u_j) \equiv \hat{u}_k$, em que $k \in \frac{-n}{2} + 1, \ldots, \frac{n}{2}$. Sabe-se que:

$$FFT\left(\frac{\partial u_j}{\partial x}\right) \equiv ik\hat{u}_k$$

Assim, para obter o valor da derivada, basta calcular a transformada rápida inversa IFFT. O código calcula a derivada de uma função discreta y no domínio x e emprega um filtro Gaussiano com fator de suavização σ. A aplicação de um filtro Gaussiano no cálculo da derivada tem por objetivo evitar a amplificação do efeito de serrilhamento (*aliasing*), quando houver, causado pela transformada de Fourier (LI, 1987). O filtro Gaussiano

cumpre este objetivo ao eliminar altas frequências presentes no sinal com a propriedade de preservar a localização de pontos importantes da mudança da curvatura (TOMASI, 2007).

Com a curvatura de cada ponto calculada, é necessário escolher os pontos que serão utilizados como âncoras para representar a curva. Porém, uma escolha simples dos pontos com maior curvatura pode fazer com que algumas regiões do contorno sejam super-representadas, enquanto outras fiquem sem pontos âncora.

Por isso, a estratégia utilizada foi a seguinte: 25% dos pontos serão escolhidos de forma linear. Os outros 75% serão enfim escolhidos por possuírem maior valor absoluto de curvatura. Para que os pontos sejam mais bem distribuídos por todo o contorno, antes de um ponto ser adicionado é feita uma verificação se este possui algum vizinho muito próximo presente no conjunto de pontos âncora já escolhidos. Caso o ponto não tenha vizinhos já adicionados no conjunto de pontos escolhidos, este pode ser então adicionado. A quantidade de vizinhos a serem analizados é constantemtente subtraída para que não haja laços de repetição infinitos. O trecho de código fonte abaixo descreve este processo.

```
# calculo da curvatura e ordenacao dos pontos por isso
kappa = calc_curvature(contour[:,0], contour[:,1], np.arange(n))
arr = sorted([(np.abs(kappa[i]), i) for i in range(n)], reverse=True)

# escolha de 25% dos pontos de forma linear
chosen = set(np.linspace(0, n - 1, qtt//4).astype(int))

# inicia com um range maior e depois diminui
delta_range = n // 20
while len(chosen) < qtt:
  for curv, idx in arr:
    can_be = len(chosen) < qtt

  # verifica se vizinhos proximos ja estao adicionados
  for delta in range(-delta_range, delta_range):
    can_be = can_be and not (idx + delta) % n in chosen

  # pode ser adicionado no conjunto
  if can_be:
    chosen.add(idx)

  # diminui o intervalo de verificacao (evitar loops infinitos)
  delta_range = max(delta_range - 5, 0)
```

2.2 Combinação dos algoritmos

Nesta seção do projeto foi implementado um *notebook* em Python que organiza todas as etapas de desenvolvimento do projeto. A versão final (FAKHOURY, 2021) foi

disponibilizada no GitHub[1].

Em grande parte, os procedimentos que envolvem imagens foram tratados com a biblioteca OpenCV (2000). As operações matriciais foram resolvidas utilizando NumPy e os gráficos feitos com a biblioteca MatPlotLib.

O *notebook* está organizado nas seguintes seções:

- **Pré-processamento** - incluindo a extração dos contornos;
- **Processamento de curvas** - a partir do conjunto de coordenadas (x, y) de cada ponto, pode ser calculada a curvatura discreta;
- **Escolha dos pontos âncora** - possibilidade de escolha dos pontos como descrito na seção anterior (2.1);
- **Reconstrução** - funções de reconstrução descritas em Sorkine (2006);
- **Testes** - finalmente, avaliações e testes, como serão descritas na seção 2.3.

2.3 Avaliação e testes

Na etapa final do projeto foram realizados testes com diversas imagens, curvas e malhas. Primeiramente serão descritas as análises utilizando as imagens de folhas do repositório ImageCLEF (2011) - inicialmente descrito no projeto como o banco de dados a ser testado - e, posteriormente, testes em curvas abertas e fechadas em $\mathbb{R}^2$ e $\mathbb{R}^3$. Também foram realizados testes em malhas poligonais, que serão descritos no último tópico desta seção.

O erro será computado como sendo a distância euclidiana entre cada ponto da curva original e da curva reconstruída: para cada índice de vértice i, o erro dos pontos obtidos $\mathbf{v}'$ em relação aos pontos originais $\mathbf{v}$ é:

$$E_i(\mathbf{v}, \mathbf{v}') = \sqrt{(\mathbf{v}_i^{(x)} - \mathbf{v}_i'^{(x)})^2 + (\mathbf{v}_i^{(y)} - \mathbf{v}_i'^{(y)})^2}$$

e o erro total da reconstrução é calculado como a soma dos erros para cada ponto:

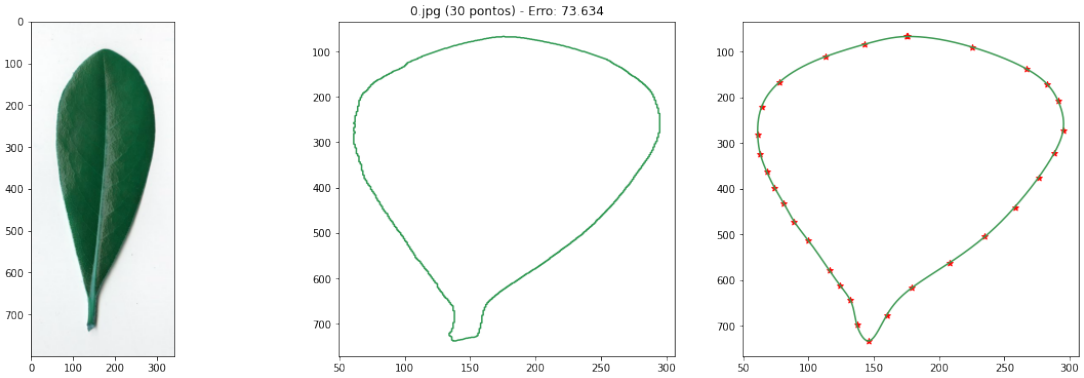
$$E(\mathbf{v}, \mathbf{v}') = \sum_{i=1}^{|V|} E_i(\mathbf{v}, \mathbf{v}')$$

2.3.1 Imagens de folhas do repositório ImageCLEF (2011)

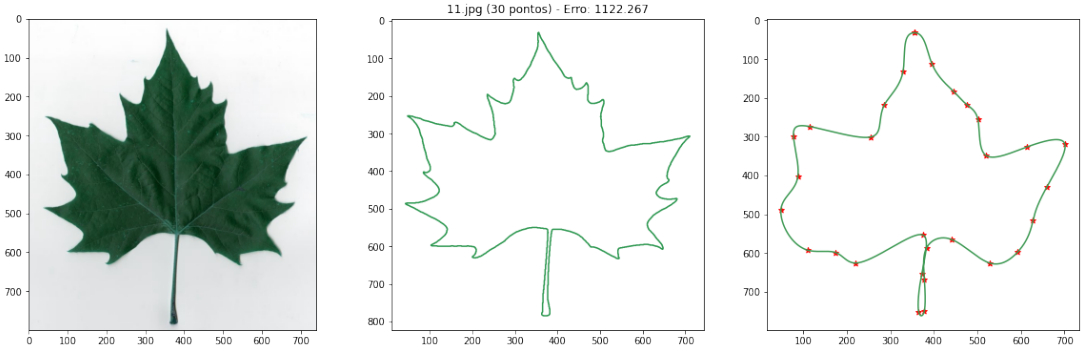
Os primeiros exemplos analisados foram imagens de folhas de árvores, retiradas do repositório do ImageCLEF (2011). Este banco de imagens foi citado no projeto original, e

[1] Disponível em <https://github.com/andrefakhoury/image-curve-reconstruction>. Acesso em 8 out 2021.

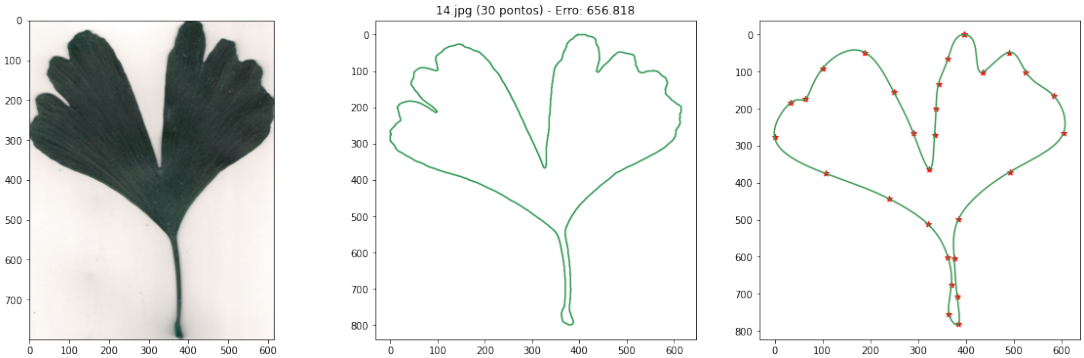
utilizado com a justificativa de que os contornos das figuras podem ser extraídos com alta precisão, facilitando a avaliação do método.



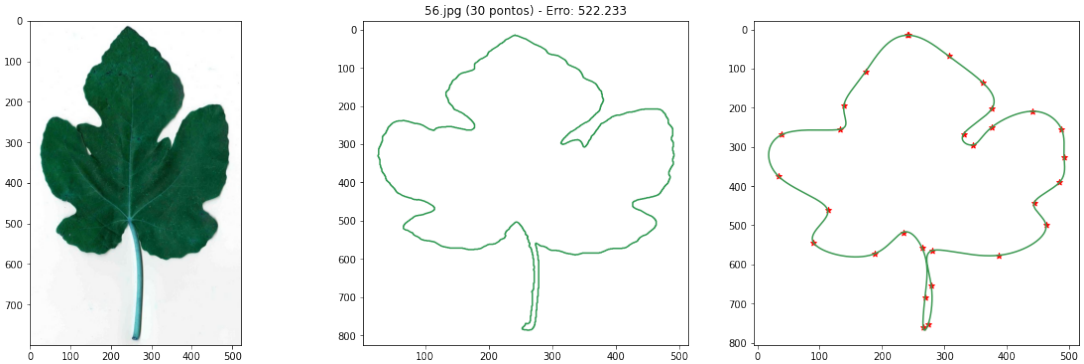
(a) Erro = 73.63.



(b) Erro = 1122.27.

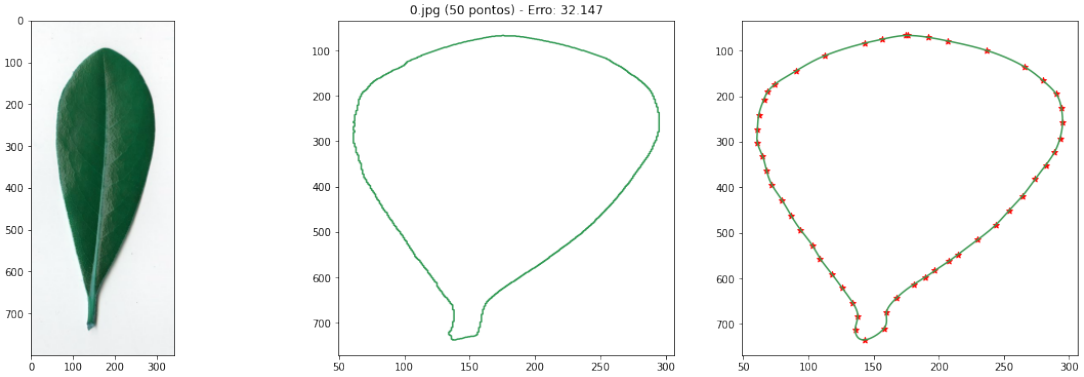


(c) Erro = 658.82.

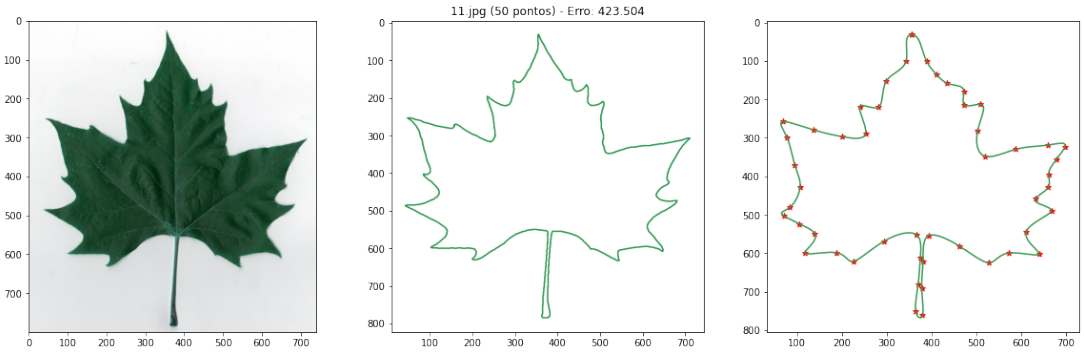


(d) Erro = 522.23.

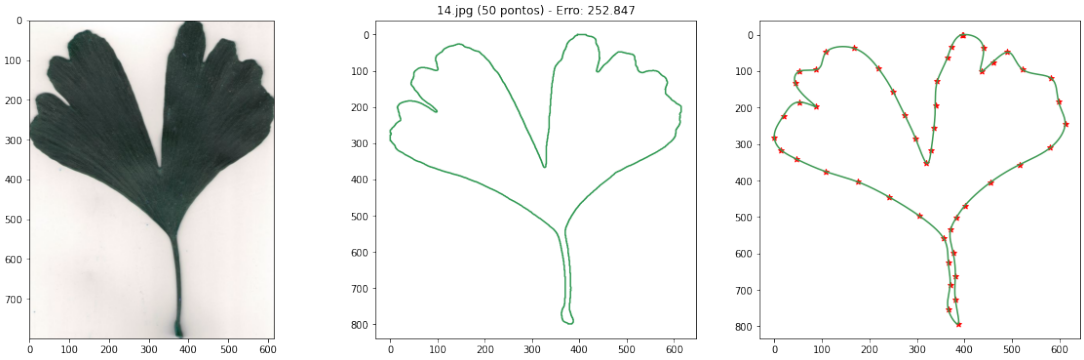
Figura 2 – Representação de folhas utilizando apenas 30 pontos como âncora.



(a) Erro = 32.15.



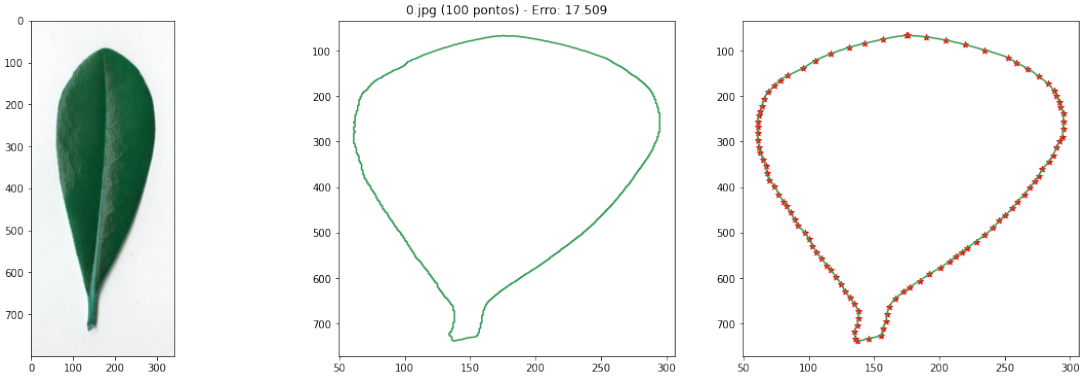
(b) Erro = 423.50.



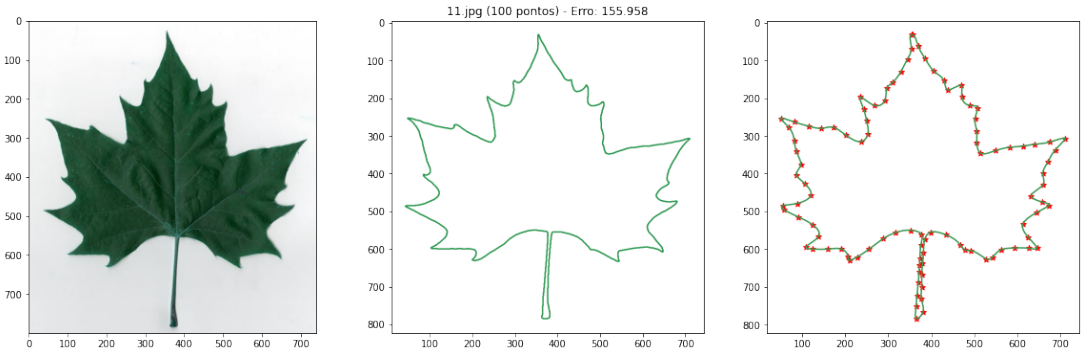
(c) Erro = 252.85.

(d) Erro = 251.18.

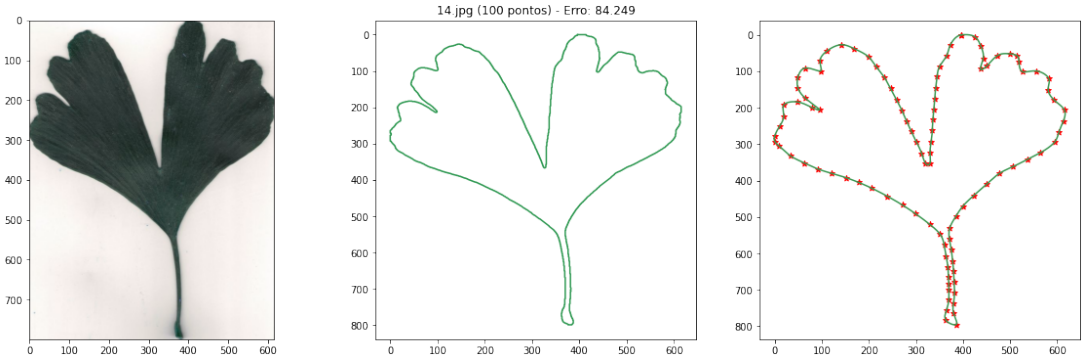
Figura 3 – Representação de folhas utilizando apenas 50 pontos como âncora.



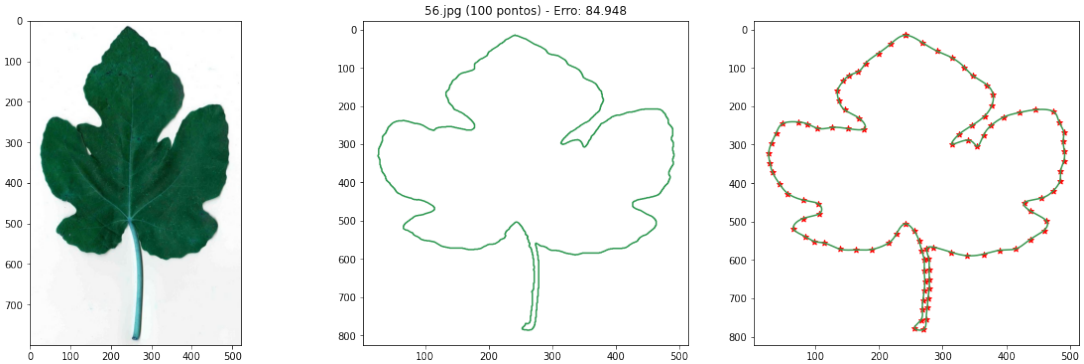
(a) Erro = 17.51.



(b) Erro = 155.96.



(c) Erro = 84.25.



(d) Erro = 84.95.

Figura 4 – Representação de folhas utilizando apenas 100 pontos como âncora.

As figuras 2, 3 e 4 apresentam os resultados para quatro folhas distintas. Em cada

uma delas, há 3 colunas: a primeira mostra a figura original, a segunda o contorno extraído e a terceira mostra a curva reconstruída, com os pontos âncora destacados com estrelas.

2.3.2 Curvas abertas e em $\mathbb{R}^3$

Outros testes foram realizados sobre curvas (discretizadas) abertas e fechadas, em domínios $\mathbb{R}^2$ e $\mathbb{R}^3$. De acordo com Sorkine (2006), o método de reconstrução funciona para quaisquer malhas que sejam aproximações lineares por partes de uma superfície suave.

A figura 5 ilustra um exemplo de reconstrução de uma função paramétrica em $\mathbb{R}^2$ e fechada. A figura 6 mostra um exemplo de função paramétrica em $\mathbb{R}^2$ e aberta. Em ambas, a figura à esquerda mostra a curva original, a do meio os pontos âncora escolhidos e a curva reconstruída a figura à direita.

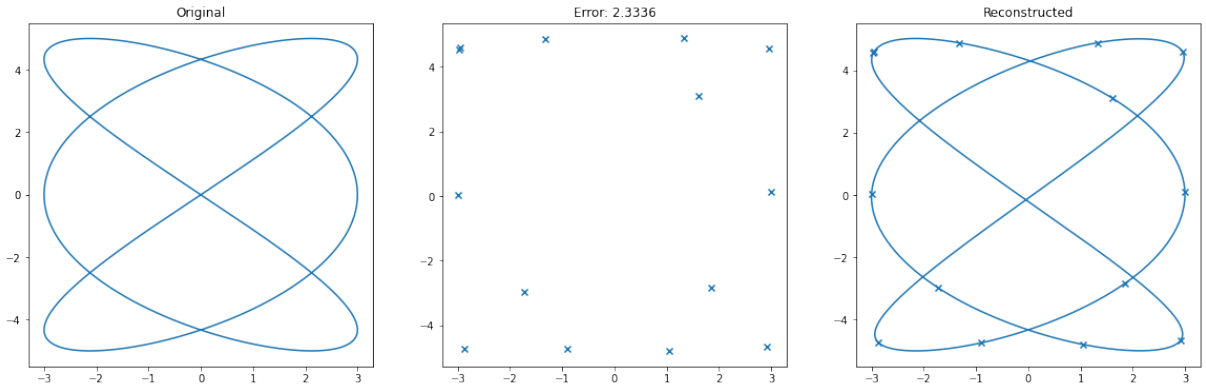


Figura 5 – Representação da função paramétrica $(x(t), y(t)) = (3cos(3t), 5sen(2t))$, com $t \in [0, 2\pi]$, utilizando 13 pontos como amostra.

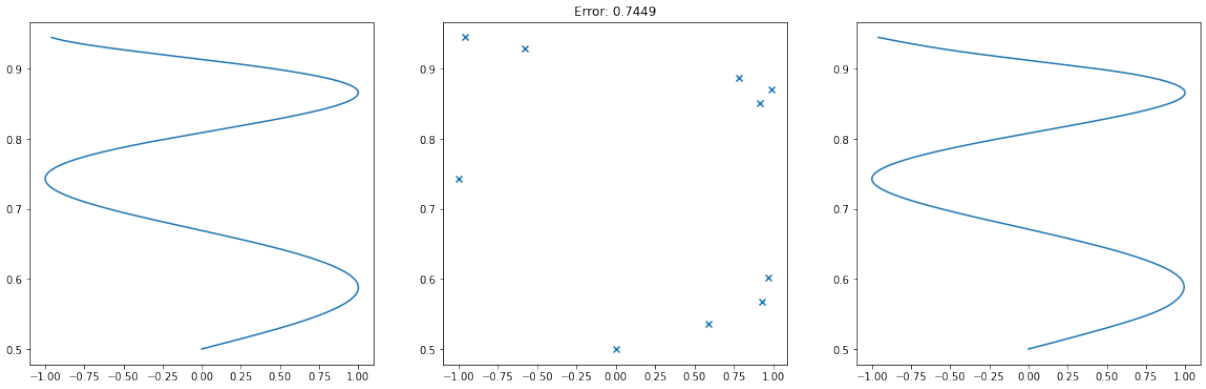


Figura 6 – Representação da função paramétrica $(x(t), y(t)) = (sin(5t), cos(t/3))$, com $t \in [1, \pi]$, utilizando 10 pontos como amostra.

A figura 7 mostra um exemplo de representação de uma curva paramétrica aberta em $\mathbb{R}^3$. A imagem à esquerda representa a curva original e a imagem à direita apresenta a curva reconstruída com os pontos âncora destacados. Neste caso, foram escolhidos pontos âncora linearmente espaçados.

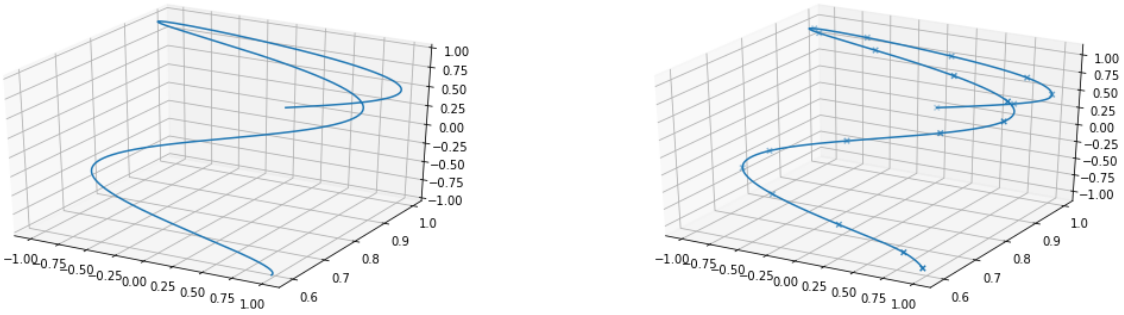
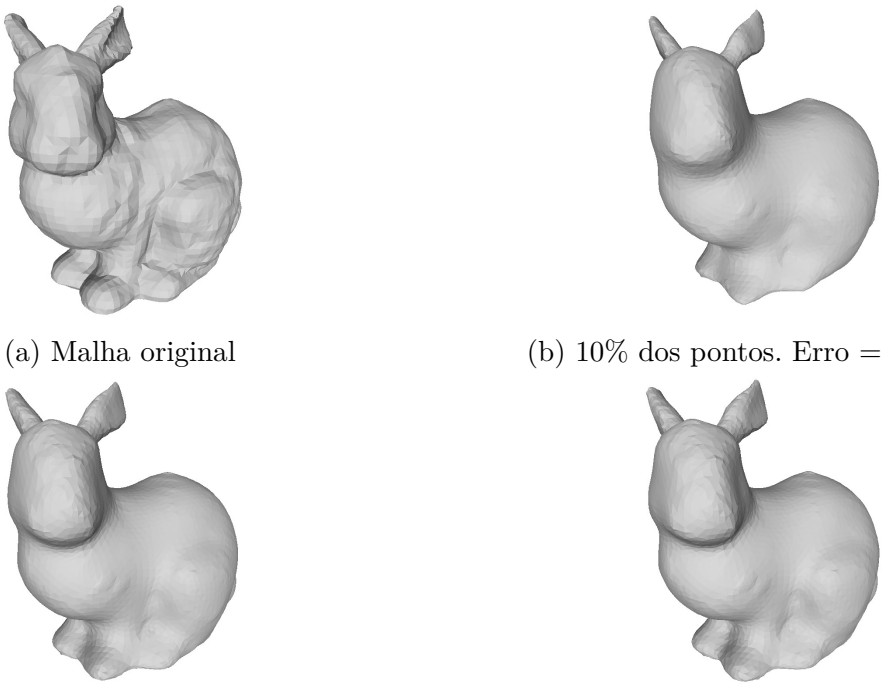


Figura 7 – Representação da função paramétrica $(x(t), y(t), z(t)) = (sin(3t), cos(t/5), sin(t))$, com $t \in [0, 1.5\pi]$, utilizando 20 pontos como amostra. Erro = 0.67.

2.3.3 Malhas poligonais

Também foram realizados testes em malhas poligonais tridimensionais. Nestes exemplos, os pontos âncora foram escolhidos a partir de índices linearmente espaçados entre si. Por isso, pode-se destacar visualmente que alguns detalhes foram perdidos.

A figura 8 mostra exemplo de reconstrução na malha de um coelho utilizando-se apenas uma amostra dos pontos. A quantidade de pontos utilizadas em cada exemplo encontra-se na legenda de cada figura.



(a) Malha original

(b) 10% dos pontos. Erro = 1037.08.

(c) 30% dos pontos. Erro = 742.05.

(d) 50% dos pontos. Erro = 590.66.

Figura 8 – Representação da malha de um coelho, utilizando apenas uma amostra dos pontos originais como âncoras.

3 Considerações finais

Durante todo o período referente ao projeto foram realizadas reuniões semanais com o grupo de professores e alunos do projeto temático em que esta iniciação científica está inserida. Nestas, foram preparados e apresentados alguns seminários, que serviram para que todos pudessem acompanhar o andamento do projeto de todos, dar *feedbacks* e planejar os próximos passos a serem executados.

Outro tópico relacionado com o projeto (e que também foi apontado no relatório parcial) foi que, durante o segundo semestre de 2020, o professor Dr. Antônio Castelo Filho, pesquisador principal do projeto temático, lecionou a disciplina "Modelagem Geométrica" para alunos de graduação e com espelho em "Tópicos em Análise Numérica II (Variedades Computacionais)" para a pós-graduação. Nela, foram abordados diversos tópicos referentes a variedades computacionais. No caso particular deste projeto, a atividade diretamente relacionada referiu-se à representação de curvas em coordenadas diferenciais pelo método descrito em Sorkine (2006). Nesta disciplina, cada estudante participou ativamente no desenvolvimento de um capítulo de um livro, que ainda será publicado.

Além disso, também foram realizados dois *workshops* relacionados ao projeto temático, em que o bolsista participou como ouvinte. O último tópico a ser mencionado é a participação deste projeto no Simpósio Internacional de Iniciação Científica e Tecnológica da USP (SIICUSP) de 2021, que será realizado em outubro de 2021.

Este relatório teve como objetivo listar as atividades realizadas no período de março a setembro de 2021 e exibir um panorama geral de desenvolvimento do projeto. Apesar de algumas dificuldades encontradas neste período (referentes à pandemia de COVID-19) e de alterações sobre o cronograma original, pode-se finalizar o projeto no tempo previsto e foi possível promover conhecimento entre os membros do grupo de pesquisa do projeto temático em que esta pesquisa está inserida.

Referências

BRETHERTON, C. *The Fourier spectral method*. 2019. Disponível em: <https://atmos.washington.edu/~breth/classes/AM585/lect/DFT_FS_585_notes.pdf>. Acesso em: 1 mar 2021.

FAKHOURY, A. L. M. *Image Curve Reconstruction*. São Carlos: GitHub, 2021. <https://github.com/andrefakhoury/image-curve-reconstruction>.

IMAGECLEF. 2011. Disponível em: <https://www.imageclef.org/2011/Plants>. Acesso em: 01 may 2020.

LI, J. *Shape Recognition by Curvature Changing Points and Fourier Transform*. Dissertação (Mestrado) — Western Michigan University, 1987.

OLIVEIRA, A.; MARROQUIM, R. *Processamento de Imagens*: aula 09 - curvatura. 2016. Disponível em: <https://www.lcg.ufrj.br/marroquim/courses/cos756/lessons/09-curvature.pdf>. Acesso em: 7 mar 2020.

OPENCV. 2000. Disponível em: <https://opencv.org>. Acesso em: 20 feb 2021.

SORKINE, O. Differential representations for mesh processing. *Computer Graphics Forum*, European Association for Computer Graphics, v. 25, n. 4, p. 789–807, 2006.

TOMASI, C. *Convolution, Smoothing, and Image Derivatives*. 2007. Disponível em: <https://www2.cs.duke.edu/courses/fall07/cps296.1/convolution.pdf>. Acesso em: 1 mar 2021.