Mateus Souza Oliveira

Espaço das projeções conformes da esfera para o plano

Florianópolis, Santa Catarina

2019

Mateus Souza Oliveira

Espaço das projeções conformes da esfera para o plano

Trabalho de Conclusão de Curso apresentado ao Curso de Matemática, do Departamento de Matemática - Centro de Ciências Físicas e Matemáticas da Universidade Federal de Santa Catarina, para obtenção de grau de Bacharel em Matemática.

Universidade Federal de Santa Catarina

Centro de Ciências Físicas e Matemáticas

Departamento de Matemática

Bacharelado em Matemática

Orientador: Prof. Dr. Leonardo Koller Sacht

Florianópolis, Santa Catarina

2019

Mateus Souza Oliveira

Espaço das projeções conformes da esfera para o plano

Trabalho de Conclusão de Curso apresentado ao Curso de Matemática, do Departamento de Matemática - Centro de Ciências Físicas e Matemáticas da Universidade Federal de Santa Catarina, para obtenção de grau de Bacharel em Matemática.

Trabalho aprovado. Florianópolis, Santa Catarina, 2019.

Prof. Dra. Silvia Martini de Holanda
Coordenadora do Curso

Banca Examinadora:

Prof. Dr. Leonardo Koller Sacht
(Orientador)
Universidade Federal de Santa Catarina

Prof. Dr. Douglas Soares Gonçalves
Universidade Federal de Santa Catarina

Prof. Dra. Marianna Ravara Vago
Universidade Federal de Santa Catarina

Florianópolis, Santa Catarina
2019

Ao meu avô.

Agradecimentos

Ingressei na universidade sem saber o que era matemática, sem saber o que queria cursar. Meu medo me impediu de fazer algum curso mais voltado a programação, mas felizmente durante este período que estive na faculdade, nas disciplinas cursadas e períodos como bolsista, tive a oportunidade de estudar e aprender um pouco sobre algumas linguagens de programação. Mas além disso, eu também aprendi outras virtudes como ter responsabilidade, organização e alcançar minhas metas. Saio maduro.

Além do título de Bacharel em Matemática e confiança para buscar o que quero no mercado de trabalho ou seguir em um mestrado, não posso deixar de agradecer às pessoas que me ajudaram nesta jornada. À minha família: mãe, pai e mana, que sempre me apoiaram. Aos professores que me ajudaram a amadurecer: Alda, Ivan, Sérgio, Fermín, Fernando, entre outros. Um obrigado especial ao meu orientador, o professor Leonardo Koller Sacht, que sempre foi atencioso e compreensivo comigo nestes 2 anos e 5 meses de pesquisa. Sem você, não teria feito um trabalho que me deixasse tão orgulhoso.

Agradecimento especial aos meus amigos. Aos meus "amigos de Crici": Rocha, Porto, Lucchese, Xanxan, Faraco, Távinho, Pedrinho, João Paulinho, entre outros. Aos meus "amigos da matemática": Ben, Sasa, Gabriel, Tata, Carlos, Sid, Carlos, Dedé, Dudu, Zaza, Jean, Lelele, Ana, entre outros. Aos meus "amigos do centro": Henrique, Bruno, Zuzu, Tiago, Monda, Gabi, entre outros. Aos meus "amigos do mestrado": Natã, Bossa, Cesinha, Aragon e Luquinhas. À minha "amiga do oeste", Gabirela. Vocês foram minha maior conquista nesse tempo que passei na faculdade. Amo vocês.

"It's not about money... it's about sending a message."
(The Joker em THE Dark Knight. Direção de Christopher Nolan. Warner Bros. Pictures, 2008.)

"Don't be what they made you..."
(Logan em LOGAN. Direção de James Mangold. 20th Century Fox, 2017.)

"You know, kid... sometimes... you find things and, uh... they change your life."
(The Punisher em THE Whirlwind (Temporada 2, ep. 13). The Punisher [Seriado]. Direção: Jeremy Webb. Netflix, 2019.)

Resumo

Imagens e fotos aparecem em nosso dia a dia, tanto nas redes sociais para entretenimento, quanto em revistas ou jornais para transmitir informações. Mas muitas vezes essas imagens apresentam uma característica em comum: campo de visão limitado. Neste trabalho, abordamos as dificuldades para obter imagens com grandes campos de visão, as ditas imagens panorâmicas.

Motivações para o estudo, além da maneira como abordamos o problema, são apresentadas. Formalizamos o problema mostrando ferramentas, motivações e as dificuldades para correção das distorções nelas presentes. Dentre elas, duas são escolhidas como principais: a preservação de linhas retas e das formas de objetos na imagem. A última, que é dita conformalidade, é estudada de um jeito mais teórico voltado a um problema novo: tentar caracterizar o espaço das projeções que satisfazem essa propriedade.

Voltado à apresentação e compreensão dos resultados práticos obtidos, um Capítulo é dedicado a mostrar as ideias por trás dos códigos implementados para o trabalho, os quais estão disponíveis na internet.

Palavras-chave: imagens panorâmicas, conformalidade, espaço das projeções conformes.

Abstract

Images and photographs appear in our everyday life in social networks for entertainment, as well as in journals and newspapers to pass informations. But many times these images show one common characteristic: limited fields of view. In this work, we approach the difficulties of obtaining images of large fields of view, also known as panoramic images.

Motivations for studying and the way we have approached the problem, are presented. We formalize the problem showing tools, motivations and the difficulties of correcting the distortions present in the images. Among them, two are chosen as majors: preserving straight lines and shapes of objects in the image. The latter, which is known as conformality, is studied in a more theoretic manner turned to a new problem: try to characterize the space of projections that respect this property.

Devoted to the presentation and comprehension of the practical results obtained, a Chapter is dedicated to showing the ideas behind the codes implemented for this work, which are available on the internet.

Keywords: panoramic images, conformality, conformal projections space.

Sumário

Introdução

Motivação

Nosso trabalho trata de imagens de alto campo de visão (também ditas imagens panorâmicas ou até mesmo panoramas). Na figura 0.0.1, podemos ver um exemplo deste tipo de imagem.

Figura 0.0.1 – Exemplo de imagem panorâmica. Imagem retirada de `https://www.flickr.com/photos/fabianosouza/`.

Existem diversas motivações para o estudo dessas imagens. Citemos algumas:

- **Sistema de monitoramento**: com vídeos panorâmicos (isto é, vídeos cujos quadros são imagens panorâmicas) é possível ter menos câmeras de monitoramento, gerando uma diminuição de consumo de energia elétrica e armazenamento de dados.
- **Realidade virtual**: são vídeos e imagens panorâmicos do campo de visão esférico total que produzem os efeitos de realidade aumentada que vemos diariamente nas redes sociais. Citamos o *Google Maps* (`https://www.google.com.br/maps/`) como exemplo desta funcionalidade.
- **Visão artística**: estas imagens são uma maneira mais completa de enxergar o mundo e podem ser usadas para produções artísticas. Citamos dois vídeos como exemplo: `https://www.youtube.com/watch?v=G7Dt9ziemYA` e `https://www.youtube.com/watch?v=F1eLeIocAcU`.

Ressaltamos que não trataremos de vídeos panorâmicos neste trabalho. Para mais informações sobre o assunto, indicamos o Capítulo 4 de (SACHT, 2010).

Mas também temos problemas ao trabalharmos com imagens panorâmicas. Na figura 0.0.2, podemos observar a presença de distorções não presentes na percepção humana do mundo real.

Figura 0.0.2 – Exemplo de imagem panorâmica. Observe o achatamento do edifício, na parte direita da imagem, e o corrimão, que de deveria ser reto, curvado, no centro da imagem. Imagem retirada de `https://www.flickr.com/photos/fabianosouza/`.

Seguindo nossas principais referências (SACHT, 2010) e (ZORIN, 1995), consideramos a preservação de linhas retas do mundo na imagem e o não esticamento de objetos como principais características a serem mantidas em uma imagem de alto campo de visão. Ambas condições são formalizadas matematicamente, sendo chamadas curvatura nula e conformalidade, respectivamente. Também é visto que não podemos ter uma imagem panorâmica satisfazendo ambas condições anteriores.

Objetivos

Tivemos um objetivo principal em nosso trabalho: caracterizar o espaço das projeções que satisfazem a condição de conformalidade, as quais chamaremos de conformes. Acreditamos que este seja um tema novo na área, o qual nos motivou a estudá-lo.

Estrutura do trabalho

No Capítulo 1, apresentamos as definições matemáticas básicas de imagens panorâmicas, seguidas de exemplos. Na sequência, apresentamos definições mais formais e os teoremas que usaremos para implementação dos códigos, além de explicar o título do

trabalho (no Capítulo 2). Prosseguindo, além de apresentar os resultados que obtivemos em nossos testes no Capítulo 3, mostramos como pensamos a escrita de diversos códigos (os quais são apresentados no Apêndice A) usados para execução de testes. Por fim, apresentamos nossas conclusões no Capítulo 4, além de apontar possíveis caminhos futuros na área.

1 Imagens Panorâmicas

Neste Capítulo, apresentaremos as primeiras definições matemáticas em nosso trabalho, com a preocupação do leitor não precisar de pré-requisitos de nível superior para a compreensão do mesmo. Além de dar as motivações para as definições, são apresentados exemplos.

1.1 Fundamentos Matemáticos

Acreditamos que o ponto de partida para o estudo de nosso tema era entender o objeto com o qual estávamos trabalhando, isto é, as imagens panorâmicas. Em (SACHT, 2010) e (ZORIN, 1995), este conceito não é apresentado formalmente (do ponto de vista matemático). Portanto, achamos importante formalizá-lo. Para tal, tivemos duas principais motivações:

- **Olhos humanos**: a motivação por trás dos panoramas é extrapolar o campo de visão usual. Assim sendo, é esperado que estas tenham um campo de visão maior que o de nossos olhos. É estimado que o campo de visão dos olhos humanos seja 120° de longitude por 60° de latitude aproximadamente ((GROSS, 2008)). Porém, não vemos tudo da mesma forma, nos extremos temos apenas uma percepção do que está lá, sendo a resolução maior no centro da imagem produzida em nosso sistema visual.
- **Câmeras fotográficas de celulares**: os celulares se tornaram companheiros inseparáveis do nosso dia a dia. Transmitir informação de maneira rápida é algo fundamental no mundo globalizado em que vivemos. Uma destas formas é através das redes sociais. Postar uma *selfie* é algo corriqueiro, mas as imagens produzidas por estes aparelhos, como foi reparado por nós, costuma ter um campo de visão menor que 90° de longitude por 90° de latitude.

Com base nisso, formalizamos uma imagem panorâmica da seguinte maneira:

Definição 1.1.1. Uma imagem cujo campo de visão é superior a 180° de longitude por 90° de latitude é dita uma imagem panorâmica, um panorama ou uma imagem com alto campo de visão.

Na figura 1.1.1, podemos ver um exemplo deste tipo de imagem.

Figura 1.1.1 – Exemplo de imagem panorâmica. Aqui, temos o campo de visão como 225° de longitude por 135° de latitude. Imagem gerada por um de nossos códigos a partir de imagem retirada de `https://www.flickr.com/photos/fabianosouza/`.

Achamos válido também apresentar um conceito mais restrito de imagens de alto campo de visão, mas que está popular atualmente. Trata-se de uma imagem que abranja todo o campo de visão, as ditas imagens equirretangulares.

Definição 1.1.2. Uma imagem cuja campo de visão é total, ou seja, 360° de longitude por 180° de latitude é dita uma imagem equirretangular.

Apresentamos um exemplo de imagem equirretangular na figura 1.1.2.

Figura 1.1.2 – Exemplo de imagem equirretangular. Note como o edifício na borda direita da imagem continua na borda esquerda, isto é, realmente enxergamos todo campo visual neste panorama. Imagem retirada de `https://www.flickr.com/photos/fabianosouza/`.

Ressaltamos que as figuras 1.1.1 e 1.1.2, foram retiradas do site FLICKR

(`https://www.flickr.com/`). A grande maioria das imagens usadas neste trabalho foram retiradas de lá. Achamos relevante citar este fato, visto que a produção de imagens panorâmicas é um assunto não trivial, e não trabalhamos com isso neste trabalho. Fazemos apenas um processamento de imagens (ver figura 1.1.3, sobre todas as sub-áreas da área chamada computação visual), isto é, dada uma imagem de entrada (no caso, do FLICKR) geramos uma outra de saída a partir desta.

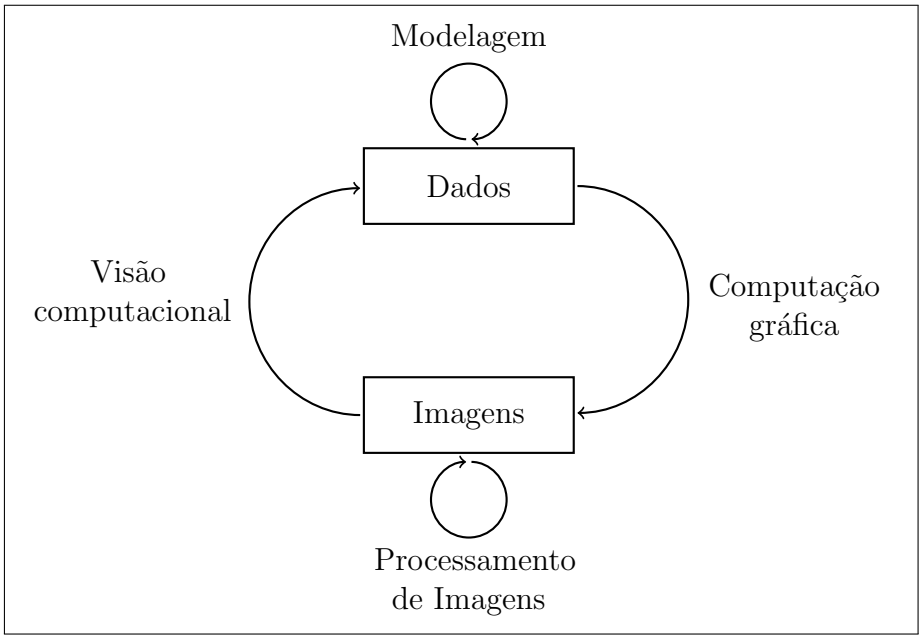


Figura 1.1.3 – Esquematização da Computação Visual, área de pesquisa deste trabalho.

Como exemplo destas dificuldades, citamos o *ghosting* (traduzindo literalmente do inglês, aparição de fantasmas), que é gerado por uma das técnicas de construção de uma imagem panorâmica: a captura de diversas imagens (não necessariamente panorâmicas) e depois a colagem dela (chamada de *stitching* na literatura em inglês). Podemos observar este efeito na figura 1.1.2, perto da borda esquerda da imagem. Para mais informações, indicamos `https://www.visgraf.impa.br/riohk/`.

A partir de agora, vamos sempre abreviar campo de visão por FOV (do inglês *field of view*), no decorrer do texto, tanto por simplicidade quanto por ser usado frequentemente nas literaturas da área. Além disso, vamos denotar o FOV como um produto cartesiano de intervalos que representam a longitude e a latitude em radianos. Por exemplo, denotamos o FOV de 180° de longitude por 90° de latitude por $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$.

Os códigos que implementamos aceitam imagens com diferentes FOVs (inclusive imagens de baixo campo de visão) de entrada. Como trataremos mais adiante no Capítulo 3 (no qual também abordamos como usá-los), basta que respeitemos as restrições do código em questão.

Agora, já sabendo o objeto matemático que iremos trabalhar, podemos prosseguir

e ver as técnicas usadas para manipulá-las.

1.2 Modelagem Matemática

Para trabalhar com as imagens equirretangulares (e panorâmicas ou até mesmo uma imagem de baixo campo de visão) neste trabalho, abordamos elas através da representação por longitudes e latitudes r:

$$\begin{aligned} r : [-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] &\longrightarrow \mathbb{S}^2 \\ (\lambda, \phi) &\longmapsto r(\lambda, \phi) = (\cos(\lambda) \cdot \cos(\phi), \operatorname{sen}(\lambda) \cdot \cos(\phi), \operatorname{sen}(\phi)), \end{aligned} \tag{1.2.1}$$

em que $\mathbb{S}^2$ é a esfera unitária centrada na origem do $\mathbb{R}^3$ (ver figura 1.2.1).

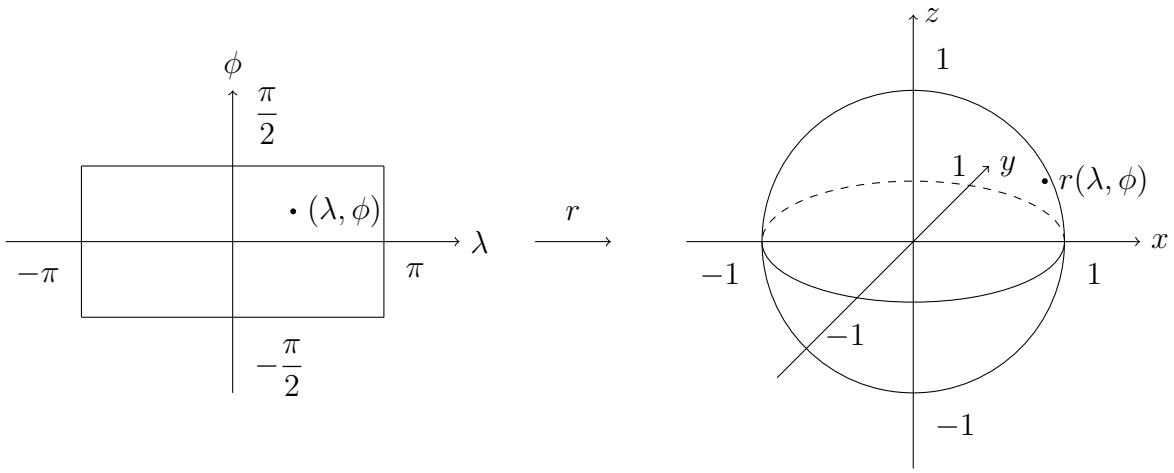


Figura 1.2.1 – Representação de como podemos associar o domínio equirretangular à esfera.

Algumas vezes durante o texto, trataremos o domínio equirretangular e a esfera unitária como o mesmo objeto, apesar de aplicação r não ser uma bijeção. Isso ocorre quando restringimos o domínio de r para $[-\pi, \pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$.

Como veremos no decorrer deste trabalho, uns dos principais assuntos do nosso tema são as distorções presentes nas imagens de alto campo de visão. Algumas são introduzidas por levarmos informações da esfera para o plano. Uma pergunta natural passa a ser "por que não deixar a informação guardada na esfera para evitar as distorções?". Para efeitos artísticos, pode ser uma ideia interessante, mas para nós que queremos algo prático, não. A tela dos computadores, dos celulares e das folhas dos livros que usamos diariamente são planas, por isso queremos que o resultado obtido também seja plano.

A seguir, apresentaremos algumas das projeções que mapeiam a esfera para o plano mais usadas no decorrer do nosso trabalho.

1.3 Projeções

Uma outra ferramenta importante para nosso trabalho são as projeções:

Definição 1.3.1. Uma função P de um subconjunto do domínio equirretangular S para o plano é dita uma projeção.

Ressaltamos que estamos considerando o domínio equirretangular e a esfera como o mesmo objeto. Portanto, podemos escrever uma projeção P como

$$P : S \subseteq \mathbb{S}^2 \longrightarrow \mathbb{R}^2. \tag{1.3.1}$$

Durante o texto podemos nos referir a uma projeção por mapeamento, transformação, função ou aplicação.

A seguir, apresentaremos as formas algébricas das principais projeções usadas neste trabalho. Para nós, será importante o conceito de domínio máximo de uma projeção, isto é, o maior subconjunto do domínio equirretangular que podemos ter como domínio da função em questão. Porém, restringiremos um pouco o conjunto, da seguinte maneira: não queremos quebras ou descontinuidades no nosso conjunto, pois isso acarretará em resultado visual não agradável ao aplicarmos em uma imagem. Em outras palavras, estamos interessados em conjuntos retangulares, isto é, produtos cartesiano de dois intervalos.

Ressaltamos que a dedução das projeções perspectiva, Mercator e estereográfica encontram-se em (SACHT, 2010).

1.3.1 Perspectiva

A projeção perspectiva é dada por

$$\begin{aligned} P : \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda,\phi), v(\lambda,\phi)) = \left(\tan(\lambda), \frac{\tan(\phi)}{\cos(\lambda)}\right). \end{aligned} \tag{1.3.2}$$

Para fixar a ideia de domínio máximo, observemos que não podemos estender o domínio de ϕ, visto que a função tangente diverge quando $\phi = \pm\frac{\pi}{2}$. Também não podemos estender o domínio de λ, pois o denominador da função v é nulo quando $\lambda = \pm\frac{\pi}{2}$ (e a função tangente também diverge nestes pontos).

Para exemplificar a projeção perspectiva (e as demais projeções da Seção), vamos usar a figura 1.3.1 como imagem equirretangular de entrada.

Figura 1.3.1 – Outro exemplo de imagem equirretangular. Imagem retirada de `https://www.flickr.com/photos/fabianosouza/`.

Vejamos na figura 1.3.2 alguns exemplos de imagens geradas usando a projeção perspectiva.

(a) FOV: $\left[-\frac{\pi}{4}, \frac{\pi}{4}\right] \times \left[-\frac{\pi}{4}, \frac{\pi}{4}\right]$.

(b) FOV: $\left[-\frac{3 \cdot \pi}{8}, \frac{3 \cdot \pi}{8}\right] \times \left[-\frac{3 \cdot \pi}{8}, \frac{3 \cdot \pi}{8}\right]$.

Figura 1.3.2 – Imagens da projeção perspectiva geradas por um de nossos códigos. O resultado de (a) é bem satisfatório (apesar do FOV extremamente limitado), mas podemos reparar esticamentos (nos paralelepípedos) perto da borda de (b).

Achamos válido ressaltar alguns pontos com base na figura 1.3.2:

- **Linhas retas**: podemos observar que os prédios ao centro da figura 1.3.1 tinham linhas curvadas em seus andares mais altos, mas na figura 1.3.2, isso não ocorre.

Em outras palavras, as linhas retas do mundo (que são curvadas pelo mapeamento equirretangular) são mapeadas em linhas retas pela projeção perspectiva.

- **Esticamentos**: apesar de termos um resultado bastante satisfatório em 1.3.2(a), vemos esticamentos bastante presentes nas bordas de 1.3.2(b). Esse efeito é bem pronunciado na projeção perspectiva, para campos de visão maiores que 120°.
- **Imagem panorâmica**: de acordo com a definição 1.1.1, não é possível gerar panoramas através da projeção perspectiva, pois esta produz campos de visão sempre inferiores a 180°.
- **Geometria da imagem**: tanto para as imagens geradas a partir da perspectiva quanto pelas outras projeções que apresentaremos nesta Seção, não esperamos que as imagens finais sejam retângulos. Isso ocorre devido a geometrias das próprias projeções, que curvam retângulos do domínio equirretangular. Aparelhos que utilizam estas projeções, não apresentam fotografias nestes formatos por fazerem um processo de corte antes de apresentar o resultado final ao usuário.

1.3.2 Mercator

A projeção de Mercator é dada por

$$\begin{aligned} M : [-\pi, \pi] \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = (\lambda, \log(\sec(\phi) + \tan(\phi))). \end{aligned} \tag{1.3.3}$$

Note que não podemos estender o domínio de ϕ, visto que a função v diverge quando $\phi \to \pm\frac{\pi}{2}$.

Vejamos na figura 1.3.3 alguns exemplos de imagens geradas usando a projeção Mercator.

(a) FOV: $\left[-\frac{\pi}{2} + 0,4, \frac{\pi}{2} - 0,4\right] \times \left[-\frac{\pi}{2} + 0,4, \frac{\pi}{2} - 0,4\right]$.

(b) FOV: $[-\pi + 0,4, \pi - 0,4] \times \left[-\frac{\pi}{2} + 0,4, \frac{\pi}{2} - 0,4\right]$.

Figura 1.3.3 – Imagens da projeção Mercator geradas por um de nossos códigos.

Achamos válido ressaltar alguns pontos com base na figura 1.3.3:

- **Linhas retas**: as linhas que eram curvadas na imagem original, continuaram curvadas (como no topo dos edifícios).
- **Proporção**: a projeção manteve a proporção dos objetos, evitando esticamentos na imagem produzida.

1.3.3 Estereográfica

A projeção estereográfica é dada por

$$E : (-\pi, \pi) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \longrightarrow \mathbb{R}^2$$
$$(\lambda, \phi) \longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = \left(\frac{2 \cdot \text{sen}(\lambda) \cdot \cos(\phi)}{\cos(\lambda) \cdot \cos(\phi) + 1}, \frac{2 \cdot \text{sen}(\phi)}{\cos(\lambda) \cdot \cos(\phi) + 1}\right). \tag{1.3.4}$$

Não é possível estender o domínio de λ, visto que o denominador das funções u e v zeram quando $\lambda = \pm\pi$ e $\phi = 0$.

Vejamos na figura 1.3.4 alguns exemplos de imagens geradas usando a projeção estereográfica.

(a) FOV: $\left[-\frac{\pi}{2} + 0,4, \frac{\pi}{2} - 0,4\right] \times \left[-\frac{\pi}{2} + 0,4, \frac{\pi}{2} - 0,4\right]$.

(b) FOV: $[-\pi + 0,4, \pi - 0,4] \times \left[-\frac{\pi}{2} + 0,4, \frac{\pi}{2} - 0,4\right]$.

Figura 1.3.4 – Imagens da projeção estereográfica geradas por um de nossos códigos.

Achamos válidos ressaltar alguns pontos com base na figura 1.3.4:

- **Linhas retas**: podemos notar as linhas retas do mundo curvadas, como nos edifícios no centro da imagem, que aparenta estar caindo.
- **Proporção**: a projeção manteve a proporção dos objetos, evitando esticamentos na imagem produzida.

1.3.4 Identidade

A projeção identidade é dada por

$$\begin{aligned} I : [-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = (\lambda, \phi)\,. \end{aligned} \tag{1.3.5}$$

Na figura 1.3.5, vemos um exemplo de imagem gerada usando a projeção identidade.

Figura 1.3.5 – Imagem da projeção identidade geradas por um de nossos códigos com FOV: $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \times \left[-\frac{\pi}{3}, \frac{\pi}{3}\right]$.

Ressaltamos que o único efeito que a projeção identidade faz na imagem de entrada é um corte do FOV selecionado. Para efeitos práticos, não é tão interessante, mas é importante para os testes que realizamos e por isso achamos importante apresentá-la.

Observamos que apesar de ter o efeito de uma função identidade, esta tem um domínio diferente de sua imagem (como todas as projeções apresentadas). Porém, chamaremos esta de projeção identidade nesse trabalho.

1.3.5 Constante

Uma projeção constante é dada por

$$\begin{aligned} C : [-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = (\lambda_0, \phi_0)\,. \end{aligned} \tag{1.3.6}$$

Note que definimos “uma” e não “a” projeção constante, isto é, para cada par de números reais, temos uma projeção distinta. Enfatizamos que esta projeção (assim como a identidade) não tem efeitos práticos para nós, mas é importante para os testes realizados. Para os testes, tomamos o ponto $(\lambda_0, \phi_0) = (2, 1)$ como sendo a imagem da projeção. Não apresentamos um exemplo, pois será apenas uma imagem de uma cor (cor do pixel na posição $(2, 1)$).

2 Espaço das Projeções Conformes

Neste Capítulo, introduzimos as proposições e teoremas (com suas devidas demonstrações) do nosso trabalho (as quais serviram de base para elaboramos os códigos). Porém, assumimos que o leitor já tenha alguma familiaridade com álgebra linear, análise e geometria diferencial, já que nossas demonstrações usaram estes conceitos. Para aqueles que não estão habituados ou ainda não estudaram estas áreas e querem ver o assunto abordado de maneira mais cautelosa, indicamos as seguintes referências: (STRANG, 1988), (LIMA, 2006), (LIMA, 2000) e (MANFREDO, 1976).

2.1 Distorções

Como podemos ver no Capítulo 1, distorções estão presentes em imagens panorâmicas. Vejamos na figura 2.1.1 as destacadas.

Figura 2.1.1 – Exemplo de distorções (destacadas em verde) que podemos encontrar em uma imagem panorâmica: linhas retas do mundo, curvadas (linha destacada) e achatamentos de objetos (elipse destacada). Imagem editada a partir de imagem retirada de `https://www.flickr.com/photos/54144402@N03/`.

Apresentamos essas em especial por, juntamente com nossas referências principais, (SACHT, 2010) e (ZORIN, 1995), considerarmos as principais:

- **Preservar linhas retas**: desejamos que as linhas que percebemos retas no mundo, sejam também retas na imagem.

- **Preservar formas e proporção**: queremos que objetos do mundo não apareçam com formas distintas (como achatamentos) e preservem proporção (não aparecendo com partes muito menores, ou maiores, do que percebemos no mundo).

Nas Seções seguintes, vamos definir estes conceitos matematicamente.

2.2 Curvatura Nula

A partir de agora vamos precisar de uma teoria mais robusta para apresentar os conceitos. Apesar de definirmos alguns conceitos prévios (baseado em quão acostumados estávamos em trabalhar com estes), indicamos que o leitor já tenha conhecimentos de geometria diferencial para total compreensão desta Seção.

Definição 2.2.1. Sejam um intervalo aberto $I \subset \mathbb{R}$ e uma função

$$\begin{aligned} \alpha : I &\longrightarrow \mathbb{R}^n \\ t &\longmapsto \alpha(t) = (x_1(t), x_2(t), \ldots, x_n(t)). \end{aligned} \tag{2.2.1}$$

A função α é dita uma curva diferenciável parametrizada quando $\forall\ i \in \{1, 2, \ldots, n\}$, as funções reais x_i são diferenciáveis.

Por simplicidade, chamaremos uma curva diferenciável parametrizada apenas de curva e sempre que tomarmos um conjunto I, fica subentendido que este é um intervalo aberto da reta real. Além disso, temos $\alpha'(t) = (x_1'(t), x_2'(t), \ldots, x_n'(t))$ e $|\cdot|$ representa a norma euclidiana, isto é, dado um vetor $x \in \mathbb{R}^n$, $|x| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$. Prosseguimos com a definição que dá o nome desta Seção.

Definição 2.2.2. Seja uma curva $\alpha : I \longrightarrow \mathbb{R}^n$. O número $|\alpha''(t)|$ é dito a curvatura de α em t.

Com esta definição, podemos apresentar a seguinte proposição:

Proposição 2.2.1. Seja uma curva $\alpha : I \longrightarrow \mathbb{R}^n$. Temos $\forall\ t \in I, |\alpha''(t)| = 0$ se, e somente se, α é uma reta.

Demonstração. ($\Leftarrow$) Lembremos que uma reta em $\mathbb{R}^n$ pode ser escrita como $r(t) = a \cdot t + b$, com $a, b \in \mathbb{R}^n$. Ou seja, $r(t) = (a_1 \cdot t + b_1, a_2 \cdot t + b_2, \ldots, a_n \cdot t + b_n)$ e portanto a curvatura de r é $|r''(t)| = |(0, 0, \ldots, 0)| = 0, \forall\ t \in I$.

($\Rightarrow$) Como $|\alpha''(t)| = 0$, temos $\alpha''(t) = 0$. Integrando $\alpha''(t) = 0$, obtemos $\alpha'(t) = k_1$, para algum $k_1 \in \mathbb{R}^n$. Agora, integrando $\alpha'(t) = k_1$, temos $\alpha(t) = k_1 \cdot t + k_2$, para algum $k_2 \in \mathbb{R}^n$. Logo, α é uma reta. ■

Com base no nosso último resultado, enunciamos a primeira condição que gostaríamos que nossas imagens apresentassem.

Condição 2.2.1 (Curvatura nula). Linhas que percebemos retas no mundo aparecem retas na imagem.

Uma ressalva importante a se fazer é como averiguamos a condição de curvatura nula em uma imagem. Apesar de termos enunciado e provado um resultado, na maioria das vezes isso é checado de forma não rigorosa, da seguinte maneira: olhamos a imagem a fim de perceber as linhas que sabemos (ou de antemão ou por experiência) serem retas no mundo mas que aparecem curvadas nas imagens. Chamamos esse processo de observação do conteúdo da imagem (o qual é a abordagem da maioria dos trabalhos da área). Voltando a figura 2.1.1, por ser tratar de uma casa, sabíamos que a linha destacada deveria ser reta (ver figura 2.2.1).

Figura 2.2.1 – Imagem da projeção perspectiva gerada por um de nossos códigos. Observe como a linha curvada anteriormente, agora aparece retificada.

Agora, vamos discutir a segunda condição que queremos que nossa imagem satisfaça, na Seção a seguir.

2.3 Conformalidade

Nesta Seção vamos introduzir o conceito de um difeomorfismo conforme, além de apresentar algumas de suas propriedades, que servirão de motivação para enunciarmos a condição de conformalidade de uma imagem. Começamos lembrando de alguns conceitos de análise e geometria diferencial.

Definição 2.3.1. Seja uma função $f : A \longrightarrow B$. A função f é dita um difeomorfismo quando é diferenciável, bijetora e sua inversa é diferenciável.

Definição 2.3.2. Sejam uma função $F : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$ diferenciável, dois pontos $p \in U$ e $w \in \mathbb{R}^n$ e uma curva $\alpha : I \longrightarrow U$ tal que $\alpha(0) = p$ e $\alpha'(0) = w$. A derivada da função $F \circ \alpha : I \longrightarrow \mathbb{R}^m$ em 0, $(F \circ \alpha)'(0)$, é dita a diferencial de F em p.
Notação: denotamos $(F \circ \alpha)'(0)$ por $dF_p(w)$.

Agora, veremos a definição do plano tangente à esfera (ver figura 2.3.1).

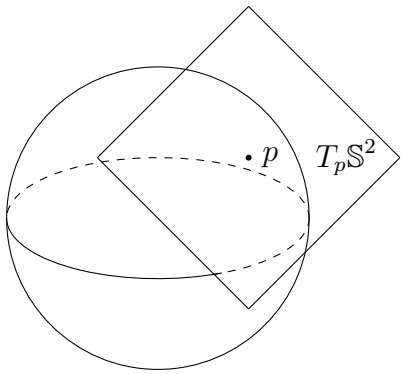


Figura 2.3.1 – Plano tangente à esfera no ponto p.

Definição 2.3.3. Sejam uma parametrização $\varphi : U \subset \mathbb{R}^2 \longrightarrow \mathbb{S}^2$ da esfera e um ponto $q \in U$. O conjunto $d\varphi_q(\mathbb{R}^2)$ é dito o plano tangente à esfera em $\varphi(q) = p$.
Notação: denotamos $d\varphi_q(\mathbb{R}^2)$ por $T_p\mathbb{S}^2$.

Munidos destes conceitos, podemos definir um difeomorfismo conforme da esfera para o plano.

Definição 2.3.4. Sejam uma função $\Theta : \mathbb{S}^2 \longrightarrow \mathbb{R}$ diferenciável tal que $\forall\ p \in \mathbb{S}^2$, temos $\Theta(p) \neq 0$ e um difeomorfismo $\psi : \mathbb{S}^2 \longrightarrow \mathbb{R}^2$. A função ψ é dita um difeomorfismo conforme quando para todo $p \in \mathbb{S}^2$ e para todo $v_1, v_2 \in T_p\mathbb{S}^2$ temos

$$\langle d\psi_p(v_1), d\psi_p(v_2)\rangle = \Theta^2(p) \cdot \langle v_1, v_2\rangle. \tag{2.3.1}$$

Observe que a função Θ da definição 2.3.4 representa uma contração ou uma expansão na imagem dos vetores da esfera quando esta é maior ou menor que 1, respectivamente.

Agora, vamos provar uma propriedade importante que os difeomorfismos conformes apresentam: a preservação de ângulos. Isto é, o ângulo entre duas curvas no domínio da função é o mesmo que o ângulo formado pelas curvas mapeadas por esta função.

Proposição 2.3.1. Seja um difeomorfismo conforme $\psi : \mathbb{S}^2 \longrightarrow \mathbb{R}^2$. O difeomorfismo ψ preserva ângulos.

Demonstração. Sejam duas curvas $\alpha, \beta : I \longrightarrow \mathbb{S}^2$ em $\mathbb{S}^2$ que se intersectam em algum ponto. Sem perda de generalidade, assuma que isto ocorre em $t = 0$, ou seja, $\alpha(0) = \beta(0)$. Lembre que o cosseno do ângulo θ_1 entre as curvas α e β em $t = 0$ é dado por

$$\cos(\theta_1) = \frac{\langle \alpha'(0), \beta'(0) \rangle}{\|\alpha'(0)\| \cdot \|\beta'(0)\|}, \tag{2.3.2}$$

para $\theta_1 \in (0, \pi)$. Agora, aplicando ψ as curvas α e β, obtemos as curvas $\psi \circ \alpha : I \longrightarrow \mathbb{R}^2$ e $\psi \circ \beta : I \longrightarrow \mathbb{R}^2$, as quais também se intersectam em $t = 0$, já que $\psi(\alpha(0)) = \psi(\beta(0))$. Lembrando que $\|a\| = \sqrt{\langle a, a \rangle}$, temos

$$\|d\psi_{\alpha(0)}(\alpha'(0))\| = \sqrt{\langle d\psi_{\alpha(0)}(\alpha'(0)), d\psi_{\alpha(0)}(\alpha'(0)) \rangle} \tag{2.3.3}$$

$$= \sqrt{\Theta^2(\alpha(0)) \cdot \langle \alpha'(0), \alpha'(0) \rangle} = \sqrt{\Theta^2(\alpha(0))} \cdot \|\alpha'(0)\|. \tag{2.3.4}$$

Logo, o cosseno do ângulo entre elas em $t = 0$ é

$$\cos(\theta_2) = \frac{\langle (\psi \circ \alpha)'(0), (\psi \circ \beta)'(0) \rangle}{\|(\psi \circ \alpha)'(0)\| \cdot \|(\psi \circ \beta)'(0)\|} = \frac{\langle d\psi_{\alpha(0)}(\alpha'(0)), d\psi_{\beta(0)}(\beta'(0)) \rangle}{\|d\psi_{\alpha(0)}(\alpha'(0))\| \cdot \|d\psi_{\beta(0)}(\beta'(0))\|}. \tag{2.3.5}$$

Lembrando que $\alpha(0) = \beta(0)$, obtemos

$$\cos(\theta_2) = \frac{\Theta^2(\alpha(0)) \cdot \langle \alpha'(0), \beta'(0) \rangle}{\Theta^2(\alpha(0)) \cdot \|\alpha'(0)\| \cdot \|\beta'(0)\|} = \frac{\langle \alpha'(0), \beta'(0) \rangle}{\|\alpha'(0)\| \cdot \|\beta'(0)\|}. \tag{2.3.6}$$

Portanto, segue que $\cos(\theta_1) = \cos(\theta_2)$ e temos o resultado. ■

O último resultado juntamente com a definição de difeomorfismo conforme, nos fornecem a seguinte informação: localmente, os objetos só podem ser rotacionados ou escalados proporcionalmente. Baseado nisso, temos os ingredientes para enunciar nossa segunda condição que gostaríamos que nossas imagens satisfizessem.

Condição 2.3.1 (Conformalidade). Objetos do mundo aparecem com o mesmo formato e proporção na imagem.

Retornando novamente a figura 2.1.1, vejamos na figura 2.3.2 um exemplo de imagem que satisfaz a condição de conformalidade.

Figura 2.3.2 – Imagem da projeção estereográfica gerada por um de nossos códigos. Veja como o telhado que estava achatado na imagem inicial do Capítulo, agora aparece proporcional.

De acordo com a definição 2.3.4, o conceito de conformalidade não é simples de ser checado na prática. Em vista disso, dedicamos a próxima Seção a apresentaremos um resultado que vai facilitar essa verificação.

2.4 Equações de Cauchy-Riemann

O objetivo dessa Seção é apresentar e provar um resultado que facilitará a checagem se um difeomorfismo é ou não conforme. Após isto, iremos apresentar o teorema que dá nome a esta Seção: o teorema das equações de Cauchy-Riemann da esfera para o plano (teorema 2.4.1). Para tal, vamos precisar introduzir mais alguns conceitos de geometria diferencial, além de conceitos de álgebra linear.

Seja $p = (\lambda, \phi) \in (-\pi, \pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$. Mostremos que o conjunto $\left\{ \frac{\partial r}{\partial \lambda}(p), \frac{\partial r}{\partial \phi}(p) \right\}$, em que

$$\frac{\partial r}{\partial \lambda}(p) = \begin{pmatrix} -\operatorname{sen}(\lambda) \cdot \cos(\phi) \\ \cos(\lambda) \cdot \cos(\phi) \\ 0 \end{pmatrix} \quad \text{e} \quad \frac{\partial r}{\partial \phi}(p) = \begin{pmatrix} -\cos(\lambda) \cdot \operatorname{sen}(\phi) \\ -\operatorname{sen}(\lambda) \cdot \operatorname{sen}(\phi) \\ \cos(\phi) \end{pmatrix}, \tag{2.4.1}$$

é a base de $T_p\mathbb{S}^2$ associada a aplicação r (equação (1.2.1)). Sejam $c_1, c_2 \in \mathbb{R}$, então temos

$$c_1 \cdot \begin{pmatrix} -\operatorname{sen}(\lambda)\cdot\cos(\phi) \\ \cos(\lambda)\cdot\cos(\phi) \\ 0 \end{pmatrix} + c_2 \cdot \begin{pmatrix} -\cos(\lambda)\cdot\operatorname{sen}(\phi) \\ -\operatorname{sen}(\lambda)\cdot\operatorname{sen}(\phi) \\ \cos(\phi) \end{pmatrix} = 0 \tag{2.4.2}$$

$$\Leftrightarrow \begin{cases} -c_1\cdot\operatorname{sen}(\lambda)\cdot\cos(\phi) - c_2\cdot\cos(\lambda)\cdot\operatorname{sen}(\phi) = 0 \\ c_1\cdot\cos(\lambda)\cdot\cos(\phi) - c_2\cdot\operatorname{sen}(\lambda)\cdot\operatorname{sen}(\phi) = 0 \\ c_2\cdot\cos(\phi) = 0 \end{cases} \Leftrightarrow c_1, c_2 = 0, \tag{2.4.3}$$

pois se $\phi \in \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$, $\cos(\phi)$ não se anula e $\operatorname{sen}(\lambda)$ e $\cos(\lambda)$ não se anulam simultaneamente. Ou seja, $\frac{\partial r}{\partial \lambda}(p)$ e $\frac{\partial r}{\partial \phi}(p)$ são linearmente independentes. Por outro lado, tomando

$$r_1(\lambda,\phi) = \cos(\lambda)\cdot\cos(\phi), r_2(\lambda,\phi) = \operatorname{sen}(\lambda)\cdot\cos(\phi) \;\text{ e }\; r_3(\lambda,\phi) = \operatorname{sen}(\phi), \tag{2.4.4}$$

temos

$$T_p\mathbb{S}^2 = dr_p\left((-\pi,\pi)\times\left(-\frac{\pi}{2},\frac{\pi}{2}\right)\right) = r'(p)\cdot w = \begin{pmatrix} \frac{\partial r_1}{\partial\lambda} & \frac{\partial r_1}{\partial\phi} \\ \frac{\partial r_2}{\partial\lambda} & \frac{\partial r_2}{\partial\phi} \\ \frac{\partial r_3}{\partial\lambda} & \frac{\partial r_3}{\partial\phi} \end{pmatrix}(p)\cdot w, \tag{2.4.5}$$

em que $w \in (-\pi,\pi)\times\left(-\frac{\pi}{2},\frac{\pi}{2}\right)$. Segue que $\left\{\frac{\partial r}{\partial\lambda}(p), \frac{\partial r}{\partial\phi}(p)\right\}$ gera $T_p\mathbb{S}^2$ e portanto é base.

Nesta Seção, vamos fixar a projeção

$$\begin{aligned} \psi : S \subset \mathbb{S}^2 &\longrightarrow \mathbb{R}^2 \\ (\lambda,\phi) &\longmapsto \psi(\lambda,\phi) = (u(\lambda,\phi), v(\lambda,\phi)). \end{aligned} \tag{2.4.6}$$

Assumindo que ψ é um difeomorfismo, temos $d\psi_p : T_p\mathbb{S}^2 \longrightarrow T_{\psi(p)}\mathbb{R}^2 = \mathbb{R}^2$, com

$$d\psi_p(w) = \begin{pmatrix} \frac{\partial u}{\partial\lambda}(p) & \frac{\partial u}{\partial\phi}(p) \\ \frac{\partial v}{\partial\lambda}(p) & \frac{\partial v}{\partial\phi}(p) \end{pmatrix}\cdot w_{\left\{\frac{\partial r}{\partial\lambda}(p), \frac{\partial r}{\partial\phi}(p)\right\}}, \tag{2.4.7}$$

em que $w_{\left\{\frac{\partial r}{\partial\lambda}(p), \frac{\partial r}{\partial\phi}(p)\right\}}$ é um vetor de $T_p\mathbb{S}^2$ escrito na base $\left\{\frac{\partial r}{\partial\lambda}(p), \frac{\partial r}{\partial\phi}(p)\right\}$. Como o produto interno entre $\frac{\partial r}{\partial\lambda}(p)$ e $\frac{\partial r}{\partial\phi}(p)$ é nulo, segue que são ortogonais, porém não têm a mesma norma, visto que

$$\left\|\frac{\partial r}{\partial\lambda}(p)\right\| = |\cos(\phi)| = \cos(\phi) \;\text{ e }\; \left\|\frac{\partial r}{\partial\phi}(p)\right\| = \sqrt{(\cos(\lambda)^2 + \operatorname{sen}(\lambda)^2)\cdot\operatorname{sen}(\phi)^2 + \cos(\phi)^2} = 1. \tag{2.4.8}$$

Assim, definimos a seguinte base ortonormal para $T_p\mathbb{S}^2$:

$$\widetilde{\frac{\partial r}{\partial \phi}(p)} = \frac{\partial r}{\partial \phi}(p) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}_{\left\{\frac{\partial r}{\partial \lambda}(p), \frac{\partial r}{\partial \phi}(p)\right\}} \tag{2.4.9}$$

e

$$\widetilde{\frac{\partial r}{\partial \lambda}(p)} = \frac{1}{\cos(\phi)} \cdot \frac{\partial r}{\partial \lambda}(p) = \begin{pmatrix} \frac{1}{\cos(\phi)} \\ 0 \end{pmatrix}_{\left\{\frac{\partial r}{\partial \lambda}(p), \frac{\partial r}{\partial \phi}(p)\right\}} . \tag{2.4.10}$$

Para abordarmos nossos próximos teoremas, precisamos relembrar as matrizes (ortogonais) de rotação por $90°$ e $-90°$:

- **Matriz de rotação por $90°$**: $\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$, que denotaremos por R_{90}.
- **Matriz de rotação por $-90°$**: $\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$, que denotaremos por R_{-90}.

Note que $R_{-90} = -R_{90}$.

Agora, munidos das ferramentas anteriores, podemos enunciar o seguinte lema, o qual servirá de base para o teorema que dá nome a esta Seção.

Lema 2.4.1. Seja um difeomorfismo $\psi : \mathbb{S}^2 \longrightarrow \mathbb{R}^2$. ψ é um difeomorfismo conforme se, e somente se,

$$d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right) = R_{\pm 90} \cdot d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right), \forall\ p \in (-\pi, \pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right). \tag{2.4.11}$$

Demonstração. ($\Rightarrow$) Como ψ é um difeomorfismo conforme, existe uma função $\Theta : \mathbb{S}^2 \longrightarrow \mathbb{R}$ diferenciável tal que $\forall\ p \in \mathbb{S}^2$, temos $\Theta(p) \neq 0$ e

$$\langle d\psi_p(v_1), d\psi_p(v_2)\rangle = \Theta(p)^2 \cdot \langle v_1, v_2\rangle, \forall\ v_1, v_2 \in T_p\mathbb{S}^2. \tag{2.4.12}$$

Tomando $v_1 = \widetilde{\frac{\partial r}{\partial \phi}(p)}$ e $v_2 = \widetilde{\frac{\partial r}{\partial \lambda}(p)}$, obtemos $\left\langle d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right), d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)\right\rangle = 0$, ou seja, $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right)$ e $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)$ são ortogonais. Agora, se tomamos $v_1 = v_2 = \widetilde{\frac{\partial r}{\partial \phi}(p)}$, temos $\left\|d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right)\right\| = |\Theta(p)|$. Por outro lado, se tomamos $v_1 = v_2 = \widetilde{\frac{\partial r}{\partial \lambda}(p)}$, obtemos $\left\|d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)\right\| = |\Theta(p)|$. Logo, $\left\|d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right)\right\| = \left\|d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)\right\|$. Isso nos leva

às seguintes conclusões: $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right)$ e $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)$ são ortogonais e têm o mesmo comprimento em $\mathbb{R}^2$. Portanto, temos apenas duas possibilidades:

- $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right) = R_{90} \cdot d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)$ (caso de v_2' na figura 2.4.1).
- $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right) = R_{-90} \cdot d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)$ (caso de v_2 na figura 2.4.1).

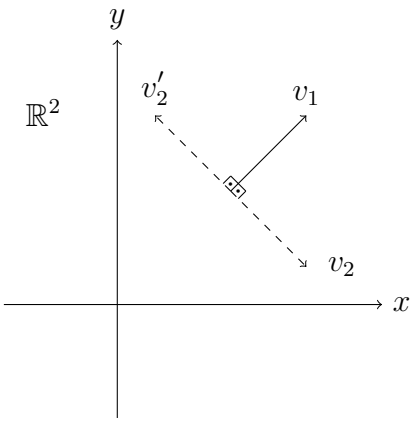


Figura 2.4.1 – Possíveis vetores ortogonais (e de mesmo comprimento) a v_1, em $\mathbb{R}^2$.

($\Leftarrow$) Seja um difeomorfismo $\psi : \mathbb{S}^2 \longrightarrow \mathbb{R}^2$ tal que $d\psi_p\left(\widetilde{\frac{\partial r}{\partial \phi}(p)}\right) = R_{\pm 90} \cdot d\psi_p\left(\widetilde{\frac{\partial r}{\partial \lambda}(p)}\right)$. Mostremos que ψ é conforme. Seja $p \in (-\pi, \pi) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right)$ e $v_1, v_2 \in T_p\mathbb{S}^2$. Como vimos, uma base ortonormal de $T_p\mathbb{S}^2$ é $\left\{\widetilde{\frac{\partial r}{\partial \lambda}(p)}, \widetilde{\frac{\partial r}{\partial \phi}(p)}\right\}$, logo existem $\alpha_1, \alpha_2, \beta_1, \beta_2 \in \mathbb{R}$ tais que $v_1 = \alpha_1 \cdot \widetilde{\frac{\partial r}{\partial \phi}(p)} + \beta_1 \cdot \widetilde{\frac{\partial r}{\partial \lambda}(p)}$ e $v_2 = \alpha_2 \cdot \widetilde{\frac{\partial r}{\partial \phi}(p)} + \beta_2 \cdot \widetilde{\frac{\partial r}{\partial \lambda}(p)}$. Então temos

$$\langle v_1, v_2\rangle = \left\langle \alpha_1 \cdot \widetilde{\frac{\partial r}{\partial \phi}(p)} + \beta_1 \cdot \widetilde{\frac{\partial r}{\partial \lambda}(p)}, \alpha_2 \cdot \widetilde{\frac{\partial r}{\partial \phi}(p)} + \beta_2 \cdot \widetilde{\frac{\partial r}{\partial \lambda}(p)} \right\rangle = \tag{2.4.13}$$

$$= \alpha_1 \cdot \left\langle \widetilde{\frac{\partial r}{\partial \phi}(p)}, \alpha_2 \cdot \widetilde{\frac{\partial r}{\partial \phi}(p)} + \beta_2 \cdot \widetilde{\frac{\partial r}{\partial \lambda}(p)} \right\rangle + \beta_1 \cdot \left\langle \widetilde{\frac{\partial r}{\partial \lambda}(p)}, \alpha_2 \cdot \widetilde{\frac{\partial r}{\partial \phi}(p)} + \beta_2 \cdot \widetilde{\frac{\partial r}{\partial \lambda}(p)} \right\rangle = \tag{2.4.14}$$

$$= \alpha_1 \cdot \alpha_2 \underbrace{\left\langle \widetilde{\frac{\partial r}{\partial \phi}(p)}, \widetilde{\frac{\partial r}{\partial \phi}(p)} \right\rangle}_{1} + \alpha_1 \cdot \beta_2 \underbrace{\left\langle \widetilde{\frac{\partial r}{\partial \phi}(p)}, \widetilde{\frac{\partial r}{\partial \lambda}(p)} \right\rangle}_{0} + \tag{2.4.15}$$

$$+ \alpha_2 \cdot \beta_1 \underbrace{\left\langle \widetilde{\frac{\partial r}{\partial \lambda}(p)}, \widetilde{\frac{\partial r}{\partial \phi}(p)} \right\rangle}_{0} + \beta_1 \cdot \beta_2 \underbrace{\left\langle \widetilde{\frac{\partial r}{\partial \lambda}(p)}, \widetilde{\frac{\partial r}{\partial \lambda}(p)} \right\rangle}_{1} = \tag{2.4.16}$$

$$= \alpha_1 \cdot \alpha_2 + \beta_1 \cdot \beta_2. \tag{2.4.17}$$

Por outro lado, temos

$$\langle d\psi_p(v_1), d\psi_p(v_2)\rangle = \tag{2.4.18}$$

$$= \left\langle \alpha_1 \cdot d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) + \beta_1 \cdot d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right), \alpha_2 \cdot d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) + \beta_2 \cdot d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right) \right\rangle = \tag{2.4.19}$$

$$= \alpha_1 \cdot \alpha_2 \cdot \left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\|^2 + \alpha_1 \cdot \beta_2 \cdot \left\langle d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right), d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right) \right\rangle + \tag{2.4.20}$$

$$+ \beta_1 \cdot \alpha_2 \cdot \left\langle d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right), d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\rangle + \beta_1 \cdot \beta_2 \cdot \left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right) \right\|^2. \tag{2.4.21}$$

Como, por hipótese, os vetores $d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right)$ e $d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right)$ são ortogonais em $\mathbb{R}^2$ e como $R_{\pm 90}$ são matrizes ortogonais (e portanto preservam norma), temos

$$\left\langle d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right), d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right) \right\rangle = \left\langle d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right), d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\rangle = 0 \tag{2.4.22}$$

e

$$\left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\| = \left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right) \right\|. \tag{2.4.23}$$

Logo, temos

$$\langle d\psi_p(v_1), d\psi_p(v_2)\rangle = \left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\|^2 \cdot (\alpha_1 \cdot \alpha_2 + \beta_1 \cdot \beta_2) = \left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\|^2 \cdot \langle v_1, v_2\rangle. \tag{2.4.24}$$

Como ψ é um difeomorfismo, $d\psi_p$ tem posto completo em todos os pontos. Assim, $\left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\|$ é uma função que não se anula. Logo, tomando $\Theta(p) = \left\| d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) \right\|$ na definição de projeção conforme, segue que ψ é conforme. ■

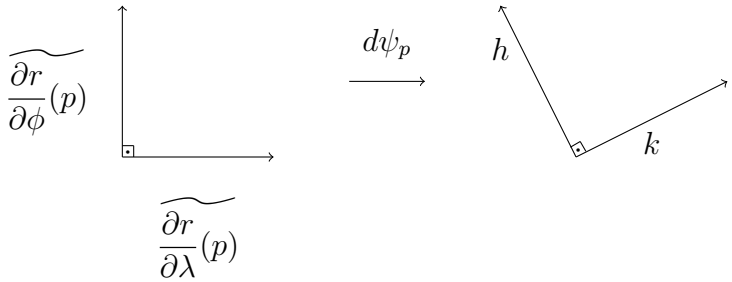
Vamos definir os seguintes vetores

$$h = d\psi_p \left(\widetilde{\frac{\partial r}{\partial \phi}(p)} \right) = \begin{pmatrix} \frac{\partial u}{\partial \lambda}(p) & \frac{\partial u}{\partial \phi}(p) \\ \frac{\partial v}{\partial \lambda}(p) & \frac{\partial v}{\partial \phi}(p) \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\partial u}{\partial \phi}(p) \\ \frac{\partial v}{\partial \phi}(p) \end{pmatrix} \tag{2.4.25}$$

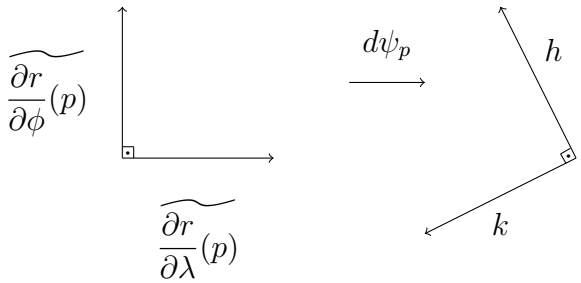
e

$$k = d\psi_p \left(\widetilde{\frac{\partial r}{\partial \lambda}(p)} \right) = \begin{pmatrix} \frac{\partial u}{\partial \lambda}(p) & \frac{\partial u}{\partial \phi}(p) \\ \frac{\partial v}{\partial \lambda}(p) & \frac{\partial v}{\partial \phi}(p) \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\cos(\phi)} \\ 0 \end{pmatrix} = \frac{1}{\cos(\phi)} \cdot \begin{pmatrix} \frac{\partial u}{\partial \lambda}(p) \\ \frac{\partial v}{\partial \lambda}(p) \end{pmatrix}. \tag{2.4.26}$$

O lema 2.4.1 nos diz que uma projeção ψ ser conforme é equivalente a $h = R_{90} \cdot k$ ou $h = R_{-90} \cdot k$. Ilustramos essa situação na figura 2.4.2.



(a) Preserva a orientação.



(b) Não preserva a orientação.

Figura 2.4.2 – Possibilidades para a imagem dos vetores $\widetilde{\frac{\partial r}{\partial \phi}(p)}$ e $\widetilde{\frac{\partial r}{\partial \lambda}(p)}$.

Pedimos que ψ preserve a orientação da base ortonormal do plano tangente, isto é, não aceitaremos quando $h = R_{-90} \cdot k$ (figura 2.4.2(b)). Isso poderia acarretar em efeito de reflexões (espelhamento) em nossos resultados de imagem. Com essas considerações, podemos finalmente apresentar o teorema que dá nome a esta Seção.

Teorema 2.4.1. Sejam um ponto $p = (\lambda, \phi)$ no interior de um subconjunto V do domínio equirretangular e

$$\begin{aligned} P : V \subset \mathbb{S}^2 &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto P(\lambda, \phi) = (u(\lambda, \phi), v(\lambda, \phi)). \end{aligned} \tag{2.4.27}$$

P é conforme e não permite reflexões se, e somente se, satisfaz as equações de Cauchy-Riemann da esfera para o plano, ou seja,

$$\frac{\partial u}{\partial \phi}(p) = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(p) \quad \text{e} \quad \frac{\partial v}{\partial \phi}(p) = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(p). \tag{2.4.28}$$

Demonstração. Sendo P conforme que não permite reflexões, temos

$$h = R_{90} \cdot k \Leftrightarrow \begin{pmatrix} \frac{\partial u}{\partial \phi}(p) \\ \frac{\partial v}{\partial \phi}(p) \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \cdot \frac{1}{\cos(\phi)} \cdot \begin{pmatrix} \frac{\partial u}{\partial \lambda}(p) \\ \frac{\partial v}{\partial \lambda}(p) \end{pmatrix} \tag{2.4.29}$$

$$\Leftrightarrow \begin{cases} \frac{\partial u}{\partial \phi}(p) = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(p) \\ \frac{\partial v}{\partial \phi}(p) = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(p) \end{cases}. \tag{2.4.30}$$

■

Observação 2.4.1. Definimos o que é um difeomorfismo conforme (definição 2.3.4), mas como veremos nos exemplos que seguem, existem projeções que não são difeomorfismos que satisfazem o teorema 2.4.1. Assim, chamaremos de projeções conformes as projeções que satisfazem esse teorema.

Para fixar a ideia do teorema, vejamos a seguir alguns exemplos de sua aplicação.

2.4.1 Exemplos

Voltando às projeções apresentadas na Seção 1.3, vejamos quais satisfazem ou não o teorema 2.4.1.

Exemplo 2.4.1. Vejamos que a projeção perspectiva P não é conforme. Lembrando que a projeção perspectiva é dada por

$$\begin{aligned} P : \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = \left(\tan(\lambda), \frac{\tan(\phi)}{\cos(\lambda)}\right), \end{aligned} \tag{2.4.31}$$

seja um ponto x no domínio da projeção P. Temos

$$\frac{\partial u}{\partial \phi}(x) = 0 \neq -\frac{1}{\cos(\phi)} \cdot \frac{\tan(\phi) \cdot \tan(\lambda)}{\cos(\lambda)} = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(x), \tag{2.4.32}$$

$$\frac{\partial v}{\partial \phi}(x) = \frac{1}{\cos(\lambda)} \cdot \frac{1}{\cos(\phi)^2} \neq \frac{1}{\cos(\phi)} \cdot \frac{1}{\cos(\lambda)^2} = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(x). \tag{2.4.33}$$

Logo, P não é conforme. Claro que bastaria termos efetuado uma das contas anteriores (2.4.32 ou 2.4.33) para mostrar o fato. Fizemos ambas apenas para fixação de ideias.

Exemplo 2.4.2. Observemos que a projeção de Mercator M é conforme. Lembrando que a projeção de Mercator é dada por

$$\begin{aligned} M : [-\pi, \pi] \times \left(-\frac{\pi}{2}, \frac{\pi}{2}\right) &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = (\lambda, \log(\sec(\phi) + \tan(\phi))), \end{aligned} \tag{2.4.34}$$

seja um ponto x no domínio da projeção M. Temos

$$\frac{\partial u}{\partial \phi}(x) = 0 = -\frac{1}{\cos(\phi)} \cdot 0 = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(x), \tag{2.4.35}$$

$$\frac{\partial v}{\partial \phi}(x) = \frac{1}{\sec(\phi) + \tan(\phi)} \cdot \frac{\operatorname{sen}(\phi) + 1}{\cos(\phi)^2} = \frac{1}{\cos(\phi)} \cdot 1 = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(x). \tag{2.4.36}$$

Portanto, M é conforme.

Exemplo 2.4.3. Notemos que a projeção estereográfica E é também conforme. Lembrando que a projeção estereográfica é dada por

$$\begin{aligned} E : (-\pi, \pi) \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = \left(\frac{2 \cdot \operatorname{sen}(\lambda) \cdot \cos(\phi)}{\cos(\lambda) \cdot \cos(\phi) + 1}, \frac{2 \cdot \operatorname{sen}(\phi)}{\cos(\lambda) \cdot \cos(\phi) + 1}\right), \end{aligned} \tag{2.4.37}$$

seja um ponto x no domínio da projeção E. Lembrando que a regra do quociente é dada por

$$\left(\frac{f}{g}\right)'(w) = \frac{f'(w) \cdot g(w) - f(w) \cdot g'(w)}{g(w)^2}, \tag{2.4.38}$$

observemos que $2 \cdot \operatorname{sen}(\lambda) \cdot (-\operatorname{sen}(\phi)) \cdot (\cos(\lambda) \cdot \cos(\phi) + 1)$ e $(2 \cdot \operatorname{sen}(\lambda) \cdot \cos(\phi)) \cdot \cos(\lambda) \cdot (-\operatorname{sen}(\phi))$ são os termos que aparecem da regra do quociente no numerador de $\dfrac{\partial u}{\partial \phi}(x)$. Por outro lado, 0 e $2 \cdot (-\operatorname{sen}(\lambda)) \cdot \cos(\phi)$ são os termos que aparecem da regra do quociente no numerador de $\dfrac{\partial v}{\partial \lambda}(x)$. Logo,

$$\frac{\partial u}{\partial \phi}(x) = \frac{-2 \cdot \operatorname{sen}(\lambda) \cdot \operatorname{sen}(\phi)}{(\cos(\lambda) \cdot \cos(\phi) + 1)^2} = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(x). \tag{2.4.39}$$

Agora, note que $2 \cdot \cos(\phi) \cdot (\cos(\lambda) \cdot \cos(\phi) + 1)$ e $2 \cdot \operatorname{sen}(\phi) \cdot (\cos(\lambda) \cdot (-\operatorname{sen}(\phi)))$ são os termos que aparecem da regra do quociente no numerador de $\dfrac{\partial v}{\partial \phi}(x)$, respectivamente. Similarmente, $2 \cdot \cos(\lambda) \cdot \cos(\phi) \cdot (\cos(\lambda) \cdot \cos(\phi) + 1)$ e $2 \cdot \cos(\lambda) \cdot \cos(\phi) \cdot (-\operatorname{sen}(\lambda) \cdot \cos(\phi))$ são os termos que aparecem da regra do quociente no numerador de $\dfrac{\partial u}{\partial \lambda}(x)$, respectivamente. Assim, segue que

$$\frac{\partial v}{\partial \phi}(x) = \frac{2 \cdot \cos(\lambda) + 2 \cdot \cos(\phi)}{(\cos(\lambda) \cdot \cos(\phi) + 1)^2} = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(x). \tag{2.4.40}$$

Portanto, E é conforme.

Exemplo 2.4.4. Vejamos que a projeção identidade não é conforme. Lembrando que a projeção identidade é dada por

$$\begin{aligned} I : [-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = (\lambda, \phi)\,, \end{aligned} \tag{2.4.41}$$

seja um ponto x no domínio da projeção I. Temos

$$\frac{\partial u}{\partial \phi}(x) = 0 = -\frac{1}{\cos(\phi)} \cdot 0 = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(x), \tag{2.4.42}$$

$$\frac{\partial v}{\partial \phi}(x) = 1 \neq \frac{1}{\cos(\phi)} \cdot 1 = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(x). \tag{2.4.43}$$

Assim, I não é conforme.

Exemplo 2.4.5. Observamos que a projeção constante é conforme. Lembrando que a projeção constante é dada por

$$\begin{aligned} C : [-\pi, \pi] \times \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] &\longrightarrow \mathbb{R}^2 \\ (\lambda, \phi) &\longmapsto (u(\lambda, \phi), v(\lambda, \phi)) = (2, 1)\,, \end{aligned} \tag{2.4.44}$$

seja um ponto x no domínio da projeção C. Temos

$$\frac{\partial u}{\partial \phi}(x) = 0 = -\frac{1}{\cos(\phi)} \cdot 0 = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v}{\partial \lambda}(x), \tag{2.4.45}$$

$$\frac{\partial v}{\partial \phi}(x) = 0 = \frac{1}{\cos(\phi)} \cdot 0 = \frac{1}{\cos(\phi)} \cdot \frac{\partial u}{\partial \lambda}(x). \tag{2.4.46}$$

Assim, C é conforme.

Para finalizar este Capítulo, apresentamos uma última Seção que explica o problema que abordamos (e consequentemente, o título de nosso trabalho) e tirando conclusões que servirão de guia para nossos códigos.

2.5 Apresentação do Problema

Agora apresentamos um resultado sem provar (a demonstração encontra-se em (ZORIN, 1995) e por envolver muitos conceitos que não foram apresentados neste trabalho, preferimos não apresentá-la) que nos levará a como foi nossa abordagem ao problema de imagens panorâmicas. Em (ZORIN, 1995), há um ampla discussão que mostra que apenas a projeção perspectiva e algumas variações dela satisfazem a condição de curvatura nula.

Como comentamos no Capítulo anterior, não podemos gerar imagens panorâmicas através da projeção perspectiva. Logo, só nos resta tentar satisfazer a condição de conformalidade com as projeções conformes. Como veremos mais na frente, temos maneiras de contornar esse problema de curvatura nula. Mas, por agora, foquemos no conjunto das projeções que satisfazem Cauchy-Riemann, o qual denotamos por $\mathscr{C}$. Prosseguindo, vamos considerar algum erro (a ser definido no próximo Capítulo) $E(P)$, para uma projeção P. Assim temos o seguinte problema:

$$\begin{aligned} &\text{minimizar} \quad E(P), \\ &\text{sujeito a} \quad P \in \mathscr{C}. \end{aligned} \tag{2.5.1}$$

Esta será nossa abordagem ao problema, que acaba priorizando a conformalidade sobre a curvatura nula. Além disso, munidos do teorema 2.4.1, apresentamos o seguinte resultado:

Proposição 2.5.1. O conjunto $\mathscr{C}$ é um espaço vetorial sobre o corpo dos números reais com as operações usuais de soma de funções e produto por escalar.

Demonstração. Primeiro, vamos fixar duas projeções conformes $P, Q : V \subset \mathbb{S}^2 \longrightarrow \mathbb{R}^2 \in \mathscr{C}, P(\lambda, \phi) = (u_P(\lambda, \phi), v_P(\lambda, \phi)), Q(\lambda, \phi) = (u_Q(\lambda, \phi), v_Q(\lambda, \phi))$ e um escalar $a \in \mathbb{R}$. Agora, vejamos que o conjunto é fechado para a multiplicação por escalar. Ou seja, verifiquemos que $a \cdot P \in \mathscr{C}$. Para tal, fixemos $x \in V$ e basta observar que a derivada é uma aplicação linear nas equações de Cauchy-Riemann, isto é,

$$\frac{\partial(a \cdot u_P)}{\partial \phi}(x) = a \cdot \frac{\partial u_P}{\partial \phi}(x) = a \cdot \left(-\frac{1}{\cos(\phi)} \cdot \frac{\partial v_P}{\partial \lambda}(x) \right) = -\frac{1}{\cos(\phi)} \cdot \frac{\partial(a \cdot v_P)}{\partial \lambda}(x), \tag{2.5.2}$$

e

$$\frac{\partial(a \cdot v_P)}{\partial \phi}(x) = a \cdot \frac{\partial v_P}{\partial \phi}(x) = a \cdot \left(\frac{1}{\cos(\phi)} \cdot \frac{\partial u_P}{\partial \lambda}(x) \right) = \frac{1}{\cos(\phi)} \cdot \frac{\partial(a \cdot u_P)}{\partial \lambda}(x). \tag{2.5.3}$$

Logo, $a \cdot P \in \mathscr{C}$. Prosseguindo, vejamos que o conjunto é fechado para a soma, isto é, $P + Q \in \mathscr{C}$. Novamente, segue da linearidade da derivada nas equações de Cauchy-Riemann, ou seja,

$$\frac{\partial(u_P + u_Q)}{\partial \phi}(x) = \frac{\partial u_P}{\partial \phi}(x) + \frac{\partial u_Q}{\partial \phi}(x) = -\frac{1}{\cos(\phi)} \cdot \frac{\partial v_P}{\partial \lambda}(x) - \frac{1}{\cos(\phi)} \cdot \frac{\partial v_Q}{\partial \lambda}(x) = \tag{2.5.4}$$

$$= -\frac{1}{\cos(\phi)} \cdot \left(\frac{\partial v_P}{\partial \lambda}(x) + \frac{\partial v_Q}{\partial \lambda}(x) \right) = -\frac{1}{\cos(\phi)} \cdot \frac{\partial(v_P + v_Q)}{\partial \lambda}(x), \tag{2.5.5}$$

e

$$\frac{\partial(v_P + v_Q)}{\partial\phi}(x) = \frac{\partial v_P}{\partial\phi}(x) + \frac{\partial v_Q}{\partial\phi}(x) = \frac{1}{\cos(\phi)} \cdot \frac{\partial u_P}{\partial\lambda}(x) + \frac{1}{\cos(\phi)} \cdot \frac{\partial u_Q}{\partial\lambda}(x) = \tag{2.5.6}$$

$$= \frac{1}{\cos(\phi)} \cdot \left(\frac{\partial u_P}{\partial\lambda}(x) + \frac{\partial u_Q}{\partial\lambda}(x) \right) = -\frac{1}{\cos(\phi)} \cdot \frac{\partial(u_P + u_Q)}{\partial\lambda}(x). \tag{2.5.7}$$

E assim, $P + Q \in \mathscr{C}$. A associatividade, comutatividade e a distributividade da soma e multiplicação por escalar seguem destas propriedades para funções reais. Como visto nos exemplos, a projeção constante de imagem $(0, 0)$ satisfaz Cauchy-Riemann, logo o elemento neutro da soma pertencem a $\mathscr{C}$. O escalar $1 \in \mathbb{R}$ é o elemento neutro da multiplicação. Por fim, pelo que vimos no início desta demonstração, $-P \in \mathscr{C}$, isto é, os inversos aditivos das projeções pertencem a $\mathscr{C}$. Portanto, $\mathscr{C}$ é um espaço vetorial sobre $\mathbb{R}$. ■

Agora que apresentamos o problema que trabalharemos e algumas propriedades, podemos começar a explicar como foram elaborados os nossos códigos, no próximo Capítulo.

3 Implementação dos Códigos e Resultados

Neste Capítulo, apresentamos o principal esforço do trabalho: como passar as ideias teóricas que foram apresentadas no decorrer do trabalho para um computador. Limitações e outros detalhes de se trabalhar com a máquina são apresentados, mas com o foco voltado às intuições que levaram à implementação de nossos códigos. Como esta abordagem foi cuidadosa, desejamos que mesmo o leitor não acostumado a programar, consiga entender as ideias. No fim, são apresentados os resultados obtidos com testes realizados por nossos próprios códigos.

3.1 Implementação dos Códigos

Nesta Seção, discutimos como pensamos a implementação de alguns dos códigos usados no trabalho. Apesar de termos usados uma linguagem de programação em específico, o MATLAB R2011b ((MATLAB, 2011)), desejamos que as ideias e intuições que apresentaremos sirvam para a implementação de códigos com os mesmos propósitos em outras linguagens de programação.

Antes de tratar cada código em específico, falemos sobre a dificuldade inicial de trabalhar com o problema no computador: a discretização. Vamos exemplificar o problema para o entendermos melhor. Suponha que queremos provar que a função $f : \mathbb{R} \longrightarrow \mathbb{R}, f(x) = e^x$ é crescente. Uma maneira seria calcular sua derivada (que neste caso, é a própria função) e ver se em todos os pontos do domínio, $f'(x) > 0$. Sabemos que a função exponencial é estritamente maior que zero e portanto é crescente. Agora, queremos que o computador faça essa tarefa para nós. Assumindo que queremos checar o teste da derivada nele, mas que o computador não saiba que a função exponencial é estritamente maior que zero nos pontos de seu domínio. O que seria feito então é o cálculo de $f'(x)$ em todos os infinitos pontos do domínio e a averiguação se em cada ponto $f'(x) > 0$. Quanto tempo essa tarefa iria tomar? Tempo infinito, o qual não temos. Para contornar esse problema, tomamos uma quantidade finita de pontos do domínio para se executar a tarefa (Note ainda que não podemos tomar quantos pontos quisermos, devemos levar em consideração a capacidade de memória da máquina). Por este motivo, na maioria das vezes não obtemos valores idênticos ao da teoria no computador, mas que são suficientes para termos intuições ou resultados práticos satisfatórios.

Precisamos então discretizar o problema, isto é, em vez de tomarmos um infinidade de pontos no domínio, tomamos um quantidade finita de pontos. Os conjuntos que trabalhamos são subconjuntos do domínio equirretangular, ou seja, um subconjunto $S \subset \mathbb{R}^2$. O que fazemos então é tomar uma quantidade finita de pontos de S para avaliar

uma projeção. Entenderemos isto melhor nas próximas subseções.

3.1.1 Malha Discretizada

Para ilustrar nossas ideias nesta Subseção e nas próximas, usaremos a figura 3.1.1 como base.

Figura 3.1.1 – Exemplo de imagem equirretangular com resolução de 2048 × 1024 *pixels*. Imagem retirada de `https://www.flickr.com/photos/54144402@N03/`.

Uma das tarefas necessárias de nossos códigos é aplicar uma projeção a uma imagem equirretangular. Para isso, é calculado a imagem de cada ponto da projeção. Na teoria, isso são infinitos pontos, na prática cada imagem tem uma limitação de pontos: a quantidade de *pixels*. Um *pixel* é um ponto da imagem que carrega as seguintes informações:

- **posição**: as coordenadas de largura e altura deste ponto.
- **cor**: a cor que este ponto carrega.

Uma resolução usual que as imagens apresentam é de 2048 × 1024 *pixels*, em que 2048 é a quantidade de *pixels* na largura e 1024 é quantidade de *pixels* na altura. Quanto maior a quantidade de *pixels*, melhor a resolução das imagens. Por isso é comum ouvirmos que uma imagem de baixa qualidade está *pixelada*.

Mesmo já tendo a limitação de quantidade de *pixels* em uma imagem, por limitação da máquina, normalmente ainda colocamos uma limitação extra: a discretização (ver figura 3.1.2). Discretizar um conjunto significa tomar uma quantidade finita de pontos dele. A partir disto, temos uma malha discretizada do conjunto. Em tese, podemos tomar este pontos de forma arbitrária, porém (como será visto mais adiante) na prática precisamos seguir uma regra: a distância dos espaçamentos deve ser uniforme. Vamos

exemplificar isto. Suponha que $P = (\alpha, \beta)$ seja um ponto da malha discretizada. Diremos que um ponto Q está na mesma linha que o ponto P se suas ordenadas são iguais. Similarmente, diremos que Q está na mesma coluna que P se suas abscissas são iguais. Assim, um conjunto será uma malha discretizada para nós se existem números reais l, h tais que todos os pontos na mesma linha que P são da forma $(\alpha \pm k \cdot l, \beta)$ e os pontos na mesma coluna que P são da forma $(\alpha, \beta \pm k \cdot h)$, para algum $k \in \mathbb{N}$. Em outras palavras, os espaçamentos horizontais e verticais não se alteram entre pontos subsequentes.

Figura 3.1.2 – Exemplo de uma malha discretizada de uma imagem equirretangular (um subconjunto do domínio equirretangular). Note como os espaçamentos longitudinais são todos iguais, assim como os espaçamentos latitudinais.

Agora que sabemos o que é uma malha discretizada, podemos passar a ideia de nosso código que faz esta tarefa. Para tal, vamos fixar alguns conceitos:

- **espaçamento longitudinal**: é o espaçamento horizontal entre dois pontos subsequentes na mesma linha.
- **espaçamento latitudinal**: é o espaçamento vertical entre dois pontos subsequentes na mesma coluna.

Por fim, apresentamos as variáveis de entrada do código.

- `lambda_min`: menor valor da abscissa dos pontos.
- `lambda_max`: maior valor da abscissa dos pontos.
- `phi_min`: menor valor da ordenada dos pontos.

- `phi_max`: maior valor da ordenada dos pontos.
- `n`: número de espaçamentos longitudinais.
- `m`: número de espaçamentos latitudinais.

Com estas variáveis podemos definir a nossa malha. `lambda_min` e `lambda_max` definem a amplitude longitudinal do FOV enquanto `phi_min` e `phi_max` a amplitude latitudinal. Note que é aqui que definimos se iremos trabalhar com uma imagem panorâmica ou não. Já `n` e `m` definem quantos pontos terão na malha. Note que teremos assim uma malha com `n`$+1$ pontos longitudinais e `m`$+1$ pontos latitudinais. Observe ainda que os espaçamentos são dados por:

$$\text{espaçamento longitudinal: } \Delta\lambda = \frac{\texttt{lambda\_max} - \texttt{lambda\_min}}{\texttt{n}} \tag{3.1.1}$$

e

$$\text{espaçamento latitudinal: } \Delta\phi = \frac{\texttt{phi\_max} - \texttt{phi\_min}}{\texttt{m}}. \tag{3.1.2}$$

Na figura 3.1.3, podemos ver uma esquematização dessas ideias.

Figura 3.1.3 – Esquematização de ideias do código de malha discretizada. Os círculos pretos na imagem de saída representam os pontos da malha.

Agora, falta explicar em qual ordem os pontos são tomados na malha discretizada (como veremos na sequência, isto também faz diferença para a implementação de códigos mais a frente). Escolhemos fazer isso no procedimento chamado de por linhas, isto é, as linhas da malha são completadas antes das colunas. No nosso caso, os pontos são distribuídos na malha de baixo pra cima, e da esquerda para a direita. Denotamos os pontos da malha da seguinte maneira: $P_{\text{coluna,linha}}$. Portanto, o ponto na primeira coluna e na segunda linha será denotado por $P_{1,2}$. Exemplifiquemos isto na figura 3.1.4.

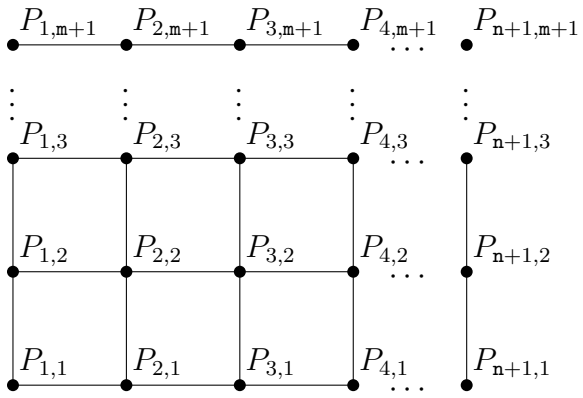
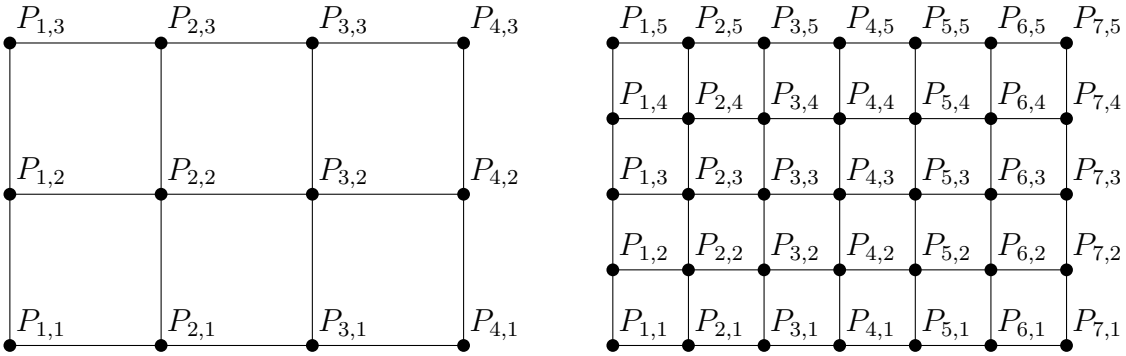


Figura 3.1.4 – Exemplo de malha discretizada com `n` espaçamentos longitudinais e `m` espaçamentos latitudinais, totalizando $(\mathtt{n}+1)\times(\mathtt{m}+1)$ pontos no total.

Uma malha com poucos pontos é dita uma malha grosseira. Para termos uma malha mais próxima da teoria fazemos um refinamento dela (ver figura 3.1.5), obtendo uma malha fina. Como comentamos, temos de respeitar a capacidade de processamento da máquina e além de tomar cuidado com a aritmética finita do computador.



(a) Malha grosseira com `n=3` e `m=2`. (b) Malha fina com `n=6` e `m=4`.

Figura 3.1.5 – Exemplos de uma malha grosseira (a) e um refinamento dela (b).

Na implementação, armazenamos as posições dos pontos da malha como vetores. Apresentamos as variáveis de saída do código.

- `Dlambda`: vetor do $\mathbb{R}^{\mathtt{n}+1}$ que armazena as coordenadas longitudinais dos pontos da malha.
- `Dphi`: vetor do $\mathbb{R}^{\mathtt{m}+1}$ que armazena as coordenadas latitudinais dos pontos da malha.

Agora, vejamos como acessar as informações de posição de cada ponto da malha discretizada. Sabendo que o comando $\mathtt{Dlambda}(i)$ acessa a i-ésima entrada do vetor `Dlambda` e $\mathtt{Dphi}(j)$ acessa a j-ésima entrada do vetor `Dphi`, temos a seguinte regra:

$$P_{i,j} = (\mathtt{Dlambda}(i), \mathtt{Dphi}(j))\,. \tag{3.1.3}$$

Vejamos um exemplo de uso do código.

Exemplo 3.1.1. Colocando como variáveis de entrada: `lambda_min`=-2, `lambda_max`=2, `phi_min`=-1, `phi_max`=1, `m`=1 e `n`=2, recebemos de variáveis de saída:

$$\texttt{Dlambda} = \begin{pmatrix} -2 & 0 & 2 \end{pmatrix} \text{ e } \texttt{Dphi} = \begin{pmatrix} -1 & 1 \end{pmatrix}.$$

Na figura 3.1.6, podemos ver os pontos $P_{i,j}$ obtidos desta discretização.

Figura 3.1.6 – Exemplo de como os pontos são distribuídos na malha discretizada. Nesse caso temos `lambda_min`=-2, `lambda_max`=2, `phi_min`=-1, `phi_max`=1, `m`=1 e `n`=2.

Implementamos o código A.2.1 para realizar estas tarefas.

3.1.2 Projeções Discretizadas

Agora, nossa ideia é tomar a imagem por uma projeção dos pontos de uma malha discretizada, ou seja, aplicar uma projeção aos pontos da malha. Novamente, vamos usar um vetor para guardar essas informações. Para entendermos melhor, vejamos primeiramente as variáveis de entrada do código:

- `Dlambda`: vetor que armazena as coordenadas longitudinais dos pontos da malha.
- `Dphi`: vetor que armazena as coordenadas latitudinais dos pontos da malha.

Como veremos nos códigos seguintes, muitas vezes usaremos resultados de códigos anteriores como entrada de outros códigos. Neste caso, a melhor maneira é usar o código A.2.1 para produzir as variáveis de entrada dos códigos das projeções discretizadas. Como variável de saída, temos:

- `X`: vetor de dimensão igual ao dobro do números de pontos na malha discretizada.

Dada uma projeção $T : S \subseteq \mathbb{S}^2 \longrightarrow \mathbb{R}^2, T(P_{i,j}) = (u(P_{i,j}), v(P_{i,j}))$, em uma malha de $(\mathtt{n}+1) \times (\mathtt{m}+1)$ pontos, gostaríamos de ter

$$\mathtt{X} = \Big(\; T(P_{1,1}) \;\; T(P_{2,1}) \;\; T(P_{3,1}) \;\; \cdots \;\; T(P_{\mathtt{m},\mathtt{n}+1}) \;\; T(P_{\mathtt{m}+1,\mathtt{n}+1}) \; \Big)^T_{(\mathtt{n}+1)\cdot(\mathtt{m}+1)\times 1}.$$

Para contornar o problema de $T(P_{i,j})$ ser um elemento de $\mathbb{R}^2$, fazemos

$$\mathtt{X} = \Big(\; u(P_{1,1}) \;\; v(P_{1,1}) \;\; u(P_{2,1}) \;\; v(P_{2,1}) \;\; \cdots \;\; u(P_{\mathtt{m}+1,\mathtt{n}+1}) \;\; v(P_{\mathtt{m}+1,\mathtt{n}+1}) \; \Big)^T_{2\cdot(\mathtt{n}+1)\cdot(\mathtt{m}+1)\times 1}.$$

Agora a dificuldade é como acessar as informações guardadas no vetor `X`. Lembre que alocamos as informações na ordem da malha, isto é, linhas de baixo para cima. Assim, cada $2 \cdot (\mathtt{n}+1)$ entradas de `X` contem informações de uma linha. Por simplicidade, vamos denotar $u(P_{i,j}) = u_{i,j}$ e $v(P_{i,j}) = v_{i,j}$. Observe então que o vetor `X` tem a seguinte estrutura

$$X = \left(\begin{array}{c} u_{1,1} \\ v_{1,1} \\ \vdots \\ u_{\mathtt{n}+1,1} \\ v_{\mathtt{n}+1,1} \end{array} \right\} 2\cdot(\mathtt{n}+1) \text{ elementos bloco 1}$$
$$\left. \begin{array}{c} u_{1,2} \\ v_{1,2} \\ \vdots \\ u_{\mathtt{n}+1,2} \\ v_{\mathtt{n}+1,2} \end{array} \right\} 2\cdot(\mathtt{n}+1) \text{ elementos bloco 2}$$
$$\vdots$$
$$\left. \begin{array}{c} u_{1,\mathtt{m}+1} \\ v_{1,\mathtt{m}+1} \\ \vdots \\ u_{\mathtt{n}+1,\mathtt{m}+1} \\ v_{\mathtt{n}+1,\mathtt{m}+1} \end{array} \right) \} 2\cdot(\mathtt{n}+1) \text{ elementos bloco } \mathtt{m}+1 \quad .$$

Note que as informações do ponto $(u_{i,j}, v_{i,j})$ está no bloco j. Logo, para retirar a informação de $T(P_{i,j})$ devemos retirar as entradas $(j-1)\cdot 2\cdot(\mathtt{n}+1) + 2\cdot i - 1$ (para a $u_{i,j}$) e $(j-1)\cdot 2\cdot(\mathtt{n}+1) + 2\cdot i$ (para a $v_{i,j}$).

Vejamos um exemplo para fixar a ideia.

Exemplo 3.1.2. Para este exemplo, vamos adotar a projeção perspectiva (cujo código é o A.2.2). Mantendo `Dlambda` e `Dphi` (e tomando estes como variáveis de entrada) como

obtidos no exemplo 3.1.1, recebemos de variável de saída:

$$\mathtt{X} = \begin{pmatrix} u_{1,1} = \tan(-2) \\ v_{1,1} = \dfrac{\tan(-1)}{\cos(-2)} \\ u_{2,1} = \tan(0) \\ v_{2,1} = \dfrac{\tan(-1)}{\cos(0)} \\ u_{3,1} = \tan(2) \\ v_{3,1} = \dfrac{\tan(-1)}{\cos(2)} \\ u_{1,2} = \tan(-2) \\ v_{1,2} = \dfrac{\tan(1)}{\cos(-2)} \\ u_{2,2} = \tan(0) \\ v_{2,2} = \dfrac{\tan(1)}{\cos(0)} \\ u_{3,2} = \tan(2) \\ v_{3,2} = \dfrac{\tan(1)}{\cos(2)} \end{pmatrix}_{12\times 1}.$$

Implementamos os seguintes códigos para as demais projeções apresentadas na Seção 1.3: códigos A.2.3, A.2.4, A.2.5 e A.2.6.

3.1.3 Geração de Imagens

Nosso próximo código gera um resultado visual a partir dos resultados dos códigos anteriores. Exemplificamos isto na figura 3.1.7.

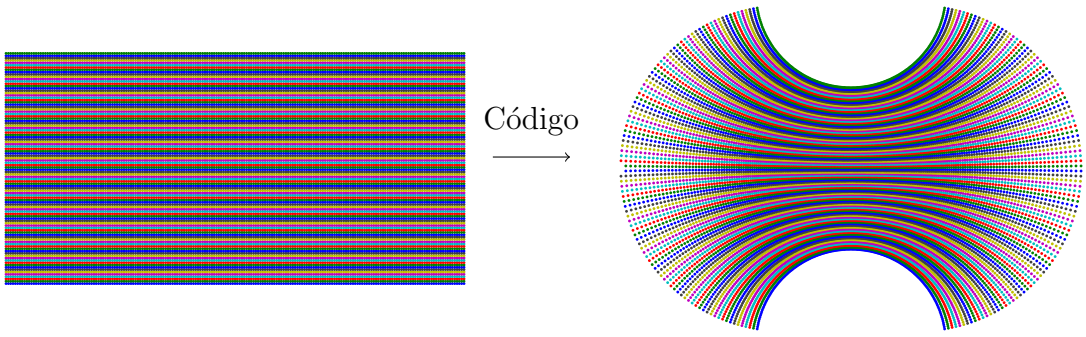


Imagem equirretangular de entrada Imagem de saída

Figura 3.1.7 – Desejamos a partir de uma imagem equirretangular de entrada produzir um resultado visual da imagem de uma projeção. Neste exemplo, a projeção tomada foi a estereográfica.

Selecionamos a porção da imagem que desejamos trabalhar (FOV), com quantos

pontos iremos trabalhar e qual projeção será utilizada. Depois é calculado a imagem de cada ponto da malha discretizada e produzido o resultado gráfico. Chamamos este processo de *plot*. Ressaltamos que a imagem de cada ponto será a cor associada ele, na nova posição. Como já abordamos, uma malha mais fina irá produzir resultados visuais mais agradáveis. Na figura 3.1.7, podemos ver na imagem de saída como a densidade de pontos na borda é inferior se comparada ao centro dela.

Exemplo 3.1.3. Para este exemplo, vamos usar a projeção estereográfica (cujo código é o A.2.4) e tomar `Dlambda` e `Dphi` no código A.2.1 com `lambda_min`=-2, `lambda_max`=2, `phi_min`=-1, `phi_max`=1, `n`=99 e `m`=49. Vemos o resultado na figura 3.1.8.

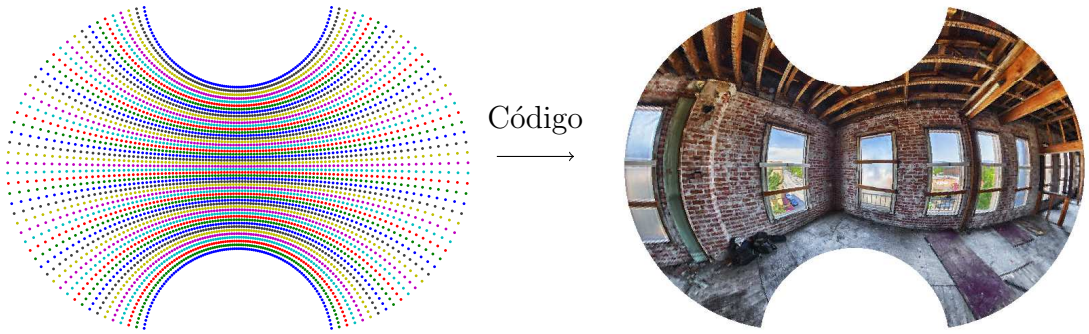


Imagem dos pontos da malha discretizada Imagem de saída

Figura 3.1.8 – Esquematização do processo que o código realiza.

Para este trabalho, implementamos o código A.2.7.

3.1.4 Conformalidade Discretizada

Nosso objetivo agora é discretizar as equações de Cauchy-Riemann da esfera para o plano. Como isto envolve equações diferenciais parciais (E.D.P.s), precisamos de um método de discretizações de E.D.P.s. Existem diversas técnicas para efetuarmos essas discretizações. Escolhemos usar o método de diferenças finitas, por precisarmos aproximar as derivadas parciais em uma malha regular. Faremos uma breve introdução ao método para entendermos como foi feito o código voltado a resolver as equações de Cauchy-Riemann de forma discretizada. Para mais exemplos de métodos e uma abordagem mais cautelosa do método de diferenças finitas, indicamos (BURDEN; FAIRES, 2008).

Lembremos que as derivadas parciais de uma função $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$ em um ponto (a, b) são dadas por

$$\frac{\partial f}{\partial x}(a,b) = \lim_{l\to 0} \frac{f(a+l,b) - f(a,b)}{l} \quad \text{e} \quad \frac{\partial f}{\partial y}(a,b) = \lim_{h\to 0} \frac{f(a,b+h) - f(a,b)}{h}. \tag{3.1.4}$$

Então, dada uma projeção $T : S \subseteq \mathbb{S}^2 \longrightarrow \mathbb{R}^2,\ T(\lambda, \phi) = (u(\lambda, \phi), v(\lambda, \phi))$, as derivadas parciais das funções u e v em um ponto $P_{i,j}$ são

$$\frac{\partial u}{\partial \lambda}(P_{i,j}) = \lim_{\Delta\lambda \to 0} \frac{u_{i+\Delta\lambda,j} - u_{i,j}}{\Delta\lambda},\ \frac{\partial u}{\partial \phi}(P_{i,j}) = \lim_{\Delta\phi \to 0} \frac{u_{i,j+\Delta\phi} - u_{i,j}}{\Delta\phi}, \tag{3.1.5}$$

$$\frac{\partial v}{\partial \lambda}(P_{i,j}) = \lim_{\Delta\lambda \to 0} \frac{v_{i+\Delta\lambda,j} - v_{i,j}}{\Delta\lambda} \text{ e } \frac{\partial v}{\partial \phi}(P_{i,j}) = \lim_{\Delta\phi \to 0} \frac{v_{i,j+\Delta\phi} - v_{i,j}}{\Delta\phi}. \tag{3.1.6}$$

Dados $\Delta\lambda > 0$ (equação 3.1.1) e $\Delta\phi > 0$ (equação 3.1.2), uma maneira de discretizar as derivadas parciais das funções u e v em um ponto $P_{i,j}$ seria tomar

$$\frac{\partial u}{\partial \lambda}(P_{i,j}) \approx \frac{u_{i+\Delta\lambda,j} - u_{i,j}}{\Delta\lambda},\ \frac{\partial u}{\partial \phi}(P_{i,j}) \approx \frac{u_{i,j+\Delta\phi} - u_{i,j}}{\Delta\phi}, \tag{3.1.7}$$

$$\frac{\partial v}{\partial \lambda}(P_{i,j}) \approx \frac{v_{i+\Delta\lambda,j} - v_{i,j}}{\Delta\lambda} \text{ e } \frac{\partial v}{\partial \phi}(P_{i,j}) \approx \frac{v_{i,j+\Delta\phi} - v_{i,j}}{\Delta\phi}. \tag{3.1.8}$$

Aqui, como estamos tomando espaçamentos longitudinais $\Delta\lambda$ e latitudinais $\Delta\phi$ constantes, podemos reescrever as aproximações como

$$\frac{\partial u}{\partial \lambda}(P_{i,j}) \approx \frac{u_{i+1,j} - u_{i,j}}{\Delta\lambda},\ \frac{\partial u}{\partial \phi}(P_{i,j}) \approx \frac{u_{i,j+1} - u_{i,j}}{\Delta\phi}, \tag{3.1.9}$$

$$\frac{\partial v}{\partial \lambda}(P_{i,j}) \approx \frac{v_{i+1,j} - v_{i,j}}{\Delta\lambda} \text{ e } \frac{\partial v}{\partial \phi}(P_{i,j}) \approx \frac{v_{i,j+1} - v_{i,j}}{\Delta\phi}. \tag{3.1.10}$$

Note que as aproximações são mais próximas das reais derivadas à medida que $\Delta\lambda \to 0$ e $\Delta\phi \to 0$. Ou seja, trabalhar com malhas mais finas resultará em aproximações mais condizentes com a teoria.

Portanto, discretizamos as equações de Cauchy-Riemann (equação 2.4.28) da seguinte forma

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta\phi} = -\frac{1}{\cos(\texttt{Dphi}(j))} \cdot \frac{v_{i+1,j} - v_{i,j}}{\Delta\lambda} \text{ e } \frac{v_{i,j+1} - v_{i,j}}{\Delta\phi} = \frac{1}{\cos(\texttt{Dphi}(j))} \cdot \frac{u_{i+1,j} - u_{i,j}}{\Delta\lambda}. \tag{3.1.11}$$

Note que $\cos(\texttt{Dphi}(j))$ pode ser um valor muito próximo a zero gerando valores muito altos nas contas das equações. Então, para evitar problemas de aritmética com este fator, multiplicamos as equações por $\cos(\texttt{Dphi}(j))$, obtendo

$$\cos(\texttt{Dphi}(j)) \cdot \frac{u_{i,j+1} - u_{i,j}}{\Delta\phi} + \frac{v_{i+1,j} - v_{i,j}}{\Delta\lambda} = 0 \tag{3.1.12}$$

e

$$\cos(\texttt{Dphi}(j)) \cdot \frac{v_{i,j+1} - v_{i,j}}{\Delta\phi} - \frac{u_{i+1,j} - u_{i,j}}{\Delta\lambda} = 0. \tag{3.1.13}$$

Observe que queremos avaliar estas equações nos $(\texttt{n}+1) \times (\texttt{m}+1)$ pontos da malha, porém na última linha (e na última coluna) não existem pontos mais à direita e mais acima, não

sendo possível avaliar 3.1.12 e 3.1.13. Para avaliar o quanto uma projeção discretizada é conforme, definimos o seguinte erro de conformalidade:

$$E_C = \sum_{i=1}^{\mathtt{n}} \sum_{j=1}^{\mathtt{m}} \left(\cos(\mathtt{Dphi}(j)) \cdot \frac{u_{i,j+1} - u_{i,j}}{\Delta\phi} + \frac{v_{i+1,j} - v_{i,j}}{\Delta\lambda} \right)^2 + \tag{3.1.14}$$

$$+ \sum_{i=1}^{\mathtt{n}} \sum_{j=1}^{\mathtt{m}} \left(\cos(\mathtt{Dphi}(j)) \cdot \frac{v_{i,j+1} - v_{i,j}}{\Delta\phi} - \frac{u_{i+1,j} - u_{i,j}}{\Delta\lambda} \right)^2. \tag{3.1.15}$$

Este erro é a soma dos quadrados das discretizações das equações de Cauchy-Riemann nos $\mathtt{n} \times \mathtt{m}$ pontos da malha que podemos fazer essa avaliação. Na teoria, este somatório seria nulo para as projeções conformes e maior que zero para as não conformes. Como veremos na Seção de resultados, na prática temos um valor baixo para as conformes e um valor alto para as não conformes. O motivo de tomarmos o quadrado de cada discretização (além de penalizar os valores altos, deixando-os maiores, e recompensar os valores baixos, deixando-os menores) é que isto nos permite escrever $E_C(\mathtt{X}) = \|\mathtt{C} \cdot \mathtt{X}\|^2$, em que $\mathtt{X}$ é o vetor da projeção avaliada e $\mathtt{C}$ é uma matriz. Chamaremos $\mathtt{C}$ de matriz de conformalidade. Note que temos $2 \cdot \mathtt{n} \cdot \mathtt{m}$ termos no somatório de E_C e que $\mathtt{X} \in \mathbb{R}^{2\cdot(\mathtt{n}+1)\cdot(\mathtt{m}+1)}$, assim devemos ter $\mathtt{C} \in \mathbb{R}^{2\cdot\mathtt{n}\cdot\mathtt{m}\times 2\cdot(\mathtt{n}+1)\cdot(\mathtt{m}+1)}$. A regra de indexação dos termos de $\mathtt{C}$ é apresentada a seguir.

$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+1, 2\cdot((j)+(i-1)\cdot(\mathtt{n}+1))+2) = \frac{1}{\Delta\lambda},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+1, 2\cdot((j-1)+(i)\cdot(\mathtt{n}+1))+1) = \frac{\cos(\mathtt{Dphi}(j))}{\Delta\phi},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+2, 2\cdot((j)+(i-1)\cdot(\mathtt{n}+1))+1) = \frac{1}{\Delta\lambda},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+2, 2\cdot((j-1)+(i)\cdot(\mathtt{n}+1))+2) = -\frac{\cos(\mathtt{Dphi}(j))}{\Delta\phi},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+1, 2\cdot((j-1)+(i-1)\cdot(\mathtt{n}+1))+2) = -\frac{1}{\Delta\lambda},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+1, 2\cdot((j-1)+(i-1)\cdot(\mathtt{n}+1))+1) = -\frac{\cos(\mathtt{Dphi}(j))}{\Delta\phi},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+2, 2\cdot((j-1)+(i-1)\cdot(\mathtt{n}+1))+1) = -\frac{1}{\Delta\lambda},$$
$$\mathtt{C}(2\cdot((j-1)+\mathtt{n}\cdot(i-1))+2, 2\cdot((j-1)+(i-1)\cdot(\mathtt{n}+1))+2) = \frac{\cos(\mathtt{Dphi}(j))}{\Delta\phi}.$$

Todos os outros elementos da matriz, são nulos. Vejamos um exemplo.

Exemplo 3.1.4. Com os valores como no exemplo 3.1.1, obtemos

$$\mathtt{C} = \begin{pmatrix} -\frac{\cos(-1)}{2} & \frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & \frac{\cos(-1)}{2} & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{\cos(1)}{2} & -\frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} & \frac{\cos(1)}{2} \\ 0 & 0 & 0 & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & 0 \\ \frac{\cos(-1)}{2} & 0 & 0 & 0 \\ 0 & -\frac{\cos(-1)}{2} & 0 & 0 \\ 0 & 0 & \frac{\cos(1)}{2} & 0 \\ 0 & 0 & 0 & -\frac{\cos(1)}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}^{T}_{12\times 4}.$$

Podemos observar com este exemplo, que as linhas da matriz `C` apresentam quatro elementos não nulos por linha.

Implementamos o código A.2.8 para discretização das equações de Cauchy-Riemann, aonde tomamos proveito da esparsidade da matriz para obter resultados para discretizações com mais pontos e de maneira mais rápida.

3.1.5 Troca de Posição de Pontos

Como vimos no Capítulo 1, existem algumas projeções conhecidas que satisfazem as equações de Cauchy-Riemann (estereográfica, Mercator, constante). Além disto, a proposição 2.5.1 mostra que qualquer combinação linear de projeções que satisfazem Cauchy-Riemann também satisfaz estas equações. Portanto, a minimização de E_C (equação 3.1.15) seria um problema mal posto, já que teria infinitas soluções. Para tornar o problema bem posto e explorar o espaço das projeções conformes, minimizamos algumas restrições de posição de uma projeção (o qual nos motivará a definir o erro de posição na sequência) sujeito a esta projeção pertencer ao espaço das projeções conformes.

Agora, nossa intenção é trocar a posição da imagem de uma projeção de um ponto da malha discretizada. Para tal, vamos precisar definir a matriz `P` de troca de posição de pontos. Desejamos que ao fazer o produto da matriz `P` pelo vetor `X` da imagem de uma projeção, obtenhamos as coordenadas do ponto que irá trocar de posição. Como a ideia

é a mesma para uma quantidade arbitrária de pontos, façamos nossa análise em cima de um problema de troca de k pontos. Suponhamos que desejamos trocar as posições dos pontos $(u_{i_1,j_1}, v_{i_1,j_1}), (u_{i_2,j_2}, v_{i_2,j_2}), \ldots, (u_{i_k,j_k}, v_{i_k,j_k})$, devemos então ter

$$\mathtt{P} \cdot \mathtt{X} = \begin{pmatrix} u_{i_1,j_1} & v_{i_1,j_1} & u_{i_2,j_2} & v_{i_2,j_2} & \cdots & u_{i_k,j_k} & v_{i_k,j_k} \end{pmatrix}^T_{2\cdot k\times 1}. \tag{3.1.16}$$

Portanto, devemos ter $\mathtt{P} \in \mathbb{R}^{2\cdot k\cdot \times 2\cdot n\cdot m}$. Então, para conseguirmos calcular `P` precisamos da seguinte variável de entrada em nosso código:

- `N`: vetor com os índices dos pontos que serão trocados.

Assim, podemos calcular `P` através da seguinte regra:

$$\mathtt{P}(2\cdot j-1, 2\cdot(\mathtt{n}+1)\cdot(\mathtt{N}(2\cdot j)-1)+2\cdot\mathtt{N}(2\cdot j-1)-1) = 1,$$
$$\mathtt{P}(2\cdot j, 2\cdot(\mathtt{n}+1)\cdot(\mathtt{N}(2\cdot j)-1)+2\cdot\mathtt{N}(2\cdot j-1)) = 1.$$

Todos os outros elementos da matriz, são nulos. Vejamos um exemplo para fixar as ideias.

Exemplo 3.1.5. Tomando a malha discretizada e o vetor da projeção perspectiva calculados nos exemplos 3.1.1 e 3.1.2 e tomando $\mathtt{N} = \begin{pmatrix} 1 & 2 & 3 & 1 \end{pmatrix}^T$, obtemos a seguinte variável de saída:

$$\mathtt{P} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

E obtemos assim

$$\mathtt{P} \cdot \mathtt{X} = \begin{pmatrix} u_{1,2} & v_{1,2} & u_{3,1} & v_{3,1} \end{pmatrix}^T_{4\times 1}.$$

Notemos que a matriz `P` tem apenas um elemento não nulo por linha e que a sequência dos pontos que iremos trocar posição faz diferença no resultado final.

Implementamos o código A.2.9 para o cálculo da matriz de troca de posição, também aproveitando sua esparsidade.

3.1.6 Erro de Posição e Erro Total

Já temos como quantificar o erro de conformalidade de uma projeção discretizada, agora restaria a nós definir o erro de curvatura nula. Em vez disso, vamos forçar as linhas a serem retas da seguinte forma: dada uma de nossas imagens produzidas, vamos observar as posições de uma linha curvada presente nela e realocá-los a fim de termos uma linha reta. Sendo `b` o vetor das novas posições dos pontos que trocarão de posição, desejamos ter $\|\mathtt{P}\cdot\mathtt{X} - \mathtt{b}\|^2 = 0$. Vejamos um exemplo.

Exemplo 3.1.6. Continuando o exemplo 3.1.5, se quiséssemos que os pontos $(u_{1,2}, v_{1,2})$ e $(u_{3,1}, v_{3,1})$ fossem movidos para os pontos $(1, 0)$ e $(0, -0.5)$, respectivamente, deveríamos ter

$$\|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|^2 = 0, \tag{3.1.17}$$

em que $b = \begin{pmatrix} 1 & 0 & 0 & -0.5 \end{pmatrix}^T_{4\times 1}$.

Assim, definimos o erro de posição:

$$E_P(\mathtt{X}) = \|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|^2. \tag{3.1.18}$$

Porém, lembramos que não queremos abrir mão da conformalidade para minimizar o erro de posição. Assim nossa abordagem ao problema foi resolver o problema de minimização

$$\begin{aligned} &\text{minimizar} \quad E_p(\mathtt{X}), \\ &\text{sujeito a} \quad E_C(\mathtt{X}) = 0. \end{aligned} \tag{3.1.19}$$

Para resolver esse problema, definimos a matriz de minimização $A_{\text{minimização}}$ e vetor de minimização $b_{\text{minimização}}$ como seguem.

$$A_{\text{minimização}} = \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot \mathtt{P} & \mathtt{C}^T \\ \mathtt{C} & 0 \end{pmatrix} \quad \text{e} \quad b_{\text{minimização}} = \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot b \\ 0 \end{pmatrix}. \tag{3.1.20}$$

Para tal, usamos as condições de Karush-Kuhn-Tucker (KKT). Indicamos (NOCEDAL; WRIGHT, 2006) para mais detalhes.

Seja um vetor $Z = \begin{pmatrix} \mathtt{X} \\ W \end{pmatrix}$ de dimensões compatíveis com a matriz e o vetor de minimização. Temos então

$$A_{\text{minimização}} \cdot Z - b_{\text{minimização}} = \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot \mathtt{P} & \mathtt{C}^T \\ \mathtt{C} & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathtt{X} \\ W \end{pmatrix} - \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot b \\ 0 \end{pmatrix} \tag{3.1.21}$$

$$= \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot \mathtt{P} \cdot \mathtt{X} + \mathtt{C}^T \cdot W \\ \mathtt{C} \cdot \mathtt{X} \end{pmatrix} - \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot b \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot \mathtt{P} \cdot \mathtt{X} - 2 \cdot \mathtt{P}^T \cdot b + \mathtt{C}^T \cdot W \\ \mathtt{C} \cdot \mathtt{X} \end{pmatrix} \tag{3.1.22}$$

$$= \begin{pmatrix} 2 \cdot \mathtt{P}^T \cdot (\mathtt{P} \cdot \mathtt{X} - b) + \mathtt{C}^T \cdot W \\ \mathtt{C} \cdot \mathtt{X} \end{pmatrix}. \tag{3.1.23}$$

Mostremos que se o vetor $Z = \begin{pmatrix} \mathtt{X} \\ W \end{pmatrix}$ é solução do sistema linear

$$A_{\text{minimização}} \cdot Z - b_{\text{minimização}} = 0, \tag{3.1.24}$$

então $\mathtt{X}$ é solução do problema 3.1.19. Seja um vetor S tal que $E_C(S) = 0$ (o que é equivalente à $\mathtt{C} \cdot S = 0$). Temos

$$\|\mathtt{P} \cdot S - \mathtt{b}\|^2 = \|\mathtt{P} \cdot (S - \mathtt{X}) + \mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|^2 = \langle \mathtt{P} \cdot (S - \mathtt{X}) + \mathtt{P} \cdot \mathtt{X} - \mathtt{b}, \mathtt{P} \cdot (S - \mathtt{X}) + \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \rangle \tag{3.1.25}$$

$$= \langle \mathtt{P} \cdot (S - \mathtt{X}), \mathtt{P} \cdot (S - \mathtt{X}) \rangle + 2 \cdot \langle \mathtt{P} \cdot (S - \mathtt{X}), \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \rangle + \langle \mathtt{P} \cdot \mathtt{X} - \mathtt{b}, \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \rangle \tag{3.1.26}$$

$$= \|\mathtt{P} \cdot (S - \mathtt{X})\|^2 + 2 \cdot \langle \mathtt{P} \cdot (S - \mathtt{X}), \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \rangle + \|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|^2. \tag{3.1.27}$$

Como Z é solução do sistema (3.1.24), segue que

$$2 \cdot \mathtt{P}^T \cdot (\mathtt{P} \cdot \mathtt{X} - b) = -\mathtt{C}^T \cdot W \quad \text{e} \quad \mathtt{C} \cdot \mathtt{X} = 0. \tag{3.1.28}$$

Assim, obtemos

$$2 \cdot \langle \mathtt{P} \cdot (S - \mathtt{X}), \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \rangle = 2 \cdot (\mathtt{P} \cdot (S - \mathtt{X}))^T \cdot (\mathtt{P} \cdot \mathtt{X} - \mathtt{b}) = (S - \mathtt{X})^T \cdot 2 \cdot \mathtt{P}^T \cdot (\mathtt{P} \cdot \mathtt{X} - \mathtt{b}) \tag{3.1.29}$$

$$= (S - \mathtt{X})^T \cdot (-\mathtt{C}^T \cdot W) = -(\mathtt{C} \cdot (S - \mathtt{X}))^T \cdot W = -(\mathtt{C} \cdot S - \mathtt{C} \cdot \mathtt{X})^T \cdot W = 0 \cdot W = 0. \tag{3.1.30}$$

Substituindo este último resultado na equação (3.1.27), temos

$$\|\mathtt{P} \cdot S - \mathtt{b}\|^2 = \|\mathtt{P} \cdot (S - \mathtt{X})\|^2 + \|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|^2 \geq \|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|^2, \tag{3.1.31}$$

e segue que $\mathtt{X}$ é um ponto de mínimo. Portanto, resolver o sistema linear 3.1.24 é equivalente a resolver o problema 3.1.19. Assim, podemos definir o erro total como

$$E(\mathtt{X}) = E_P(\mathtt{X}) + E_C(\mathtt{X}) = \|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|_2^2 + \|\mathtt{C} \cdot \mathtt{X}\|_2^2 = \tag{3.1.32}$$

$$= \left\| \begin{pmatrix} \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \\ \mathtt{C} \cdot \mathtt{X} \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \mathtt{P} \\ \mathtt{C} \end{pmatrix} \cdot \mathtt{X} - \begin{pmatrix} \mathtt{b} \\ 0 \end{pmatrix} \right\|_2^2 = \left\| A \cdot \mathtt{X} - \hat{b} \right\|_2^2, \tag{3.1.33}$$

em que $A = \begin{pmatrix} \mathtt{P} \\ \mathtt{C} \end{pmatrix}$ e $\hat{b} = \begin{pmatrix} \mathtt{b} \\ 0 \end{pmatrix}$. Isso era a tarefa desenvolvida pela primeira versão de nosso código, porém não estávamos obtendo resultados satisfatórios. Para contornar esse problema, nossa ideia foi adicionar um termo regularizador que faz com que a solução fique mais próxima de alguma projeção conforme. Para tal, introduzimos um número real α e um vetor de projeção conforme discretizada $\mathtt{Y}$ ao erro de posição, obtendo assim

$$\|\mathtt{P} \cdot \mathtt{X} - \mathtt{b}\|_2^2 + \|\alpha \cdot (\mathtt{X} - \mathtt{Y})\|_2^2 = \left\| \begin{pmatrix} \mathtt{P} \cdot \mathtt{X} - \mathtt{b} \\ \alpha \cdot (\mathtt{X} - \mathtt{Y}) \end{pmatrix} \right\|_2^2 = \left\| \begin{pmatrix} \mathtt{P} \\ \alpha \cdot I \end{pmatrix} \mathtt{X} - \begin{pmatrix} \mathtt{b} \\ \alpha \cdot \mathtt{Y} \end{pmatrix} \right\|_2^2 \tag{3.1.34}$$

$$= \left\| \tilde{P} \cdot \mathtt{X} - \tilde{b} \right\|_2^2, \tag{3.1.35}$$

em que I denota a matriz identidade. Para os nossos testes, fixamos $\alpha = 0.1$ e o vetor $\mathtt{Y}$ como sendo o vetor da discretização da projeção estereográfica. Tomando $\tilde{P}$ e $\tilde{b}$ nas matriz e vetor de minimização, obtemos uma solução $\mathtt{X}$ satisfatória ao resolver o problema 3.1.19. E por fim, definimos o novo erro total como sendo

$$\tilde{E}(\mathtt{X}) = \left\|\tilde{P}\cdot\mathtt{X} - \tilde{b}\right\|_2^2 + \|\mathtt{C}\cdot\mathtt{X}\|_2^2 = \left\|\begin{pmatrix} \tilde{\mathtt{P}}\cdot\mathtt{X} - \tilde{\mathtt{b}} \\ \mathtt{C}\cdot\mathtt{X}\end{pmatrix}\right\|_2^2 = \left\|\begin{pmatrix}\tilde{\mathtt{P}}\\ \mathtt{C}\end{pmatrix}\mathtt{X} - \begin{pmatrix}\tilde{\mathtt{b}}\\ 0\end{pmatrix}\right\|_2^2 \tag{3.1.36}$$

$$= \left\|\tilde{A}\cdot\mathtt{X} - B\right\|_2^2, \tag{3.1.37}$$

em que $\tilde{A} = \begin{pmatrix}\tilde{\mathtt{P}}\\ \mathtt{C}\end{pmatrix}$ e $B = \begin{pmatrix}\tilde{\mathtt{b}}\\ 0\end{pmatrix}$.

Implementamos o código A.2.12 para efetuar estas tarefas.

3.2 Resultados

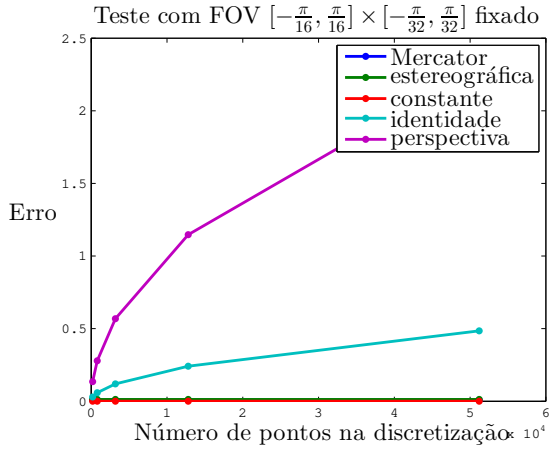
Nesta Seção apresentamos os resultados dos diversos testes que efetuamos com os nossos códigos. Dividimos em dois grupos estes testes: voltados ao erro de conformalidade e voltado ao erros envolvendo troca de posição de pontos.

Para realizar estes testes, usamos a linguagem de programação MATLAB R2011b, em um computador com um processador Intel© Core™ i7-4510U CPU 2.00GHz 2.60 GHz e memória RAM instalada de 8,00 GB.

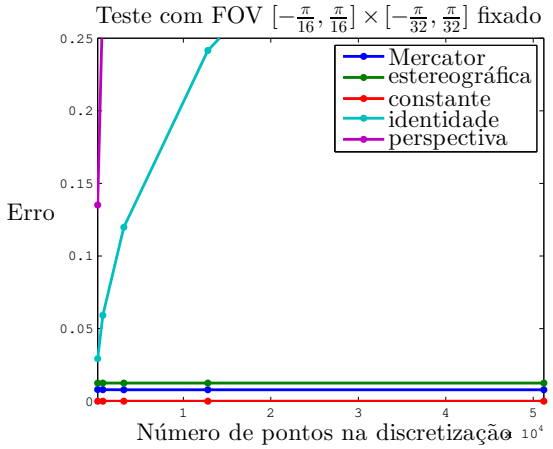
3.2.1 Erros de Conformalidade

Pra realizar estes testes, usamos os códigos A.2.10 e A.2.11. Ao fixar o FOV, estes códigos refinam uma malha discretizada em cada uma das 5 iterações, além de calcular as discretizações de cada uma das projeções apresentadas na Seção 1.3. Realizamos cada uma das iterações com uma malha de $2^{(i-1)}\cdot 10 \times 2^{(i-1)}\cdot 20$ pontos, em que i representa a i-ésima iteração. Os resultados dos testes são apresentados em forma de gráficos.

Teste 3.2.1. Neste teste fixamos o FOV $\left[-\frac{\pi}{16}, \frac{\pi}{16}\right] \times \left[-\frac{\pi}{32}, \frac{\pi}{32}\right]$. Note que este é um FOV baixo e centralizado no domínio equirretangular.



(a) Visualização geral.



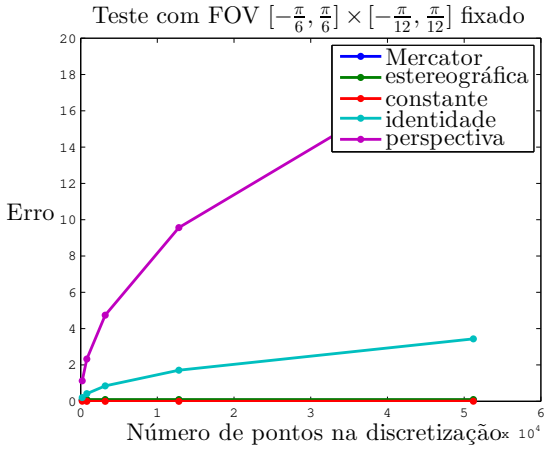
(b) Zoom.

Figura 3.2.1 – Gráfico das 5 iterações com FOV $\left[-\frac{\pi}{16}, \frac{\pi}{16}\right] \times \left[-\frac{\pi}{32}, \frac{\pi}{32}\right]$ fixado. Em (a) apresentamos uma visualização geral do gráfico e em (b) damos um zoom nos resultados das projeções conformes.

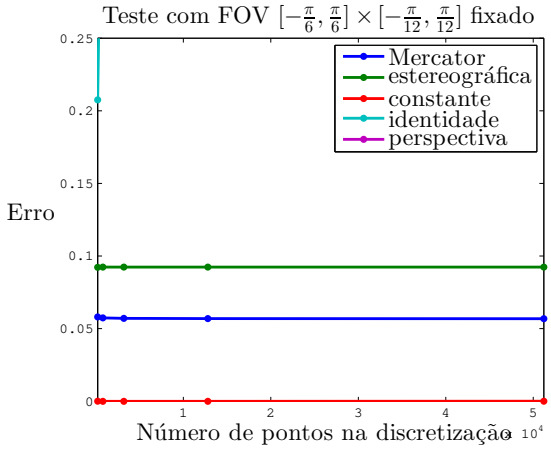
Com base na figura 3.2.1, observamos:

- Apesar de já sabermos que as projeções Mercator e estereográfica são conformes, seus erros não foram nulos. Como comentamos, isso era esperado por estarmos trabalhando com discretizações. Mas ao menos o erro ficou estabilizado para as discretizações dessas projeções.
- Podemos facilmente ver o comportamento divergente dos erros das projeções perspectiva e identidade (as quais vimos não serem conformes).
- O tempo de execução do teste foi de aproximadamente 75 segundos, ao total.

Teste 3.2.2. Teste com FOV $\left[-\frac{\pi}{6}, \frac{\pi}{6}\right] \times \left[-\frac{\pi}{12}, \frac{\pi}{12}\right]$ fixado. Observe este FOV também é centralizado, mas maior que o anterior. Ainda assim, não é o FOV de uma imagem panorâmica.



(a) Visualização geral.



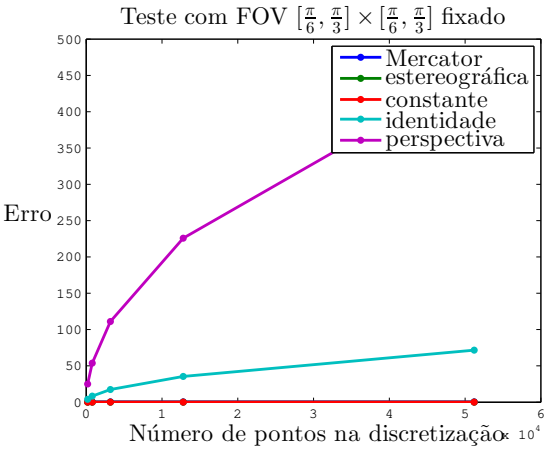
(b) Zoom.

Figura 3.2.2 – Gráfico das 5 iterações com FOV $\left[-\frac{\pi}{6},\frac{\pi}{6}\right]\times\left[-\frac{\pi}{12},\frac{\pi}{12}\right]$ fixado. Em (a) apresentamos uma visualização geral do gráfico e em (b) damos um zoom nos resultados das projeções conformes.

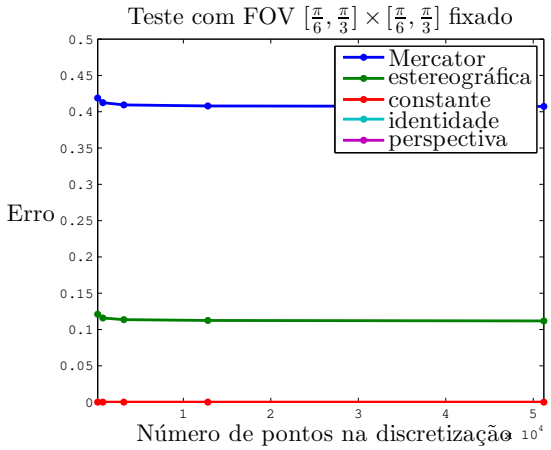
Com base na figura 3.2.2, observamos:

- Ao aumentarmos o FOV, os erros também aumentam, como era esperado. Porém as projeções conformes mantêm o comportamento estável, enquanto as projeções não conformes continuam divergindo.
- O tempo de execução do teste foi de aproximadamente 79 segundos, ao todo.

Teste 3.2.3. Teste com FOV $\left[\frac{\pi}{6},\frac{\pi}{3}\right]\times\left[\frac{\pi}{6},\frac{\pi}{3}\right]$ fixado. Aqui, descentralizamos o FOV, apesar de reduzi-lo.



(a) Visualização geral.



(b) Zoom.

Figura 3.2.3 – Gráfico das 5 iterações com FOV $\left[\frac{\pi}{6},\frac{\pi}{3}\right]\times\left[\frac{\pi}{6},\frac{\pi}{3}\right]$ fixado. Em (a) apresentamos uma visualização geral do gráfico e em (b) damos um zoom nos resultados das projeções conformes.

Com base na figura 3.2.3, observamos:

- Ao tirar o centro do FOV da origem, os erros mantêm o padrão que vinham apresentando. Ressaltamos que neste teste, a projeção estereográfica teve resultados melhores que a Mercator.
- O tempo de execução do teste foi de aproximadamente 75 segundos, ao total.

Teste 3.2.4. Teste com FOV $[-2.5, 2.5]\times[-1.2, 1.2]$ fixado (aproximadamente $\left[-\frac{4\cdot\pi}{5}, \frac{4\cdot\pi}{5}\right]\times\left[-\frac{2\cdot\pi}{5}, \frac{2\cdot\pi}{5}\right]$) . Note que agora temos um FOV de imagem panorâmica.

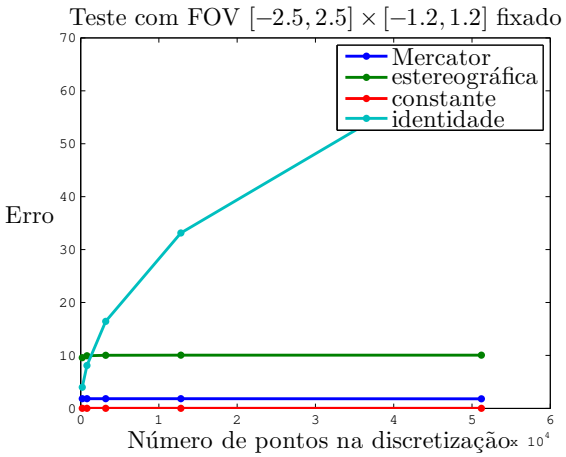


Figura 3.2.4 – Gráfico das 5 iterações com FOV $[-2.5, 2.5] \times [-1.2, 1.2]$ fixado.

Com base na figura 3.2.4, observamos:

- Retiramos a perspectiva deste teste, por termos um FOV alto, para o qual a projeção perspectiva não está definida.
- Tendo um FOV de uma imagem panorâmica, os erros ainda mantêm o padrão apresentado até aqui.
- O tempo de execução do teste foi de aproximadamente 61 segundos, ao todo.

Teste 3.2.5. Teste com FOV $[-3, 3] \times [-1.5, 1.5]$ fixado. Observe como o FOV agora é de quase todo o domínio equirretangular.

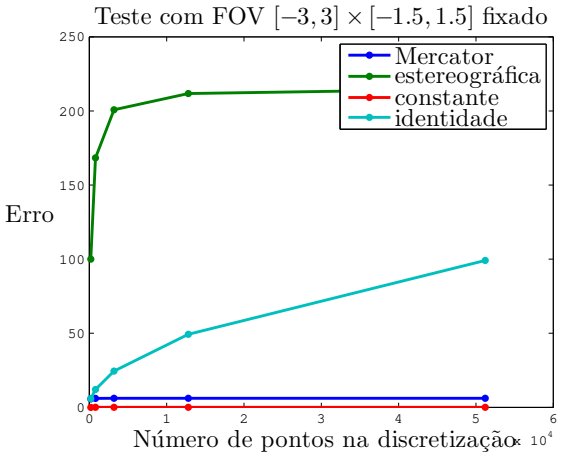


Figura 3.2.5 – Gráfico das 5 iterações com FOV $[-3,3] \times [-1.5,1.5]$ fixado.

Com base na figura 3.2.5, observamos:

- Aproximando o FOV ao de uma imagem equirretangular, o erro da projeção estereográfica tem comportamento de uma projeção não conforme. Isto ocorre pela imagem da projeção estereográfica expandir numa proporção maior que a malha discretizada tomada.
- O tempo de execução do teste foi de aproximadamente 61 segundos, ao todo.

3.2.2 Erros de Posição

Pra realizar estes testes, usamos o código A.2.13. Este código calcula uma malha discretizada e calcula os erros definidos anteriormente, com o usuário indicando os índices e imagens dos pontos que serão trocados. São feitos testes com diferentes parâmetros. Os resultados dos testes são apresentados em forma de tabelas, os quais mostram o erro total (equação 3.1.37), erro de conformalidade (equação 3.1.15) e erro de posição (equação 3.1.35).

Teste 3.2.6. Neste teste, mudamos a posição de um ponto da borda da malha, o movendo para dentro da malha. Para tal, fixamos `lambda_min` = -1, `lambda_max` = 1, `phi_min` = -0.5, `phi_max` = 0.5, `N` = $\begin{pmatrix} 1 & 1 \end{pmatrix}^T$, `b` = $\begin{pmatrix} 0.5 & 0.25 \end{pmatrix}^T$, `alpha` = 0.1, `Y`: estereográfica. Variamos `m` e `n` como apresentado na tabela 1.

	`m`=19, `n`=39	`m`=39, `n`=79	`m`=49, `n`=99
erro total	0.1092	0.1090	0.1090
erro de conformalidade	$4.2550 \cdot 10^{-27}$	$8.6876 \cdot 10^{-26}$	$1.8205 \cdot 10^{-25}$
erro de posição	0.1092	0.1090	0.1090

Tabela 1 – Resultados calculados em menos de 1 segundo, separadamente.

Ao alterar a posição de um ponto da borda da imagem da malha discretizada para uma posição dentro da imagem, os resultados foram satisfatórios, por serem baixos.

Teste 3.2.7. Neste teste, mudamos a posição de um ponto da borda da malha, o movendo para fora da malha. Fixamos `lambda_min` = -1, `lambda_max` = 1, `phi_min` = -0.5, `phi_max` = 0.5, `N` = $\begin{pmatrix} 1 & 1 \end{pmatrix}^T$, `b` = $\begin{pmatrix} 1.5 & 0.75 \end{pmatrix}^T$, `alpha` = 0.1, `Y`: estereográfica. Novamente, vamos variar `m` e `n`, como apresentado na tabela 2.

	`m`=19, `n`=39	`m`=39, `n`=79	`m`=49, `n`=99
erro total	0.2926	0.2922	0.2921
erro de conformalidade	$4.3089 \cdot 10^{-27}$	$8.6440 \cdot 10^{-26}$	$1.7833 \cdot 10^{-27}$
erro de posição	0.2926	0.2922	0.2921

Tabela 2 – Resultados calculados em menos de 1 segundo, separadamente.

Ao alterar a posição de um ponto da borda da imagem da malha discretizada para uma posição fora da imagem, tivemos resultados um pouco piores, mas mesmo assim satisfatórios.

Teste 3.2.8. Neste teste, mudamos a posição de um ponto interior da malha, o movendo para dentro da malha. Fixamos `lambda_min` = -1, `lambda_max` = 1, `phi_min` = -0.5, `phi_max` = 0.5, `N` = $\begin{pmatrix} 20 & 10 \end{pmatrix}^T$, `b` = $\begin{pmatrix} 0.5 & 0.25 \end{pmatrix}^T$, `alpha` = 0.1, `Y`: estereográfica. Variamos `m` e `n` como apresentado na tabela 3.

	`m`=19, `n`=39	`m`=39, `n`=79	`m`=49, `n`=99
erro total	0.2512	1.0570	1.2845
erro de conformalidade	$4.4006 \cdot 10^{-27}$	$8.3469 \cdot 10^{-26}$	$1.7777 \cdot 10^{-25}$
erro de posição	0.2512	1.0570	1.2845

Tabela 3 – Resultados calculados em menos de 1 segundo, separadamente.

Alterando a posição de um ponto no interior da imagem da malha discretizada, os erros aumentam a medida que refinamos a malha.

Teste 3.2.9. Neste teste, mudamos a posição de pontos tanto da borda da malha, quanto do interior da malha, os movendo para dentro da malha. Fixamos `m`=49,`n`=99,`lambda_min` = -1, `lambda_max` = 1, `phi_min` = -0.5, `phi_max` = 0.5, `alpha` = 0.1, `Y`: estereográfica. Agora, vamos variar os vetores `N` e `b`. Tomando $\mathtt{N}_1 = \begin{pmatrix} 50 & 25 & 1 & 1 \end{pmatrix}^T$, $\mathtt{b}_1 = \begin{pmatrix} 0.5 & 0.25 & 0.8 & 0 \end{pmatrix}^T$, $\mathtt{N}_2 = \begin{pmatrix} 50 & 25 & 1 & 1 & 100 & 50 \end{pmatrix}^T$ e $\mathtt{b}_2 = \begin{pmatrix} 0.5 & 0.25 & 0.8 & 0 & 0 & -0.4 \end{pmatrix}^T$, como vemos na tabela 4.

	$\mathtt{N}_1, \mathtt{b}_1$	$\mathtt{N}_2, \mathtt{b}_2$
erro total	0.4385	0.4594
erro de conformalidade	$2.0326 \cdot 10^{-25}$	$2.0326 \cdot 10^{-25}$
erro de posição	0.4385	0.4594

Tabela 4 – Resultados calculados em menos de 1 segundo, separadamente.

Apesar de efetuar a troca de mais pontos, os erros se mantiveram em um nível aceitável.

Teste 3.2.10. Neste teste, mudamos as posições de 4 pontos da malha, os movendo para dentro da malha (em articular, o mesmo ponto). Fixamos `m`=49, `n`=99, $\mathtt{N} = \begin{pmatrix} 1 & 1 & 100 & 1 & 1 & 50 & 100 & 50 \end{pmatrix}^T$, $\mathtt{b} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}^T$, `alpha` $= 0.1$, `Y`: estereográfica. Variamos os FOVs como $F_1 = [-0.5, 2] \times [-1, 0]$, $F_2 = [-2, 2] \times [-1, 1]$ e $F_3 = [-3, 3] \times [-1.5, 1.5]$. Vemos o resultado na tabela 5.

	F_1	F_2	F_3
erro total	0.2815	0.5465	4.1012
erro de conformalidade	$5.8280 \cdot 10^{-25}$	$2.0578 \cdot 10^{-25}$	$4.7330 \cdot 10^{-25}$
erro de posição	0.2815	0.5465	4.1012

Tabela 5 – Resultados calculados em menos de 1 segundo, separadamente.

Aumentando o FOV, os erros também crescem, mas em uma escala muito maior para o erro de posição.

4 Conclusões e Trabalhos Futuros

O último Capítulo do nosso trabalho é dividido em duas Seções. Na primeira, apresentamos nossas conclusões finais obtidas com o desenvolvimento deste trabalho. No segundo, são apresentados nossas futuras pretensões com o este trabalho além de indicar os caminhos naturais a serem seguidos.

4.1 Conclusões

Vimos como modelar uma imagem panorâmica a partir de uma projeção da esfera para o plano. Dentre as projeções, as ditas conformes foram as escolhidas por nós para minimizar as distorções presentes nessas imagens. Definimos como calcular os erros de conformalidade e de posição de uma projeção discretizada, o que nos possibilitaram avaliar quais produziriam melhores resultados.

Os resultados obtidos nos testes voltados exclusivamente aos erro de conformalidade mostraram que com uma quantidade relativamente baixa de pontos na malha discretizada, é possível averiguarmos se (a partir de sua discretização) uma projeção é ou não conforme. Este fato é importante para tentarmos descobrir a dimensão do espaço das projeções conformes. Tendo esta resposta em mãos poderíamos procurar as projeções com maiores graus de liberdade. O valor de graus de liberdade de uma projeção é o quanto podemos mexer nas posições dos pontos de sua imagem sem ter uma perda visual.

Com isto, poderíamos ter resultados de teste de erro de posição melhores (como colocamos um peso na projeção estereográfica, acabamos perdendo a oportunidade de testar alguma talvez melhor para o teste). Isto nos possibilitaria corrigir linhas curvadas sem a perda de conformalidade na imagem.

4.2 Trabalhos Futuros

Como vimos, nossa ideia é a partir de uma imagem que respeite a condição de conformalidade, forçar que as linhas sejam retas (ou pelo menos, certas linhas). Nosso próximo objetivo seria produzir uma interface gráfica que o usuário pudesse selecionar o FOV, projeção utilizada e linhas que deseja retificar clicando na própria imagem e tivesse resultado em tempo real.

Com este mesmo intuito, também pretendemos aprimorar (SOUTO; SACHT; VELHO, 2017). Neste trabalho os autores trabalham com uma classe de projeções conformes restritas: as projeções de Möbius. Nosso objetivo é fazer o que fizeram, mas com

projeções conformes quaisquer. Como as projeções de Möbius tem apenas 3 graus de liberdade, se conseguirmos projeções com maior valor de graus de liberdade, poderíamos evitar problemas que os autores tiveram.

Um de nossos objetivos com o trabalho era que qualquer pessoa que quisesse, pudesse ter acesso a ele. Disponibilizamos ele e também os códigos na internet, mas ainda ficamos com a sensação de estar devendo algo. A linguagem de programação adotada, o MATLAB, não é livre, isto é, para usá-lo é preciso pagar uma mensalidade. Nós não pagamos por ela, pois a nossa universidade, Universidade Federal de Santa Catarina, adquiriu o programa e assim alunos e professores dela podem usá-lo sem custos. Por já estarmos acostumados a trabalhar com ele, optamos por usá-lo no decorrer do trabalho. Assim sendo, um objetivo é implementar os códigos em uma linguagem livre a fim de nossos códigos realmente estarem ao alcance de todos. A princípio, nosso objetivo é implementar os códigos em PYTHON (((2019), 2019)).

Por fim, desejamos apresentar nosso trabalho em eventos. Temos como meta, submeter o trabalho ao SIBGRAPI 2020 (`http://www.mat.puc-rio.br/sibgrapi2019/`).

APÊNDICE A – Códigos

Neste Apêndice, apresentamos os códigos citados no decorrer do trabalho (Seção A.2) além de explicar sua sintaxe de escrita (Seção A.1). Ideias por trás de suas implementações foram apresentadas no Capítulo 3.

A.1 Sintaxe dos Códigos

Tudo que vem escrito após um sinal de porcentagem (`%`) em nossos códigos é dito um comentário. Eles não produzem efeito nenhum na execução do programa, isto é, se retirados, não alterariam em nada seu desempenho. Colocamos eles como guias tanto para quando estamos implementando-os, quanto para entender seu propósito ao usarmos eles novamente (eficaz quando ficamos muito tempo sem mexer no mesmo código). Mas o mais importante para nós, é que sirvam para que o usuário consiga manipular os códigos sem dificuldades.

Outra ressalva importante é como estão escritos estes cometários. Como a versão do MATLAB usada por nós não suportava o uso de acentos, tivemos de usar gírias para melhor escrever estes comentários. Citamos algumas:

- **h para acentos**: em vários momentos a letra h é usada como um acento agudo na letra que a antecede. Por exemplo, escrevemos já e é como `jah` e `eh`.
- **comandos do LaTeX**: por diversas vezes usamos algum comando do LaTeX no meio de nossa escrita para abreviar algo. Por exemplo, para escrevermos $\mathbb{R}^n$ e $\in$ usamos `R^n` e `\in`, respectivamente.

Além disso, nossos códigos têm o padrão da figura A.1.1.

A.2 Códigos

Apresentamos os códigos que usamos para obter os resultados do trabalho. Lembramos que todos estes códigos encontram-se na página de nosso trabalho: `http://mtm.ufsc.br/~leo/TCC_Mateus/`.

```
%NOME Explicacao breve.
%   Explicacao.
%
% Exemplo de uso:
% ---------------
%
% Variaveis de entrada:
% ---------------------
%
% Variaveis de saida:
% -------------------
%

Corpo do codigo.
```

Figura A.1.1 – Padrão de escrita de comentários que os nossos códigos seguem.

Código A.2.1.

```
function [ Dlambda, Dphi ] = discretizacao( m, n, lambda_min,...
    lambda_max, phi_min, phi_max )
%DISCRETIZACAO Discretizacao de um subconjunto do dominio equirretangular.
%   Discretizacao do conjunto [lambda_min,lambda_max]x[phi_min,phi_max] com
%   (m+1)*(n+1) pontos, respeitando -pi<=lambda_min<lambda_max<=pi e
%   -pi/2<=phi_min<phi_max<=pi/2. O numero de pontos latitudinais eh dado
%   por m+1, jah o numero de pontos longitudinuais por n+1. Para manter a
%   proporcao 2:1, devemos ter n = 2*m+1. Para mais detalhes, ver secao
%   3.1.1 do trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
% Variaveis de entrada:
% ---------------------
%   m: numero de pontos latitudinais -1,
%   n: numero de pontos longitudinais -1,
%   lambda_min: menor valor longitudinal,
%   lambda_max: maior valor longitudinal,
%   phi_min: menor valor latitudinal,
%   phi_max: maior valor latitudinal.
%
% Variaveis de saida:
% -------------------
%   Dlambda: vetor de coordenadas longitudinais da malha,
%   Dphi: vetor de coordenadas latitudinais da malha.

% Calculando a vetor de discretizacao lambda
```

```
Dlambda = linspace(lambda_min,lambda_max,n+1); % Vetor \in R^(n+1)

% Calculando a vetor de discretizacao phi
Dphi = linspace(phi_min,phi_max,m+1); % Vetor \in R^(m+1)
end
```

Código A.2.2.

```
function [ X ] = perspectiva( Dlambda, Dphi )
%PERSPECTIVA Projecao perspectiva discretizada.
%   Calculo dos valores da projecao perspectiva nos pontos de uma malha
%   discretizada. Para um trabalho mais eficiente, tome Dlambda, Dphi com o
%   codigo DISCRETIZACAO. Para mais detalhes, ver secoes 1.3.1 e 3.1.2 do
%   trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = perspectiva(Dlambda,Dphi)
%
% Variaveis de entrada:
% ---------------------
%   Dlambda: vetor de coordenadas longitudinais da malha,
%   Dphi: vetor de coordenadas latitudinais da malha.
%
% Variaveis de saida:
% -------------------
%   X: vetor de valores das funcoes reais da projecao calculado nos pontos
%   da malha, de maneira sequencial.

sl = size(Dlambda,2);
sp = size(Dphi,2);

% Vetor de valores das funcoes reais da projecao
X = zeros([2*sp*sl 1]);

% Calculando o vetor X
for i = 1:sp;
    for j = 1:sl;
        X(2*j +2*sl*i-2*sl-1) = (tan(Dlambda(j)));
        X(2*j +2*sl*i-2*sl) = (tan(Dphi(i)))/(cos(Dlambda(j)));
    end
end
end
```

Código A.2.3.

```
function [ X ] = mercator( Dlambda, Dphi )
%MERCATOR Projecao Mercator discretizada.
%   Calculo dos valores da projecao Mercator nos pontos de uma malha
%   discretizada. Para um trabalho mais eficiente, tome Dlambda, Dphi com o
%   codigo DISCRETIZACAO. Para mais detalhes, ver secoes 1.3.2 e 3.1.2 do
%   trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = mercator(Dlambda,Dphi)
%
% Variaveis de entrada:
% ---------------------
%   Dlambda: vetor de coordenadas longitudinais da malha,
%   Dphi: vetor de coordenadas latitudinais da malha.
%
% Variaveis de saida:
% -------------------
%   X: vetor de valores das funcoes reais da projecao calculado nos pontos
%   da malha, de maneira sequencial.

sl = size(Dlambda,2);
sp = size(Dphi,2);

% Vetor de valores das funcoes reais da projecao
X = zeros([2*sp*sl 1]);

% Calculando o vetor X
for i = 1:sp;
    for j = 1:sl;
        X(2*j +2*sl*i-2*sl-1) = Dlambda(j);
        X(2*j +2*sl*i-2*sl) = log(sec(Dphi(i))+tan(Dphi(i)));
    end
end
end
```

Código A.2.4.

```
function [ X ] = estereografica( Dlambda, Dphi )
%ESTEREOGRAFICA Projecao estereografica discretizada.
%   Calculo dos valores da projecao estereografica nos pontos de uma malha
%   discretizada. Para um trabalho mais eficiente, tome Dlambda, Dphi com o
%   codigo DISCRETIZACAO. Para mais detalhes, ver secoes 1.3.3 e 3.1.2 do
```

```
%   trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = estereografica(Dlambda,Dphi)
%
% Variaveis de entrada:
% ---------------------
%   Dlambda: vetor de coordenadas longitudinais da malha,
%   Dphi: vetor de coordenadas latitudinais da malha.
%
% Variaveis de saida:
% -------------------
%   X: vetor de valores das funcoes reais da projecao calculado nos pontos
%   da malha, de maneira sequencial.

sl = size(Dlambda,2);
sp = size(Dphi,2);

% Vetor de valores das funcoes reais da projecao
X = zeros([2*sp*sl 1]);

% Calculando o vetor X
for i = 1:sp;
    for j = 1:sl;
        X(2*j +2*sl*i-2*sl-1) = (2*sin(Dlambda(j))*cos(Dphi(i)))/...
            (cos(Dlambda(j))*cos(Dphi(i))+1);
        X(2*j +2*sl*i-2*sl) = (2*sin(Dphi(i)))/...
            (cos(Dlambda(j))*cos(Dphi(i))+1);
    end
end
end
```

Código A.2.5.

```
function [ X ] = identidade( Dlambda, Dphi )
%IDENTIDADE Projecao identidade discretizada.
%   Calculo dos valores da projecao identidade nos pontos de uma malha
%   discretizada. Para um trabalho mais eficiente, tome Dlambda, Dphi com o
%   codigo DISCRETIZACAO. Para mais detalhes, ver secoes 1.3.4 e 3.1.2 do
%   trabalho escrito.
%
% Exemplo de uso:
% ---------------
```

```
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = identidade(Dlambda,Dphi)
%
% Variaveis de entrada:
% ---------------------
%   DDlambda: vetor de coordenadas longitudinais da malha,
%   Dphi: vetor de coordenadas latitudinais da malha.
%
% Variaveis de saida:
% -------------------
%   X: vetor de valores das funcoes reais da projecao calculado nos pontos
%   da malha, de maneira sequencial.

sl = size(Dlambda,2);
sp = size(Dphi,2);

% Vetor de valores das funcoes reais da projecao
X = zeros([2*sp*sl 1]);

% Calculando o vetor X
for i = 1:sp;
    for j = 1:sl;
        X(2*j +2*sl*i-2*sl-1) = Dlambda(j);
        X(2*j +2*sl*i-2*sl) = Dphi(i);
    end
end
end
```

Código A.2.6.

```
function [ X ] = constante( Dlambda, Dphi )
%CONSTANTE Projecao constante discretizada.
%   Calculo dos valores de uma projecao constante nos pontos de uma malha
%   discretizada. Tomamos o ponto (2,1) como imagem da projecao. Para um
%   trabalho mais eficiente, tome Dlambda, Dphi com o codigo DISCRETIZACAO.
%   Para mais detalhes, ver secoes 1.3.5 e 3.1.2 do trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = constante(Dlambda,Dphi)
%
% Variaveis de entrada:
% ---------------------
```

```
%   Dlambda: vetor de coordenadas longitudinais da malha,
%   Dphi: vetor de coordenadas latitudinais da malha.
%
% Variaveis de saida:
% -------------------
%   X: vetor de valores das funcoes reais da projecao calculado nos pontos
%   da malha, de maneira sequencial.

sl = size(Dlambda,2);
sp = size(Dphi,2);

% Vetor de valores das funcoes reais da projecao
X = zeros([2*sp*sl 1]);

% Calculando o vetor X
for i = 1:sp;
    for j = 1:sl;
        X(2*j +2*sl*i-2*sl-1) = 2;
        X(2*j +2*sl*i-2*sl) = 1;
    end
end
end
```

Código A.2.7.

```
%GERACAOIMAGENS Imagem de uma projecao aplicada a uma malha discretizada.
%   Produz um resultado grafico das imagens dos pontos de uma malha
%   discretizada por uma projecao. Para um trabalho mais eficiente, tome X
%   com um dos codigos de projecoes discretizadas. Para mais detalhes, ver
%   secao 3.1.3 do trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   geracaoimagens

n = size(Dlambda,2)-1;
m = size(Dphi,2)-1;
I = imread('input.jpg');

% seleciona a porcao da imagem referente ao campo de visao
% [lambda_min,lambda_max] x [phi_min,phi_max]
I_crop = I(round((1+(pi/2-phi_max)/(pi/2)-1)*size(I,1)/2):...
           round((1-(pi/2+phi_min)/(pi/2)+1)*size(I,1)/2),...
           round((1+(pi+lambda_min)/pi-1)*size(I,2)/2):...
           round((1-(pi-lambda_max)/pi+1)*size(I,2)/2),:);
```

```
Xv = reshape(X(1:2:end),n+1,m+1);
Yv = reshape(X(2:2:end),n+1,m+1);

% tem que transpor
clear I_new
I_new(:,:,1) = I_crop(:,:,1)';
I_new(:,:,2) = I_crop(:,:,2)';
I_new(:,:,3) = I_crop(:,:,3)';

warp(Xv,Yv,zeros(size(Xv)),I_new);
view([0 0 1]);

%Versao de plot de pontos. Comentar da linha 17 a linha 32 e descomentar da
%linha 35 a linha 38 para usar esta versao.
% Xv = reshape(X(1:2:end),n+1,m+1);
% Yv = reshape(X(2:2:end),n+1,m+1);
% plot(Xv,Yv,'.')
```

Código A.2.8.

```
function [ C, erro ] = conformalidade( lambda_min, lambda_max, phi_min,...
    phi_max, Dlambda, Dphi, X )
%CONFORMALIDADE Calcula a matriz de conformalidade e o erro de
%conformalidade de uma projecao discretizada.
%   Calculo da matriz de conformalidade relacionada a uma malha
%   discretizada e, se especificada pelo usuario, calculo do erro de
%   conformalidade relacionado a uma projecao discretizada nessa malha.
%   Para um trabalho mais eficiente, tome Dlambda, Dphi com o codigo
%   DISCRETIZACAO e X com o codigo de alguma projecao discretizada. Para
%   mais detalhes, ver secoes 2.4 e 3.1.4 do trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = perspectiva(Dlambda,Dphi)
%
%   [C,erro] = conformalidade(-pi/3,pi/3,-pi/4,pi/5,Dlambda,Dphi,X)
%
% Variaveis de entrada:
% ---------------------
%   lambda_min: menor valor longitudinal,
%   lambda_max: maior valor longitudinal,
%   phi_min: menor valor latitudinal,
%   phi_max: maior valor latitudinal,
%   Dlambda: vetor de coordenadas longitudinais da malha,
```

```
%   Dphi: vetor de coordenadas latitudinais da malha.
%   X: vetor de valores das funcoes reais da projecao calculado nos pontos
%   da malha, de maneira sequencial.
%
% Variaveis de saida:
% -------------------
%   C: matriz de conformaldade,
%   erro: erro de conformalodade.

n = size(Dlambda,2)-1;
m = size(Dphi,2)-1;

C = sparse(2*m*n,2*(m+1)*(n+1)); % Matriz de conformalidade
deltaphi = (phi_max-phi_min)/m; % Economia de dados
valorlambda = n/(lambda_max-lambda_min); % Economia de dados

% Calculando a matriz C
for i = 1:m;
    valorphi = cos(Dphi(i))/deltaphi; % Economia de dados
    for j = 1:n;
        C(2*((j-1)+n*(i-1))+1,2*((j)+(i-1)*(n+1))+2) = valorlambda;
        C(2*((j-1)+n*(i-1))+1,2*((j-1)+(i)*(n+1))+1) = valorphi;
        C(2*((j-1)+n*(i-1))+2,2*((j)+(i-1)*(n+1))+1) = valorlambda;
        C(2*((j-1)+n*(i-1))+2,2*((j-1)+(i)*(n+1))+2) = -valorphi;
        C(2*((j-1)+n*(i-1))+1,2*((j-1)+(i-1)*(n+1))+2) = -valorlambda;
        C(2*((j-1)+n*(i-1))+1,2*((j-1)+(i-1)*(n+1))+1) = -valorphi;
        C(2*((j-1)+n*(i-1))+2,2*((j-1)+(i-1)*(n+1))+1) = -valorlambda;
        C(2*((j-1)+n*(i-1))+2,2*((j-1)+(i-1)*(n+1))+2) = valorphi;
    end
end

if nargout > 1 % Evitar contas desnecessarias
    erro = norm(C*X);
end
end
```

Código A.2.9.

```
function [ P ] = posicaoponto( n, m, N )
%POSICAOPONTO Calcula a matriz de troca de posicao de pontos.
%   Calculo da matriz P de troca de posicao de pontos da imagem de uma
%   projecao discretizada aplicada a uma malha discretizada. O usuario
%   especifica os pontos que serao trocados com o vetor N, constituido dos
%   indices dos pontos que serao trocados, de maneira sequencial. Para um
%   trabalho mais eficiente, tome o vetor X com um dos codigos de projecoes
%   discretizadas. Para mais detalhes, ver secao 3.1.5 do trabalho escrito.
```

```
%
% Exemplo de uso:
% ---------------
%   [Dlambda,Dphi] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   X = perspectiva(Dlambda,Dphi)
%
%   P = posicaoponto( X, [1,1] )
%
% Variaveis de entrada:
% ---------------------
%   m: numero de pontos latitudinais -1,
%   n: numero de pontos longitudinais -1,
%   N: vetor com os indices dos pontos que vao ser trocadas.
%
% Variaveis de saida:
% -------------------
%   P: matriz de troca de posicao de pontos.

p = size(N,1);
P = sparse(p,2*(n+1)*(m+1)); % Matriz de posicao

for j = 1:p/2;
    P(2*j-1,2*(n+1)*(N(2*j)-1)+2*N(2*j-1)-1) = 1;
    P(2*j,2*(n+1)*(N(2*j)-1)+2*N(2*j-1)) = 1;
end
end
```

Código A.2.10.

```
%ERROSCONF Calcula multiplos erros de conformalidade.
%   Calculo de multiplos erros de conformalidade de projecoes
%   discretizadas. Para mais detalhes, ver secao 3.2.1 do
%   trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   erросconf

% deve ser declarado a variavel i
m = 2^(i-1)*10-1; n = 2^(i-1)*20-1;
lambda_min = -3; lambda_max = 3; % Longitude
phi_min = -1.5; phi_max = 1.5; % Latitude

[Dlambda,Dphi] = discretizacao(m,n,lambda_min,lambda_max,phi_min,phi_max);
```

```
X = mercator(Dlambda,Dphi);
[~,erromer] = conformalidade(lambda_min,lambda_max,phi_min,phi_max,...
    Dlambda,Dphi,X);

X = estereografica(Dlambda,Dphi);
[~,erroest] = conformalidade(lambda_min,lambda_max,phi_min,phi_max,...
    Dlambda,Dphi,X);

X = constante(Dlambda,Dphi);
[~,errocon] = conformalidade(lambda_min,lambda_max,phi_min,phi_max,...
    Dlambda,Dphi,X);

X = identidade(Dlambda,Dphi);
[~,erroide] = conformalidade(lambda_min,lambda_max,phi_min,phi_max,...
    Dlambda,Dphi,X);

X = perspectiva(Dlambda,Dphi);
[~,erroper] = conformalidade(lambda_min,lambda_max,phi_min,phi_max,...
    Dlambda,Dphi,X);
```

Código A.2.11.

```
%GRAFERROSCONF Grafico de erros conformalidade.
%   Produz um resultado grafico de multiplos erros de conformalidade de
%   projecoes discretizadas em malhas discretizadas que sao refinadas
%   em cada iteracao. Para mais detalhes, ver secao 3.2.1 do
%   trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   graferrosconf

tic

P = zeros(5,1);
Ymer = zeros(5,1);
Yest = zeros(5,1);
Ycon = zeros(5,1);
Yide = zeros(5,1);
Yper = zeros(5,1);
for i=1:5
    erросconf
    P(i)=(m+1)*(n+1);
    Ymer(i) = erromer;
    Yest(i) = erroest;
    Ycon(i) = errocon;
```

```
    Yide(i) = erroide;
    Yper(i) = erroper;
end
figure
plot(P,Ymer,'.-',P,Yest,'.-',P,Ycon,'.-',P,Yide,'.-',P,Yper,'.-',...
    'LineWidth',2,'MarkerSize',15)
set(0,'defaulttextinterpreter','latex')
set(0,'DefaultTextFontname', 'latex')
set(0,'DefaultAxesFontName', 'latex')
title('Teste com FOV $X$ fixado')
xlabel('N\''{u}mero de pontos na discretiza\c{c}\~{a}o')
y=ylabel('Erro \phantom{rrrr}');
set(y,'Rotation',0);
legend({'Mercator','estereogr\''{a}fica','constante','identidade',...
    'perspectiva'},'Interpreter','latex','Location','northeast')
ht = findall(gcf,'type','text'); set(ht,'FontSize',17);

% zoom
%axis([200 51200 0 0.5]);

Tempo = toc;
```

Código A.2.12.

```
function [ X, errototal, erroconf, erropos ] = minconpos( C, P, b, m, n,...
    alpha, Y )
%MINCONPOS Calcula os erros total, de conformalidade e de posicao.
%   Calculo dos erros do problema de minimizacao a partir das variaveis de
%   entrada. Para um trabalho mais eficiente, tome C, P e Y com os codigos
%   CONFORMALIDADE, POSICAOPONTO e um dos codigos de projecao discretizada.
%   Para mais detalhes, ver secao 3.2.2 do trabalho escrito.
%
% Exemplo de uso:
% ---------------
%   [ Dlambda, Dphi ] = discretizacao(9,19,-pi/3,pi/3,-pi/4,pi/5)
%
%   Y = estereografica( Dlambda, Dphi )
%
%   N = [1;1]
%
%   b = [0;0]
%
%   alpha = 1e-1
%
%   C = conformalidade(-pi/3,pi/3,-pi/4,pi/5,Dlambda,Dphi,Y)
%
```

```
%   P = posicaoponto( 9,19,N)
%
%   [X,errototal,erroconf,erropos] = minconpos(C,P,b,9,19,alpha,Y)
%
% Variaveis de entrada:
% ---------------------
%   C: matriz de conformalidade,
%   P: matriz de troca de posicao de pontos,
%   b: vetor de nova posicoes de pontos,
%   m: numero de pontos latitudinais -1,
%   n: numero de pontos longitudinais -1,
%   alpha: termo regularizador,
%   y: projecao regularizadora.
%
% Variaveis de saida:
% -------------------
%   X: vetor resulatante da minimizacao,
%   errototal: erro total do problema de minimizacao,
%   erroconf: erro de conformalidade do problema de minimizacao,
%   erropos: erro de posicao do problema de minimizacao.

% adicionando regurlarizacao
P = [P; alpha*speye(size(C,2))];
b = [b; alpha*Y];

A = [2*(P'*P),C';C,sparse(2*m*n,2*m*n)];
b_alt = [2*P'*b;zeros(2*m*n,1)];
Z = A\b_alt;
X = Z(1:2*(m+1)*(n+1),1);

Aaumentada = [C;P];
baumentada = [zeros(size(C,1),1);b];
errototal = norm(Aaumentada*X-baumentada)^2;

erroconf = norm(C*X)^2;

erropos = norm(P*X-b)^2;
end
```

Código A.2.13.

```
%ERROSPOS Calcula e gera imagens com usando MINCONPOS.
%   Calcula e gera imagens usando o codigo MINCONPOS. Para mais detalhes,
%   ver secao 3.2.2 do trabalho escrito.
%
% Exemplo de uso:
```

```
% ---------------
%    errospos

tic
%TOL = 1e-1;
m = 49; n = 99;
lambda_min = -1; lambda_max = 1;
phi_min = -0.5; phi_max = 0.5;
N = [50;25;1;1;100;50]; b = [0.5;0.25;0.8;0;0;-0.4];
alpha = 1e-1;

[ Dlambda, Dphi ] = discretizacao( m, n, lambda_min, lambda_max,...
    phi_min, phi_max );
% Forcar conformalidade com uma projecao conforme
Y = estereografica( Dlambda, Dphi );
C = conformalidade( lambda_min, lambda_max, phi_min, phi_max,...
    Dlambda, Dphi, Y);
[ P ] = posicaoponto( m, n, N );
[ X, Normtotal, Normconf, Normpos ] = minconpos(C,P,b,m,n,alpha,Y);
figure % Abre automaticamente a figura e nao fecha alguma jah aberta
geracaoimagens
axis off % retira os numeros laterais
axis equal
Tempo = toc;
```

Referências

(2019), P. C. T. *Python: A dynamic, open source programming language*. [S.l.], 2019. Disponível em: <https://www.python.org/>.

BURDEN, R. L.; FAIRES, J. D. *Análise numérica*. [S.l.]: Cengage Learning, 2008.

GROSS, H. *Handbook of Optical Systems: Vol. 4 Survey of Optical Instruments*. [S.l.]: WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim, 2008.

LIMA, E. L. *Curso de Análise, Volume 2, 6ª edição*. [S.l.]: IMPA, 2000.

LIMA, E. L. *Curso de Análise, Volume 1, 12ª edição*. [S.l.]: IMPA, 2006.

MANFREDO, D. C. *Geometria Diferencial de Curvas e Superfícies*. [S.l.: s.n.], 1976.

MATLAB. *version R2011b*. Natick, Massachusetts: The MathWorks Inc., 2011.

NOCEDAL, J.; WRIGHT, S. *Numerical optimization*. [S.l.]: Springer Science & Business Media, 2006.

SACHT, L. K. Content-based projections for panoramic images and videos. *URL: http://w3.impa.br/~leo-ks/publications/thesis_leonardo_sacht_2010.pdf*, 2010.

SOUTO, L.; SACHT, L. K.; VELHO, L. Moebius transformations applied to omnidirectional images. *URL: https://www.visgraf.impa.br/Data/RefBib/PS_PDF/tr-02-2017/tr-02-2017.pdf*, 2017.

STRANG, G. *Linear algebra and its applications*. [S.l.]: Harcourt Brace Jodanovich San Diego, 1988.

ZORIN, D. N. Correction of geometric perceptual distortions in pictures. *URL: https://authors.library.caltech.edu/26887/3/95-22.pdf*, 1995.