UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

LAÍS BÁSSAME RODRIGUES

Abordagem não euclidiana e uso da métrica de Fisher para o processamento de imagens esféricas

Campinas

2016

Laís Bássame Rodrigues

Abordagem não euclidiana e uso da métrica de Fisher para o processamento de imagens esféricas

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutora em Matemática Aplicada.

Orientadora: Sueli Irene Rodrigues Costa

Este exemplar corresponde à versão final da Tese defendida pela aluna Laís Bássame Rodrigues e orientada pela Profa. Dra. Sueli Irene Rodrigues Costa.

Campinas

2016

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

R618a

Rodrigues, Laís Bássame, 1984-
Abordagem não euclidiana e uso da métrica de Fisher para o processamento de imagens esféricas / Laís Bássame Rodrigues. – Campinas, SP : [s.n.], 2016.

Orientador: Sueli Irene Rodrigues Costa.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Matriz de informação de Fisher. 2. Geometria hiperbólica. 3. Geometria da informação. 4. Processamento de imagens. 5. Morfologia matemática. I. Costa, Sueli Irene Rodrigues,1949-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Non-Euclidean approach and the Fisher metric use for spherical image processing
Palavras-chave em inglês:
Fisher information matrix
Hyperbolic geometry
Information geometry
Image processing
Mathematical morphology
Área de concentração: Matemática Aplicada
Titulação: Doutora em Matemática Aplicada
Banca examinadora:
Sueli Irene Rodrigues Costa [Orientador]
Leonardo Tomazeli Duarte
João Eloir Strapasson
Marcos Aurélio Batista
Charles Casimiro Cavalcante
Data de defesa: 24-06-2016
Programa de Pós-Graduação: Matemática Aplicada

Tese de Doutorado defendida em 24 de junho de 2016 e aprovada

Pela Banca Examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). SUELI IRENE RODRIGUES COSTA

Prof(a). Dr(a). LEONARDO TOMAZELI DUARTE

Prof(a). Dr(a). JOÃO ELOIR STRAPASSON

Prof(a). Dr(a). MARCOS AURÉLIO BATISTA

Prof(a). Dr(a). CHARLES CASIMIRO CAVALCANTE

A Ata da defesa com as respectivas assinaturas dos membros
encontra-se no processo de vida acadêmica do aluno.

Aos meus amores...

Agradecimentos

Agradeço primeiramente à Deus, à Nossa Senhora Aparecida e à minha avó Marta que, de onde estão, me guiam, me iluminam e me acompanham.

À minha mãe por todo carinho, atenção, cuidado e amor tão necessários nessa árdua batalha que é a vida.

Ao meu pai pelo exemplo de vida, pelo apoio, pelo encorajamento e pelo amor.

À minha irmã (e amiga) por todo amor, apoio, companheirismo e amizade e por trazer ainda mais alegria às nossas vidas com a chegada do meu afilhado João Miguel.

Ao meu irmão por me mostrar que a vida pode ser divertida e despreocupada (apesar de eu ainda não ter aprendido).

À minha madrinha pelo seu amor e carinho e por me ajudar a cuidar das minhas princesas Maria Eduarda, Mel e Dora nas minhas ausências.

À minha afilhada Maria Eduarda por entender minha ausência e sempre me esperar ansiosamente com um abraço muito amoroso.

Ao restante da minha família pelo apoio.

À minha amiga Aline pelo apoio e carinho.

À minha tia Solange por me dar colo quando eu muito precisei e por me ajudar a encontrar uma "porta" quando eu não via nem mesmo "janelas".

Ao meu amigo Fabiano pelas conversas esclarecedoras sobre algoritmos e sobre a vida.

À professora Ana Maria por me ajudar nos primeiros passos com o *Python*.

À minha orientadora Sueli por se aventurar nessa aplicação comigo e por me acompanhar e acreditar em mim desde a graduação.

Ao professor Edson Agustini por sempre orientar minha vida acadêmica desde a graduação, por acreditar em mim quando às vezes nem eu acreditava, por lutar comigo as batalhas diárias do mestrado e do doutorado. Talvez sem ele eu não teria uma tese para agradecer nesse momento.

"Uma mente que se abre a uma nova ideia nunca volta ao seu tamanho original."

(Albert Einstein)

Resumo

Este trabalho consiste no estudo de modelos matemáticos utilizando métricas não euclidianas que possam ser aplicados na obtenção, armazenamento e processamento de imagens de objetos esféricos. Trabalhamos com duas vertentes: (i) a utilização de espaços suporte circulares de pixels que permite um maior refinamento nas regiões próximas ao bordo da imagem circular que representa o objeto fotografado; (ii) a abordagem no contexto de Geometria da Informação onde a estrutura não euclidiana natural associada à métrica de Fisher em distribuições normais de probabilidade é considerada na geração de imagens capturadas por vários frames. A proposta de espaço suporte circular bifurca-se em dois modelos. Em ambos a modelagem é feita considerando a estrutura de pixels em coordenadas polares no plano cartesiano. Um deles é baseado na métrica do disco de Poincaré de raio R e curvatura K negativa. O outro foi concebido a partir do cômputo da área euclidiana de regiões sobre o objeto esférico representadas na região plana. Tais modelos reduzem em 37,2% a quantidade de pixels quando comparados com distribuições quadradas que representam o mesmo objeto. Em (ii) trabalhamos com modelos matemáticos de imagem que tratam tons de cinza como distribuições gaussianas representadas por pontos no semiplano de Poincaré. A morfologia matemática exige que sejam definidos ordenamentos entre tais pontos. Para isso, utilizamos a métrica de Fisher e consideramos ordenamentos conhecidos e dois novos mais específicos, propostos neste trabalho.

Palavras-chave: matriz de informação de Fisher, geometria hiperbólica, geometria da informação, processamento de imagens, morfologia matemática.

Abstract

This work is devoted to the study of mathematical models using non-Euclidean metrics that can be applied in the capture, storage and processing of images from spherical objects. We consider two approaches: (i) the use of circular support spaces of pixels which allows greater refinement in regions close to the edge of the circular image that represent the object photographed; (ii) the approach via Information Geometry where the natural non-Euclidean structure associated with the Fisher metric in the space of normal probability distribution is considered in the generation of images captured by several frames. The proposal of circular support space divides itself in two models. Both models consider the structure of pixels in polar coordinates in the Cartesian plane. One of them was based on the metric of Poincare disk of radius R and negative curvature K. The other was designed from the computation of the Euclidean area of the regions on the spherical object represented in the flat region. Such models reduce by 37.2% the amount of pixels when compared to squared distributions that represent the same object. In (ii) we work with mathematical models of images that deal with shades in a grayscale as Gaussian distributions represented by points in the Poincaré half-plane. Mathematical morphology requires an ordering between these points. For that, we have used the Fisher metric and considered known orderings and two new specific ones, proposed in this work.

Keywords: Fisher information matrix, hyperbolic geometry, information geometry, image processing, mathematical morphology.

Lista de ilustrações

Lista de símbolos

$\mathcal{H}^2$	Semiplano de Poincaré
ε	Erosão
δ	Dilatação
γ	Abertura
φ	Fechamento
Ω_T	Espaço suporte tradicional
Ω_H	Espaço suporte hiperbólico
Ω_E	Espaço suporte elíptico
$N_{\mathcal{G}}(v)$	Vizinhança de um ponto v em um grafo $\mathcal{G}$
$Q_{i,j}$	Pixels dos espaços Ω_H e Ω_E
MEC	Menor Círculo Envolvente
SE	Elemento Estrutural

Sumário

Introdução

Neste trabalho objetivamos o estudo de modelos matemáticos utilizando métricas não euclidianas que possam ser aplicados na obtenção, armazenamento e processamento de imagens de objetos esféricos. Há duas vertentes em nossos estudos:

(i) a utilização de um espaço suporte circular de pixels que permite um maior refinamento nas regiões próximas ao bordo da imagem circular que representa o objeto esférico fotografado e;

(ii) o aproveitamento natural da estrutura não euclidiana presente na geração de imagens capturadas por vários *frames* no semiplano de Poincaré, que conduz à área de Geometria da Informação e, em particular, à métrica de Fisher.

A motivação para os estudos da vertente (ii) vem dos artigos [7], [6], [5], e [8], nos quais encontramos também ideias básicas para processamento de imagens em variedades riemannianas quaisquer. Já, no que se refere à vertente (i) , embora sua ideia básica esteja presente em [2], [36], [35] e [56], a modelagem que propomos é nova.

Nossa proposta de espaço suporte circular se bifurca em dois modelos, que chamaremos simplificadamente de Ω_H e Ω_E. O espaço suporte Ω_H possui estrutura de pixels baseada na métrica hiperbólica do Disco de Poincaré de raio $R > 0$ e curvatura $K < 0$ arbitrários. Para tanto, fizemos a modelagem considerando a estrutura de pixels em coodenadas polares do plano cartesiano (ilimitado em todas as direções) deduzindo a estrutura métrica hiperbólica a partir das hipóteses consideradas na Seção 3.1. Já o espaço suporte Ω_E foi concebido a partir do problema do cômputo da área euclidiana de regiões sobre o objeto esférico tendo por base sua representação plana. Com isso, fizemos uma série de experimentações com projeções cilíndricas, cônicas e estereográficas para relacionar áreas de *pixels esféricos* com áreas de *pixels polares* no disco Ω_E. Chegamos ao teorema da Subseção 3.1.3 que fornece expressões analíticas para essa relação, tornando o disco Ω_E um espaço de curvatura positiva (elíptico).

Naturalmente, a distribuição de *pixels polares* em Ω_H ou Ω_E de tal modo que haja vantagens na captura, armazenamento e processamento de imagens muda consideravelmente quando comparado com o procedimento tradicional cartesiano. Tendo em vista a vertente (i) acima, propomos distribuições de pixels (Secão 3.1) que reduzem em $37,2\%$ a quantidade de pixels quando comparadas com distribuições cartesianas quadradas que representam o mesmo objeto esférico. Neste ponto temos um bom ganho em termos de armazenamento de bits da imagem. Entretanto, como não é comum a existência de câmeras

que possuem chips de captura de imagens em formato circular, para o desenvolvimento da vertente (i) em imagens circulares em Ω_H ou Ω_E, precisamos fazer uma amostragem adaptativa das imagens cartesianas para as polares, conforme propomos e detalhamos na Subseção 3.1.5.

Ainda em relação à vertente (ii), o processamento de imagens, no que diz respeito à morfologia matemática, submete ao estudo de vizinhanças de pixels em espaços suporte. É natural que quando trabalhamos com métricas diversas, a estrutura das vizinhanças mudem juntamente com suas aplicações. Na nossa proposta de modelo de espaço suporte tivemos que introduzir algumas adaptações para que os operadores morfológicos fossem viáveis em termos computacionais, conforme pode ser constatado na Subseção 3.1.4. Além disso, trabalhamos com modelos matemáticos de imagens que tratam tons de cinza como distribuições gaussianas representadas por pontos do semiplano de Poincaré. A morfologia matemática no modelo que propomos exige que sejam definidos ordenamentos no semiplano de Poincaré. Apresentamos, no Capítulo 4, o modelo matemático para imagens desenvolvido em [7] e propomos novos ordenamentos utilizando a estrutura métrica hiperbólica do semiplano de Poincaré.

A estrutura métrica hiperbólica presente na vertente (ii) é um ponto de destaque nesse tipo de trabalho. Conforme podemos ver em aplicações do Capítulo 5, imagens processadas, chamadas de imagens residuais, parecem ser, por si só, de especial interesse. Neste sentido, é importante ter em mente que tanto qualidade quanto aplicações de imagens processadas são aspectos um tanto subjetivos na área de imagens, conforme citado em [42]. Determinados tipos de imagens processadas com métricas não euclidianas podem revelar aplicações interessantes no futuro.

Para a implementação computacional e evolução das propostas acima apresentadas foi necessário o desenvolvimento de algoritmos próprios originais em linguagem *Python* que é a linguagem de programação mais utilizada em processamento de imagens. No Apêndice B apresentamos esses três algoritmos:

- Algoritmo para conversão de imagens do espaço suporte cartesiano para o espaço Ω_H;

- Algoritmo para conversão de imagens do espaço suporte cartesiano para o espaço Ω_E;

- Algoritmo para gerar imagens de acordo com a vertente (ii) e cálculo de erosão e abertura (operadores morfológicos) dessas imagens utilizando qualquer um dos sete ordenamentos estudados.

Em particular, no último algoritmo desenvolvemos uma técnica própria para determinação do *Menor Círculo Envolvente* (MEC) que é necessário para os três últimos ordenamentos propostos.

Parte deste trabalho foi apresentado, sob a forma do artigo [50], no XXXIII Simpósio Brasileiro de Telecomunicações, onde o modelo de espaço suporte Ω_H foi particularmente introduzido.

A estrutura desta tese é detalhada a seguir:

Capítulo 1: apresentamos uma breve introdução de alguns conceitos de Estatística, Geometria Hiperbólica e Métrica de Fisher.

Capítulo 2: apresentamos uma breve introduçao de alguns conceitos de Processamento de Imagens, em particular, morfologia matemática de imagens.

Capítulo 3: apresentamos os modelos Ω_H e Ω_E, vizinhanças e quantização, além de exemplos envolvendo cálculo de áreas em objetos esféricos.

Capítulo 4: apresentamos o modelo matemático de imagem do artigo [7] juntamente com os ordenamentos que propomos.

Capítulo 5: apresentamos algumas aplicações concernentes à morfologia matemática, incluindo um caso particular de imagem capturada por vários *frames*.

Capítulo 6: apresentamos algumas conclusões e perspectivas futuras, como o estudo de Transformadas de Distâncias e Morfologia Matemática em Cores.

Apêndice A: apresentamos algumas etapas do desenvolvimento de Ω_H e Ω_E.

Apêndice B: apresentamos os três algoritmos citados acima em linguagem *Python*.

1 Fundamentos de Geometria da Informação e de Geometria Hiperbólica

A Geometria da Informação é um ramo da teoria de probabilidade e estatística e surgiu do estudo da estrutura geométrica da variedade das distribuições de probabilidades. Iniciaremos o capítulo com algumas definições básicas de estatística para introduzirmos o conceito de distância entre distribuições de probabilidades. Posteriormente, faremos uma breve introdução sobre dois modelos de Poincaré para a geometria hiperbólica para, em seguida, relacionarmos a chamada métrica de Fisher, que induz uma noção de distância entre distribuições de probabilidades, com um dos modelos hiperbólicos.

Apresentamos, na subseção 1.2.1, um algoritmo original para o menor círculo envolvente e, na subseção 1.2.3, uma demonstração original da equação da métrica do disco de Poincaré de raio $R > 0$ e curvatura gaussiana $K < 0$ e uma fórmula para o cálculo da distância entre os pontos $(0, 0)$ e $(x, 0)$ deste modelo.

1.1 Conceitos de Estatística

Apresentamos inicialmente um breve resumo de alguns conceitos básicos de estatística a fim de introduzir o conceito de distância entre distribuições de probabilidade que utilizaremos no Capítulo 3. Citamos como referências para os conceitos de estatística [52], [62], [40] e [53] e para assuntos relacionados à Geometria da Informação [41], [17] e [44].

Um experimento é aleatório se seus resultados puderem variar a cada ensaio, mesmo que as condições de execução permaneçam iguais. A estatística se utiliza de experimentos aleatórios, pois sua regularidade torna possível construir um modelo matemático preciso com o qual se analisará o experimento.

Para cada experimento aleatório, definimos o espaço amostral S como o conjunto de todos os resultados possíveis desse experimento.

Dado um experimento aleatório e um espaço amostral S, uma função X que associa a cada elemento $s \in S$ um número real $X(s)$ é denominada uma variável aleatória (v.a.). O conjunto $\mathcal{X} = \{x : x = X(s), s \in S\} \subset \mathbb{R}$ é dito espaço de X.

(i) Se $\mathcal{X}$ é enumerável, então X é dita uma variável aleatória discreta;

(ii) Se $\mathcal{X}$ é não enumerável, então X é dita uma variável aleatória contínua.

Distribuição de probabilidades:

(i) Dada uma variável aleatória discreta X tal que $\mathcal{X} = \{x_1, x_2, x_3, ..\}$, definimos uma distribuição de probabilidades sobre X como sendo uma função[1] $p : \mathbb{R} \to \mathbb{R}$ tal que $p(a) = P\{X = a\}$ que satisfaz:

$$\begin{cases} p(x_i) \geqslant 0, & \forall x_i \in \mathcal{X} \\ p(x) = 0, & \forall x \notin \mathcal{X} \end{cases} \qquad \text{e} \qquad \sum_{i=1}^{+\infty} p(x_i) = 1.$$

(ii) Dada uma variável aleatória contínua X, uma distribuição de probabilidades sobre X é uma função $p : \mathbb{R} \to \mathbb{R}$

$$P\{X \in B\} = \int_B p(x)\, dx \qquad \forall B \subset \mathbb{R}$$

que satisfaz:

$$p(x) \geqslant 0, \quad \forall x \in \mathbb{R} \qquad \text{e} \qquad 1 = P\{X \in (-\infty, +\infty)\} = \int_{-\infty}^{+\infty} p(x)\, dx$$

e, caso $B = [a, b]$

$$P\{a \leqslant X \leqslant b\} = \int_a^b p(x)\, dx.$$

Dada uma v.a. X com distribuição de probabilidades $p(x)$, definimos a esperança matemática, ou valor esperado, ou ainda média de X por:

$$E(X) = \sum_{x \in \mathcal{X}} x p(x),$$

quando X é uma v.a. discreta e por

$$E(X) = \int_{-\infty}^{+\infty} x p(x)\, dx,$$

quando X é uma v.a. contínua.

De maneira geral, quando f é uma função que assume valores sobre X,

$$E(f(X)) = \sum_{x \in \mathcal{X}} f(x)\, p(x) \qquad \text{e} \qquad E(f(X)) = \int_{-\infty}^{+\infty} f(x)\, p(x)\, dx,$$

quando X é uma v.a. discreta e contínua, respectivamente.

Dada uma v.a. discreta X, a variância será dada por

$$V(X) = \sum_{x \in \mathcal{X}} (x - E(x))^2 p(x).$$

[1] A função p pode ser generalizada para $p : \mathbb{R}^n \to \mathbb{R}$.

Dada uma v.a. contínua X, a variância será dada por

$$V(X) = \int_{-\infty}^{+\infty} (x - E(X))^2 p(x)\, dx.$$

Dada uma v.a. X, o desvio padrão será dado por

$$DP(X) = \sqrt{V(X)}.$$

Exemplo: (Distribuição Normal Univariada) Dizemos que uma v.a. X tem distribuição normal com média μ, desvio padrão σ e variância σ^2, denotada por $N\left(\mu, \sigma^2\right)$, quando sua função de distribuição de probabilidades satisfaz

$$p(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right],$$

onde $\mu \in \mathbb{R}$ e $\sigma \in (0, +\infty)$.

Generalizaremos os conceitos acima para vetores aleatórios.

Um vetor aleatório $\mathbf{X}$ de dimensão n é uma aplicação que relaciona cada elemento do espaço amostral S com um ponto de $\mathbb{R}^n$. Dadas n variáveis aleatórias $X_1, ..., X_n$, então

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix}.$$

O conjunto $\mathcal{X} = \{x = (x_1, ..., x_n) : x_1 = X_1(s), ..., x_n = X_n(s), s \in S\} \subset \mathbb{R}^n$ é o espaço de $\mathbf{X}$.

Dado um vetor aleatório $\mathbf{X}$, uma distribuição de probabilidades sobre $\mathbf{X}$ é uma aplicação $p : \mathbb{R}^n \to \mathbb{R}$ tal que

$$\begin{cases} p(\mathbf{x}) \geqslant 0, & \forall \mathbf{x} \in \mathcal{X} \\ p(\mathbf{x}) = 0, & \forall \mathbf{x} \notin \mathcal{X} \end{cases} \qquad \text{e} \qquad \int_{\mathbb{R}^n} p(\mathbf{x})\, d\mathbf{x} = 1,$$

onde $d\mathbf{x}$ é o elemento volume dado por $d\mathbf{x} = dx_1 dx_2 \cdots dx_n$.

Dado um vetor aleatório $\mathbf{X} \in \mathbb{R}^n$, com distribuição de probabilidades p, a esperança e a variância de $\mathbf{X}$ são dados por

$$E(\mathbf{X}) = \begin{pmatrix} E(X_1) \\ E(X_2) \\ \vdots \\ E(X_n) \end{pmatrix} \qquad \text{e} \qquad V(\mathbf{X}) = \begin{pmatrix} V(X_1) \\ V(X_2) \\ \vdots \\ V(X_n) \end{pmatrix}.$$

De maneira geral, quando f é uma função que assume valores sobre $\mathbf{X}$, a esperança de $f(\mathbf{X})$ é definida por

$$E(f(\mathbf{X})) = \int_{\mathcal{X}} f(\mathbf{X})\, p(\mathbf{x})\, d\mathbf{x}.$$

Dado um vetor aleatório $\mathbf{X} \in \mathbb{R}^n$, a matriz de covariância de $\mathbf{X}$ é dada pela seguinte matriz de ordem n

$$\mathbf{\Sigma} = \begin{pmatrix} E\left((X_1 - E(X_1))^2\right) & \cdots & E((X_1 - E(X_1))(X_n - E(X_n))) \\ E((X_2 - E(X_2))(X_1 - E(X_1))) & \cdots & E((X_2 - E(X_2))(X_n - E(X_n))) \\ \vdots & \ddots & \vdots \\ E((X_n - E(X_n))(X_1 - E(X_1))) & \cdots & E\left((X_n - E(X_n))^2\right) \end{pmatrix}$$

Dada uma família Y de distribuições de probabilidades sobre uma v.a. X, suponhamos que cada distribuição de Y seja parametrizada por n variáveis reais $(\theta_1, \theta_2, ..., \theta_n)$, ou seja,

$$Y = \{p_\theta = p(x;\theta) : \theta = (\theta_1, ..., \theta_n) \in \Theta\},$$

onde $\Theta \subset \mathbb{R}^n$ é chamado espaço dos parâmetros e a aplicação $\theta \mapsto p_\theta$ é injetiva. Dizemos que Y é um modelo ou variedade estatística de dimensão n.

Assumiremos algumas condições de regularidade de um modelo estatístico Y a fim de dar continuidade à teoria. Consideraremos que $\Theta \subset \mathbb{R}^n$ é aberto e que $\theta \mapsto p_\theta$ é diferenciável para que derivadas como $\dfrac{\partial p(x;\theta)}{\partial \theta_i}$ e $\dfrac{\partial^2 p(x;\theta)}{\partial \theta_i \partial \theta_j}$ existam. Iremos supor ainda que as ordens de integração e diferenciação podem ser trocadas, assim

$$\int \frac{\partial p(x;\theta)}{\partial \theta i} dx = \frac{\partial}{\partial \theta_i} \int p(x;\theta)\, dx = \frac{\partial}{\partial \theta_i} 1 = 0,$$

pois $\displaystyle\int p(x;\theta) = 1$.

Dados $Y = \{p_\theta : \theta \in \Theta\}$ uma variedade estatística de dimensão n e um ponto $\theta \in \Theta$, a matriz de informação de Fisher de Y em θ de ordem n, denotada por $G(\theta) = [g_{ij}(\theta)]$ é tal que

$$g_{ij}(\theta) = \int \frac{\partial}{\partial \theta_i} (\ln p(x;\theta)) \frac{\partial}{\partial \theta_j} (\ln p(x;\theta))\, p(x;\theta)\, dx.$$

Quando $n = 1$ o escalar $G(\theta)$ é chamado de informação de Fisher.

O traço da matriz de informação de Fisher está relacionado com a área de um conjunto associado com uma dada distribuição de probabilidades, enquanto o volume deste conjunto está relacionado com a entropia [17]. A informação de Fisher é usada,

por exemplo, na delineação e planejamento de experimentos, em radares e na análise de imagens [41]. De maneira geral, se um problema envolve medida de dissimilaridade entre distribuições então é possível utilizar informação de Fisher em sua abordagem.

Existem modelos estatísticos onde a integral dada por $g_{ij}(\theta)$ diverge. Entretanto, assumiremos neste trabalho que $g_{ij}(\theta)$ é finito para todo θ e que $g_{ij} : \Theta \to \mathbb{R}$ é de classe C^∞ para todo i, j.

Dessa forma, a métrica dada pela matriz de informação de Fisher satisfaz

$$ds^2 = \sum_{i,j=1}^{n} g_{ij}(\theta)\, d\theta_i d\theta_j.$$

Como podemos ver em [44], a matriz $G(\theta)$ é simétrica semidefinida positiva.

Exemplo: Temos que a matriz de Fisher para uma distribuição normal univariada $N(\mu, \sigma^2)$ é dada por [44]

$$G(\theta) = \begin{pmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{2}{\sigma^2} \end{pmatrix}.$$

Portanto, a métrica é dada por

$$ds^2 = \frac{d\mu^2 + 2d\sigma^2}{\sigma^2}. \tag{1.1}$$

A seguir, veremos o conceito de distância entre distribuições de probabilidades proposto por Rao em 1945 [48] utilizando a métrica de Fisher.

Dada uma curva $\alpha(t) = (\alpha_1(t), ..., \alpha_n(t))$ ligando dois pontos θ_1 e θ_2 de Θ, suponha que $\alpha(t_1) = \theta_1$ e $\alpha(t_2) = \theta_2$. Assim, o comprimento de arco da curva α entre θ_1 e θ_2 [15] é dado por

$$l(\alpha) = \int_{t_1}^{t_2} \sqrt{\langle \alpha'(t), \alpha'(t) \rangle} dt = \int_{t_1}^{t_2} \left(\sum_{i,j=1}^{n} g_{ij}(\theta) \frac{d\alpha_i}{dt} \frac{d\alpha_j}{dt} \right)^{\frac{1}{2}} dt.$$

Sabemos [15] que a curva que minimiza esse comprimento de arco é chamada de geodésica e é dada pela solução do seguinte sistema de equações diferenciais:

$$\frac{d^2\alpha_k}{dt^2} + \sum_{i,j=1}^{n} \Gamma_{ij}^k \frac{d\alpha_i}{dt} \frac{d\alpha_j}{dt} = 0 \text{ com } k = 1, ..., n,$$

onde

$$\Gamma_{ij}^k = \frac{1}{2} \sum_l \left\{ \frac{\partial}{\partial \alpha_j} g_{il} + \frac{\partial}{\partial \alpha_i} g_{jl} - \frac{\partial}{\partial \alpha_l} g_{ij} \right\} g_{kl}$$

são os símbolos de Christofell da variedade.

A distância dada pela geodésica entre θ_1 e θ_2, chamada de distância de Fisher-Rao, é a distância entre distribuições de probabilidades parametrizadas por θ_1 e θ_2, proposta por Rao [48].

Por envolver soluções de equações diferenciais de segunda ordem, obter uma expressão geral para o cálculo da distância não é trivial. Entretanto, em alguns casos é possível relacionar a métrica do espaço de probabilidades com a métrica de um espaço conhecido. Esse será o assunto da Seção 1.3.

1.2 Conceitos de Geometria Hiperbólica

Resumimos nesta seção alguns conceitos fundamentais sobre a Geometria Hiperbólica Plana.

A geometria hiperbólica é uma geometria não euclidiana onde são válidos todos os axiomas de Hilbert exceto o axioma das paralelas: "Por um ponto fora de uma reta pode-se traçar uma única reta paralela à reta dada" que é substituído pelo axioma "Por um ponto fora de uma reta pode-se traçar mais de uma reta paralela à reta dada". Modelos para a geometria hiperbólica são representados em superfícies de curvatura gaussiana constante negativa. Descreveremos alguns aspectos da geometria hiperbólica em termos da geometria euclidiana para evitar a necessidade de discutir os axiomas.

Consideraremos, neste trabalho, dois modelos da geometria hiperbólica: o modelo do semiplano de Poincaré e o modelo do disco de Poincaré. Para isso, seja $\mathbb{C}$ o plano complexo com as notações usuais para as partes reais e imaginária de um ponto $z = \operatorname{Re}(z) + i \operatorname{Im}(z) \in \mathbb{C}$. Para simplificar a notação chamaremos $x = \operatorname{Re}(z)$ e $y = \operatorname{Im}(z)$. Além disso, utilizaremos a notação $\overline{z} = x - iy$ para denotar o conjugado de z. Como existe uma bijeção entre $\mathbb{C}$ e $\mathbb{R}^2$ poderemos também escrever $z = (x, y)$. Assim, se $z_1 = (x_1, y_1)$ e $z_2 = (x_2, y_2)$ então $d(z_1, z_2) = |z_2 - z_1| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

As demonstrações dos resultados apresentados nas Subseções 1.2.1 e 1.2.2 podem ser encontrados em [49].

1.2.1 O modelo do semiplano de Poincaré

No Capítulo 3 trabalharemos alguns ordenamentos de pontos do modelo do semiplano de Poincaré utilizados nas operações de erosão e dilatação de uma imagem. Vejamos alguns conceitos sobre este modelo.

O modelo euclidiano do semiplano de Poincaré, que denotaremos por $\mathcal{H}^2$, é a

parte superior do plano complexo $\mathcal{H}^2 = \{z \in \mathbb{C} \mid \mathrm{Im}\,(z) > 0\} = \mathbb{R} \times \mathbb{R}_+$ com a métrica

$$ds^2 = \frac{dx^2 + dy^2}{y^2}. \tag{1.2}$$

Sua fronteira é dada por $\partial\mathcal{H}^2 = \left\{(x, y) \in \mathbb{R}^2 \mid y = 0, x = \pm\infty, y = \infty\right\}$. Usaremos a notação $\overline{\mathcal{H}^2} = \mathcal{H}^2 \cup \partial\mathcal{H}^2$.

O semiplano de Poincaré, munido desta métrica, é uma variedade Riemanniana completa de curvatura gaussiana constante igual a -1.

Dado um caminho diferenciável por partes $\gamma : I \to \mathcal{H}^2$ onde $I = [0, 1]$ de tal forma que $\gamma(t) = x(t) + iy(t) \in \mathcal{H}^2$ com $t \in I$, então o comprimento hiperbólico $h(\gamma)$ é dado por

$$h(\gamma) = \int_0^1 \frac{\sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2}}{y(t)} dt.$$

A distância hiperbólica $\rho(z, w)$ entre dois pontos $z, w \in \mathcal{H}^2$ é dada por $\rho(z, w) = \min\{h(\gamma)\}$, onde o mínimo é tomado sobre todos os caminhos γ ligando z e w em $\mathcal{H}^2$. Temos que ρ é uma métrica em $\mathcal{H}^2$.

Agora introduziremos alguns conceitos que permitem obter uma fórmula explícita para ρ.

O grupo linear especial, denotado por $SL(2, \mathbb{R})$, é composto pelas matrizes reais $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ com $\det(M) = ad - bc = 1$, no qual a operação considerada é a multiplicação usual de matrizes.

O conjunto das transformações de Möbius (ou transformações fracionais lineares)

$$\left\{ \begin{array}{rccc} f: & \mathcal{H}^2 & \longrightarrow & \mathbb{C} \\ & z & \longmapsto & \dfrac{az + b}{cz + d} \end{array} \text{ tal que } a, b, c, d \in \mathbb{R} \text{ e } ad - bc = 1 \right\},$$

munido da operação usual de composição forma um grupo, denotado por $PSL(2, \mathbb{R})$, de tal modo que a composição de duas tranformações corresponde ao produto de duas matrizes de $SL(2, \mathbb{R})$ e a transformação inversa corresponde à matriz inversa.

Temos que $PSL(2, \mathbb{R})$ é isomorfo ao grupo quociente $SL(2, \mathbb{R}) / \{\pm I_2\}$ onde I_2 é a matriz identidade de ordem 2. Além disso, $PSL(2, \mathbb{R})$ contém todas as transformações de Möbius da forma $f(z) = \dfrac{az + b}{cz + d}$ com $a, b, c, d \in \mathbb{R}$ e $ad - bc > 0$.

Dada $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \in SL(2, \mathbb{R})$ associada a $f(z) = \dfrac{az + b}{cz + d} \in PSL(2, \mathbb{R})$,

definimos sua norma por

$$\|f\| = \|M\| = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

A distância induzida $d(f, g) = \|M - N\|$, com $N \in SL(2, \mathbb{R})$ associada a $g \in PSL(2, \mathbb{R})$, faz com que $PSL(2, \mathbb{R})$ seja um grupo topológico com topologia induzida pela norma acima. Um subgrupo de $PSL(2, \mathbb{R})$ é discreto quando a topologia induzida de $PSL(2, \mathbb{R})$ sobre tal subgrupo é discreta. Desta forma, um subgrupo discreto de isometrias de $\mathcal{H}^2$ corresponde a um conjunto discreto de pontos de $\mathbb{R}^4$. Um subgrupo discreto de $PSL(2, \mathbb{R})$ é chamado de fuchsiano [31].

Temos então os seguintes resultados:

- $PSL(2, \mathbb{R})$ age em $\mathcal{H}^2$ por homeomorfismos;

- Toda transformação de $PSL(2, \mathbb{R})$ é uma isometria.

Com esses resultados é possível provar que as geodésicas no modelo do semiplano são as semicircunferências e as semirretas ortogonais ao eixo real $\mathbb{R}$, bordo de $\mathcal{H}^2$ [49]. Como pode ser notado na Figura 1.1, dado um ponto z e uma geodésica γ existem mais de uma geodésica paralela (que não se intersecta) à γ passando por z, ou seja, a geometria em $\mathcal{H}^2$ não é euclidiana pois o axioma das paralelas não se aplica.

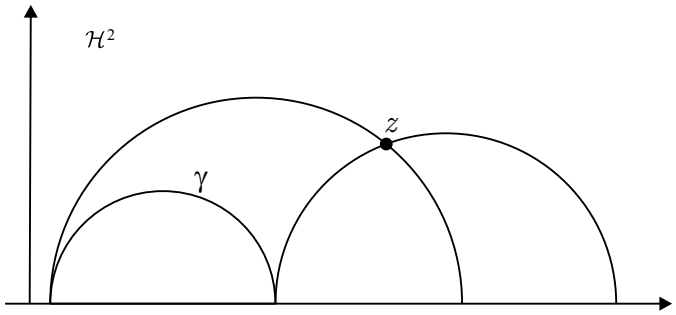


Figura 1.1: O Axioma das paralelas não vale na Geometria Hiperbólica.

A geometria euclidiana garante que dado dois pontos em $\mathcal{H}^2$, existe uma única semicircunferência ou semirreta em $\mathcal{H}^2$, ortogonal ao bordo $\mathbb{R}$ de $\mathcal{H}^2$, passando pelos pontos dados, o que condiz com o axioma da Geometria Hiperbólica que afirma que por dois pontos quaisquer de $\mathcal{H}^2$ passa uma única geodésica. Além disso, como era de se esperar, a distância entre dois pontos em $\mathcal{H}^2$ é o comprimento hiperbólico do único segmento geodésico que os une.

Colocamos, a seguir, algumas fórmulas explícitas para a distância hiperbólica em $\mathcal{H}^2$.

Para $z, w \in \mathcal{H}^2$:

$$(i)\ \rho(z,w) = \ln\left(\frac{|z-\overline{w}| + |z-w|}{|z-\overline{w}| - |z-w|}\right);$$

$$(ii)\ \cosh(\rho(z,w)) = 1 + \frac{|z-w|^2}{2\,\mathrm{Im}(z)\,\mathrm{Im}(w)};$$

$$(iii)\ \mathrm{senh}\left(\frac{1}{2}\rho(z,w)\right) = \frac{|z-w|}{2\sqrt{\mathrm{Im}(z)\,\mathrm{Im}(w)}};$$

$$(iv)\ \cosh\left(\frac{1}{2}\rho(z,w)\right) = \frac{|z-\overline{w}|}{2\sqrt{\mathrm{Im}(z)\,\mathrm{Im}(w)}};$$

$$(v)\ \tanh\left(\frac{1}{2}\rho(z,w)\right) = \left|\frac{z-w}{z-\overline{w}}\right|$$

onde $\overline{w}$ é o conjugado de w.

Circunferência Hiperbólica

O conceito de circunferência hiperbólica será importante para tratarmos sobre o problema do menor círculo envolvente (Minimum Enclosing Circle - MEC) apresentado a seguir.

Uma circunferência hiperbólica em $\mathcal{H}^2$ de centro $c_h = (a_h, b_h)$ e raio r_h é definida por $C_{r_h}(c_h) = \left\{z \in \mathcal{H}^2 : \rho(z, c_h) = r_h\right\}$. Vejamos que, geometricamente, $C_{r_h}(c_h)$ é igual à circunferência euclidiana de centro $c_e = (a_e, b_e) = (a_h, b_h\cosh(r_h))$ e raio $r_e = b_h\,\mathrm{senh}(r_h)$, ou seja,

$$C_{r_e}(c_e) = \left\{z = (x,y) \in \mathcal{H}^{\in} : (x-a_e)^2 + (y-b_e)^2 = r_e^2\right\}.$$

De fato:

- $C_{r_h}(c_h) \subset C_{r_e}(c_e)$: seja $z \in C_{r_h}(c_h) \Longrightarrow \rho(z, c_h) = r_h$. Daí,

$$\cosh(\rho(z,c_h)) = 1 + \frac{|z-c_h|^2}{2yb_h} = \frac{2yb_h + (x-a_h)^2 + (y-b_h)^2}{2yb_h} = \frac{(x-a_e)^2 + y^2 + b_h^2}{2yb_h} \Longrightarrow$$
$$(x-a_e)^2 = 2yb_h\cosh(r_h) - y^2 - b_h^2.$$

Logo,

$$\begin{aligned}(x-a_e)^2 + (y-b_e)^2 &= 2yb_h\cosh(r_h) - y^2 - b_h^2 + y^2 - 2yb_e + b_e^2\\ &= 2yb_h\cosh(r_h) - b_h^2 - 2yb_h\cosh(r_h) + b_h^2\cosh^2(r_h)\\ &= b_h^2\,\mathrm{senh}^2(r_h) = r_e^2.\end{aligned}$$

Donde $z \in C_{r_e}(c_e)$.

- $C_{r_e}(c_e) \subset C_{r_h}(c_h)$: seja $z \in C_{r_e}(c_e) \Longrightarrow (x-a_e)^2 + (y-b_e)^2 = r_e^2$. Daí,

$$\begin{aligned}\rho(z, c_h) &= \cosh^{-1}\left(1 + \frac{|z - c_h|^2}{2yb_h}\right) = \cosh^{-1}\left(\frac{(x - a_e)^2 + y^2 + b_h^2}{2yb_h}\right) \\ &= \cosh^{-1}\left(\frac{r_e^2 - (y - b_e)^2 + y^2 + b_h^2}{2yb_h}\right) \\ &= \cosh^{-1}\left(\frac{b_h^2 \operatorname{senh}^2(r_h) + 2yb_h \cosh(r_h) - b_h^2 \cosh^2(r_h) + b_h^2}{2yb_h}\right) \\ &= \cosh^{-1}(\cosh(r_h)) = r_h.\end{aligned}$$

Donde $z \in C_{r_h}(c_h)$.

As equações inversas $c_h = \left(a_e, \sqrt{b_e^2 - r_e^2}\right)$ e $r_h = \tanh^{-1}\left(\frac{r_e}{b_e}\right)$ permitem encontrar o centro e raio hiperbólico de uma circunferência dadas as medidas euclidianas. Como ilustrado na Figura 1.2, o centro hiperbólico c_h (em vermelho) da circunferência fica abaixo do centro euclidiano c_e (em azul).

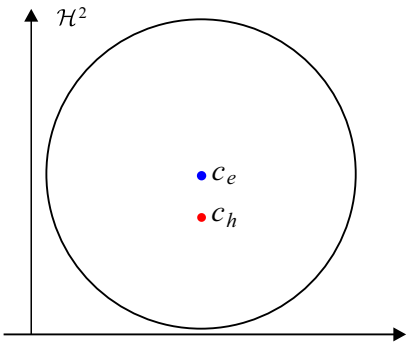


Figura 1.2: Circunferência com centros euclidiano c_e (azul) e hiperbólico c_h (vermelho).

Menor Círculo Envolvente - MEC

Um dos ordenamentos de pontos do semiplano que veremos adiante utiliza o chamado problema do menor círculo envolvente. Este problema é particularmente relevante, por exemplo, para a geometria da informação e consiste em, dados n pontos, encontrar o círculo de menor raio que contém todos os pontos. Este é um problema antigo e uma solução simples consiste em testar todos círculos que passam em cada 3 dos n pontos e verificar qual deles é o menor e contém todos os pontos. Entretanto, essa solução é bastante demorada computacionalmente. Outras soluções mais eficientes foram propostas ao longo dos anos. Em particular, o algoritmo de Bădoiu e Clarkson apresenta uma rápida e simples aproximação [7]. Esse problema tem aplicações práticas quando deseja-se por exemplo, construir um hospital que atenda um determinado número de comunidades. O centro do menor círculo envolvente seria um bom lugar para construir tal

hospital. Além disso, esse problema tem sido considerado em modelos hiperbólicos, como por exemplo, no modelo de Klein (um modelo hiperbólico que não consideramos neste trabalho), utilizando uma extensão riemanianna do algoritmo de Bădoiu e Clarkson que necessita da fórmula fechada da geodésica [7]. Neste trabalho consideraremos o conjunto de n pontos $A_1 = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$ no modelo do semiplano e visamos encontrar a geodésica (neste modelo) de menor raio euclidiano que englobe todos os pontos. Para isso, consideraremos a reflexão dos pontos de A_1 em torno do eixo x. Ou seja, consideraremos o conjunto de pontos $A_2 = \{(x_1, -y_1), (x_2, -y_2), ..., (x_n, -y_n)\}$ e encontraremos o MEC que englobe $A_1 \cup A_2$. Pela característica dos pontos sabemos que o centro do MEC estará no eixo x. Com o auxílio de parte do algoritmo de Welzl [68] *propomos um novo algoritmo*, que conjecturamos fornecer o MEC para os conjuntos específicos de pontos que utilizaremos.

Seja a a média das primeiras coordenadas dos pontos de A_1, ou seja, $a = (x_1 + ... + x_n)/n$. Seja $D = \{d((a, 0), (x_i, y_i)) \mid i = 1, .., n\}$, onde d denota a distância euclidiana e seja $m = \max D$. Seja $P = (x, y) \in A_1$ o ponto que determina m. Logo, a circunferência $C_m(a, 0)$ engloba todos os pontos do conjunto $A_1 \cup A_2$. Seja t a reta que passa por P e é perpendicular ao eixo x. Consideremos todos os pontos $P_1, ..., P_p$ de A_1 que estão à esquerda de t e $Q_1, ..., Q_q$ os pontos de A_1 à direita de t. Além disso, consideremos as circunferências $C_{r_i}(a_i, 0)$ que passam por P, $\overline{P} = (x, -y)$ e P_i; e as circunferências $C_{s_i}(b_i, 0)$ que passam por P, $\overline{P}$ e Q_i. Tomemos $\overline{a} = \min\{a_i \mid i = 1, ..., p\}$ e $\overline{b} = \max\{b_i \mid i = 1, .., q\}$. Assim, construiremos as circunferências $C_{R_1}(\overline{a}, 0)$, $C_{R_2}\left(\overline{b}, 0\right)$ e $C_y(x, 0)$ onde $R_1 = d((\overline{a}, 0), P)$ e $R_2 = d\left(\left(\overline{b}, 0\right), P\right)$. Em seguida, dentre as circunferências que englobarem todos os pontos, escolheremos a de menor raio. O exemplo a seguir ajuda a entender o procedimento delineado neste parágrafo.

Exemplo: A Figura 1.3 ilustra com um exemplo o algoritmo que estamos propondo.

O conjunto[2]

$$A_1 = \{(-5, 0.5), (2, 7), (2.1, 4.5), (-1, 7.5), (7.3, 9.4), (5.8, 3), (11, 2.2), (4.7, 2), (-3.6, 3.2), (0, 2.5)\}$$

contém os pontos representados em verde na figura. Os pontos em vermelho são os pontos do conjunto A_2 (refletidos). Os pontos em magenta são os centros das quatro circunferências computadas. A circunferência em preto é a circunferência inicial $C_m(a, 0)$, a circunferência em verde é a circunferência $C_{R_2}\left(\overline{b}, 0\right)$. A circunferência em azul é a circunferência $C_{R_1}(\overline{a}, 0)$ e a circunferência em vermelho é a $C_y(x, 0)$. Observamos que, neste caso, a circunferência vermelha não engloba todos os pontos e, por isso, não será considerada para decisão do MEC. Além disso, observamos que a circunferência em verde possui raio maior do que a circunferência em azul. Dessa forma, a circunferência em azul será tomada como o MEC.

[2] Utilizaremos a notação angloamericana para decimais

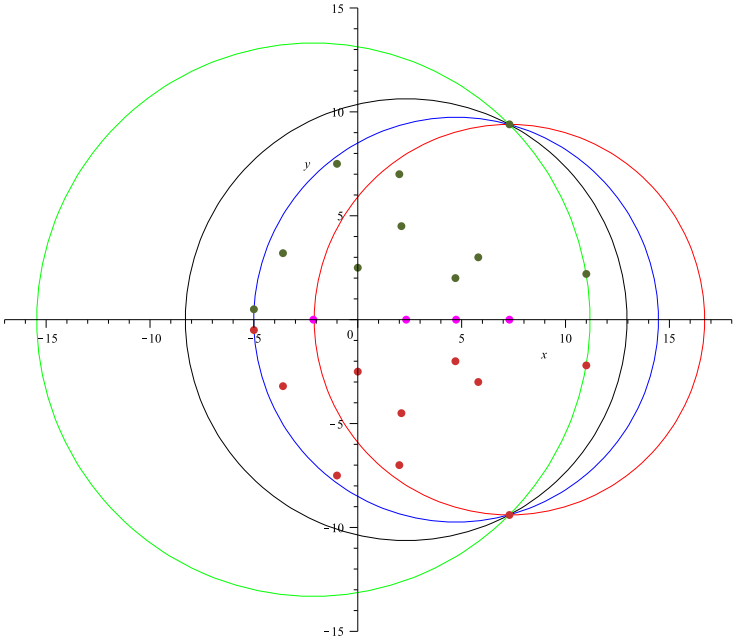


Figura 1.3: Exemplo do algoritmo MEC proposto.

Este algoritmo será utilizado para implementar ordenamentos propostos no Capítulo 3.

1.2.2 O modelo do disco de Poincaré

O assunto desta subseção será de grande importância para a definição de espaço suporte hiperbólico que apresentaremos no Capítulo 3.

O modelo euclidiano do disco de Poincaré, que chamaremos de $\mathcal{D}^2$, é o disco de raio 1 contido em $\mathbb{C}$, ou seja, $\mathcal{D}^2 = \{z \in \mathbb{C} \mid |z| < 1\}$ com a métrica

$$ds^2 = 4\frac{dx^2 + dy^2}{(1 - x^2 - y^2)^2}.$$

Sua fronteira é dada por $\partial\mathcal{D}^2 = \{z \in \mathbb{C} \mid |z| = 1\}$.

A aplicação

$$\begin{array}{rccc} \phi: & \mathcal{H}^2 & \longrightarrow & \mathcal{D}^2 \\ & z & \longmapsto & \dfrac{zi+1}{z+i} \end{array}$$

é uma isometria entre os modelos do semiplano e o modelo do disco.

Com a isometria ϕ acima podemos provar, a partir das geodésicas de $\mathcal{H}^2$, que as geodésicas de $\mathcal{D}^2$ são arcos de círculos ortogonais à fronteira $\partial\mathcal{D}^2$ ou seus diâmetros (Figura 1.4).

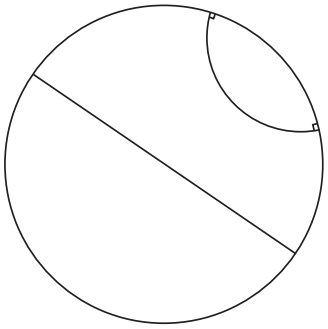


Figura 1.4: Geodésicas no modelo do Disco.

A seguir colocamos algumas fórmulas explícitas de distâncias hiperbólicas em $\mathcal{D}^2$.

Para $z, w \in \mathcal{D}^2$:

$(i)\ \rho^*(z,w) = \ln\left(\dfrac{|1-z\overline{w}| + |z-w|}{|1-z\overline{w}| - |z-w|}\right)$

$(ii)\ \cosh^2\left(\dfrac{1}{2}\rho^*(z,w)\right) = \dfrac{|1-z\overline{w}|^2}{\left(1-|z|^2\right)\left(1-|w|^2\right)}$

$(iii)\ \operatorname{senh}^2\left(\dfrac{1}{2}\rho^*(z,w)\right) = \dfrac{|z-w|^2}{\left(1-|z|^2\right)\left(1-|w|^2\right)}$

$(iv)\ \tanh\left(\dfrac{1}{2}\rho^*(z,w)\right) = \left|\dfrac{z-w}{1-z\overline{w}}\right|$

onde $\overline{w}$ é o conjugado de w.

As definições e resultados sobre grupos fuchsianos no modelo do disco são análogas ao modelo do semiplano.

O grupo linear especial sobre $\mathbb{C}$, denotado por $SL(2,\mathbb{C})$, é composto pelas matrizes complexas $M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ com $\det(M) = ad - bc = 1$, no qual a operação considerada é a multiplicação usual de matrizes. Para que $\dfrac{az+b}{cz+d}$ pertença a $\mathcal{D}^2$ é necessário que $b = \overline{c}$ e $d = \overline{a}$. Assim, temos a transformação de Möbius

$$\begin{array}{rccc} f: & \mathcal{D}^2 & \longrightarrow & \mathcal{D}^2 \\ & z & \longmapsto & \dfrac{az+\overline{c}}{cz+\overline{a}} \end{array} \tag{1.3}$$

com $a\overline{a} - c\overline{c} = 1$.

Dessa forma, o conjunto

$$A = \left\{ \begin{bmatrix} a & \overline{c} \\ c & \overline{a} \end{bmatrix} \in SL(2,\mathbb{C}) \right\} \subset SL(2,\mathbb{C})$$

é um subgrupo de $SL(2,\mathbb{C})$ se considerarmos a operação composição herdada de $SL(2,\mathbb{C})$. Além disso, o grupo das transformações (1.3), chamado de $PSL(2,\mathbb{C})$ é isomorfo ao grupo quociente $A/\{\pm I_2\}$ onde I_2 é a matriz identidade de ordem 2.

$$\text{Dado } M = \begin{bmatrix} a & \overline{c} \\ c & \overline{a} \end{bmatrix} \in A \text{ associada a } f(z) = \frac{az+\overline{c}}{cz+\overline{a}} \in PSL(2,\mathbb{C}) \text{ definimos}$$

$$\|f\| = \|M\| = \sqrt{|a|^2 + |\overline{c}|^2 + |c|^2 + |\overline{c}|^2} = \sqrt{2|a|^2 + 2|c|^2}.$$

Temos que $PSL(2,\mathbb{C})$ é um grupo topológico com a distância induzida $d(f,g) = \|M-N\|$, com $N \in A$ associada a $g \in PSL(2,\mathbb{C})$. A topologia, induzida pela norma definida acima, é também a topologia do $\mathbb{R}^4$. Um subgrupo de $PSL(2,\mathbb{C})$ é discreto quando a topologia induzida de $PSL(2,\mathbb{C})$ sobre tal subgrupo é discreta. Um subgrupo discreto de $PSL(2,\mathbb{C})$ é chamado de fuchsiano.

Temos os seguintes resultados:

- $PSL(2,\mathbb{C})$ age em $\mathcal{D}^2$ por homeomorfismos;

- Toda transformação de $PSL(2,\mathbb{C})$ é isometria.

A isometria entre os modelos também faz correlação entre os elementos de $PSL(2,\mathbb{R})$ e $PSL(2,\mathbb{C})$.

1.2.3 Representação do Plano Polar em um Disco

Esta subseção é importante para a proposta e desenvolvimento de um espaço suporte de pixels em formato de disco que será trabalhado ao longo desta tese. Apesar dos resultados desta seção serem relativamente bem conhecidos, as referências clássicas não trazem detalhamentos e demonstrações. O texto apresentado nesta subseção é um desenvolvimento próprio visando as aplicações no Capítulo 3.

Mostramos a seguir que se quisermos que um disco de raio R, radialmente homogêneo, com coordenadas polares, tenha uma métrica cujo comprimento dos raios seja infinito, então a curvatura gaussiana deve ser constante e negativa. Deduzimos também uma fórmula para o cálculo da distância entre os pontos $(0,0)$ e $(x,0)$ deste modelo.

Começamos com uma parametrização do plano polar dada por

$$X(r,\theta) = (r\cos(\theta), r\,\mathrm{sen}(\theta)) \text{ com } r,\theta \in \mathbb{R}.$$

Como estamos interessados em trabalhar com as ferramentas da Geometria Diferencial, façamos as restrições $r > 0$ e $\theta_0 - \pi < \theta < \theta_0 + \pi$, sendo θ_0 um real qualquer fixado. Com isso, a imagem de X é o plano $\mathbb{R}^2$ menos uma semirreta com origem em $(0,0)$. Essa

restrição não causa problemas pois, sendo θ_0 arbitrário, as propriedades geométricas que deduzirmos para as curvas coordenadas $\alpha_\theta : \mathbb{R}_+ \to \mathbb{R}^2$ dada por $\alpha_\theta(r) = X(r, \theta)$ valerão para a semirreta excluída da parametrização. A superfície parametrizada por X é regular e a chamaremos de *plano polar parametrizado* e indicaremos por P.

Nosso próximo passo é mapear, por meio de uma aplicação diferenciável $\sigma : P \to \sigma(P) \subset D_R$ (Figura 1.5), todo o plano polar parametrizado em um disco aberto de raio $R > 0$ com centro em $(0, 0)$, que estamos indicando por D_R, de tal modo que qualquer semirreta com origem em $(0, 0)$ seja mapeada em um segmento de comprimento R com origem em $(0, 0)$ sobre a própria semirreta. É como estivéssemos "encolhendo" a semirreta em um raio do disco. Além disso, precisamos que σ realize tal transformação agindo do mesmo modo para qualquer semirreta com origem em $(0, 0)$, em outras palavras, queremos que o disco seja radialmente homogêneo, assim como o plano polar, em termos métricos.

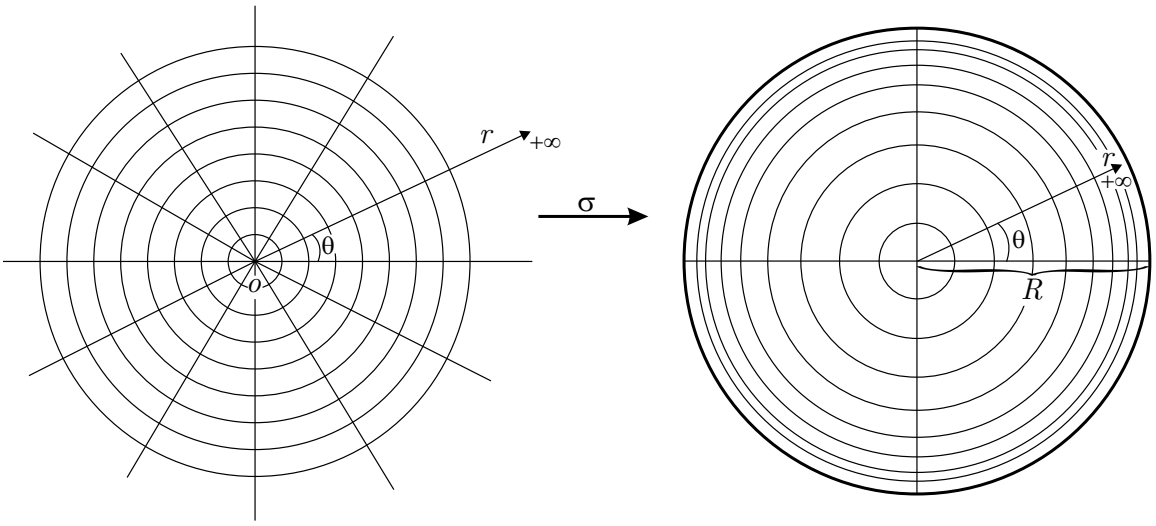


Figura 1.5: Aplicação σ.

Para nossos propósitos, em termos de imagens digitais, é importante que à medida que nos aproximemos do bordo do disco, tenhamos um número muito alto de pixels. Como cada pixel é identificado por um par ordenado de números inteiros (as coordenadas polares do centro do pixel), não podemos limitar superiormente o comprimento de cada raio de D_R. Eles devem ter comprimento infinito, o que significa, naturalmente, que não podemos utilizar a métrica euclidiana em cada raio do disco.

Até o momento temos várias imposições matemáticas para viabilizar o modelo de espaço de pixels que desejamos e a principal pergunta aqui é a seguinte: A aplicação σ existe? A resposta é sim, pelo Teorema de Hadamard [15], desde que a curvatura gaussiana de P seja não positiva. Neste caso, a aplicação σ seria uma restrição da aplicação exponencial de $T_p\mathbb{R}^2$ em D_R, sendo $p = (0, 0)$ (a aplicação exponencial, nessas condições, é um difeomorfismo).

De modo geral, sendo S superfície regular, a aplicação exponencial $\exp : T_pS \to S$ é definida como segue: dado o vetor $\vec{v}$ com origem em p no plano T_pS tangente a S em p, a imagem $\exp(\vec{v})$ é o extremo do segmento geodésico de S, de comprimento $|\vec{v}|$, que parte de $p \in S$, na direção e sentido de $\vec{v}$. Notemos que se exp for difeomorfismo, o comprimento de um raio geodésico de S (na métrica de S) tem que ser infinito. Além disso, é conveniente lembrar que dado p em S e $\vec{v}$ em T_pS, existe uma única geodésica de S passando por p na direção de $\vec{v}$, o que faz com que a aplicação exponencial esteja bem definida.

Desta forma, temos que trabalhar com a curvatura gaussiana do plano polar. A homogeneidade que estamos exigindo para a métrica em cada raio de D_R nos conduz naturalmente à curvatura gaussiana constante (espaços homogêneos em Geometria Diferencial são aqueles que possuem curvatura gaussiana constante). Entretanto, não podemos ter curvatura gaussiana nula em D_R pois, como já dissemos, não queremos limitar o comprimento dos raios do disco, o que nos leva a considerar a curvatura gaussiana do disco como sendo **constante e negativa**.

Por outro lado, queremos aproveitar a parametrização X definida acima para introduzirmos a métrica de curvatura constante negativa no disco, o que nos leva a considerar o plano polar com métrica não euclidiana (a euclidiana tem curvatura zero), ou seja, com métrica de curvatura constante negativa. Se conseguirmos uma métrica de curvatura negativa para o plano polar, a aplicação σ induzirá uma métrica de curvatura constante negativa no disco.

A proposição a seguir ([15]) garante que existe métrica de curvatura gaussiana constante e negativa para o plano polar parametrizado P.

Proposição: Sejam S superfície regular, $\mathcal{P}$ plano polar e $Y : \mathcal{P} \to S$ parametrização polar geodésica de S, ou seja, $Y(r,\theta) = \exp(r\cos(\theta), r\,\mathrm{sen}(\theta))$, sendo (r,θ) coordenadas polares no plano $\mathcal{P}$. Então, os coeficientes da primeira forma quadrática de S nesta parametrização são tais que:

$$E(r,\theta) = 1,\ F(r,\theta) = 0,\ \lim_{r\to 0}\sqrt{G(r,\theta)} = 0 \text{ e } \lim_{r\to 0}\frac{\partial}{\partial r}\sqrt{G(r,\theta)} = 1$$

onde $E(r,\theta) = \left\langle \frac{\partial Y}{\partial r}, \frac{\partial Y}{\partial r} \right\rangle (r,\theta)$; $F(r,\theta) = \left\langle \frac{\partial Y}{\partial r}, \frac{\partial Y}{\partial \theta} \right\rangle (r,\theta)$ e $G(r,\theta) = \left\langle \frac{\partial Y}{\partial \theta}, \frac{\partial Y}{\partial \theta} \right\rangle (r,\theta)$.

Este resultado é válido para S com qualquer curvatura, não necessariamente negativa, e vamos aplicá-lo fazendo $S = P$, plano polar parametrizado.

Vamos em busca do coeficiente $G(r,\theta)$ que nos interessa.

Dos textos de Geometria Diferencial [15] sabemos que a curvatura gaussiana K de uma superfície regular parametrizada é dada pela Equação de Gauss:

$$-EK = \left(\Gamma_{12}^2\right)_r - \left(\Gamma_{11}^2\right)_\theta + \Gamma_{12}^1\Gamma_{11}^2 - \Gamma_{11}^1\Gamma_{12}^2 + \left(\Gamma_{12}^2\right)^2 - \Gamma_{11}^2\Gamma_{22}^2$$

sendo Γ_{jk}^{i} símbolos de Christoffel da superfície parametrizada:

$$\Gamma_{11}^{1}=\frac{GE_r-2FF_r+FE_\theta}{2\left(EG-F^2\right)};\ \Gamma_{12}^{1}=\frac{GE_\theta-FG_r}{2\left(EG-F^2\right)};\ \Gamma_{22}^{1}=\frac{2GF_\theta-GG_r-FG_\theta}{2\left(EG-F^2\right)}$$
$$\Gamma_{11}^{2}=\frac{2EF_r-EE_\theta-FE_r}{2\left(EG-F^2\right)};\ \Gamma_{12}^{2}=\frac{EG_r-FE_\theta}{2\left(EG-F^2\right)};\ \Gamma_{22}^{2}=\frac{EG_\theta-2FF_\theta+FG_r}{2\left(EG-F^2\right)}$$

(a equação de Gauss e os símbolos de Christoffel dados acima é a base da prova do Teorema Egrégio de Gauss, que afirma que a curvatura gaussiana depende apenas da primeira forma quadrática da superfície).

No nosso caso, a parametrização Y é ortogonal, ou seja, $F\left(r,\theta\right)=0$, o que simplifica a Equação de Gauss:

$$-2EGK=G_{rr}+E_{\theta\theta}-\frac{G_r^2+E_\theta G_\theta}{2G}-\frac{E_\theta^2+E_rG_r}{2E}.$$

Ainda temos $E\left(r,\theta\right)=1$, portanto,

$$-2GK=G_{rr}-\frac{G_r^2}{2G}\Rightarrow K=-\frac{2GG_{rr}-G_r^2}{4G^2}\Rightarrow K=-\frac{2GG_{rr}-G_r^2}{4G^2}\Rightarrow K=-\frac{\frac{2GG_{rr}}{\sqrt{G}}-\frac{G_r^2}{\sqrt{G}}}{4G\sqrt{G}}\Rightarrow$$
$$K=-\frac{\frac{2GG_{rr}}{\sqrt{G}}-\frac{2G_r^2}{2\sqrt{G}}}{4G\sqrt{G}}\Rightarrow K=-\frac{\frac{G_{rr}2\sqrt{G}-\frac{G_r2G_r}{2\sqrt{G}}}{4G}}{\sqrt{G}}\Rightarrow K=\frac{\left(\frac{G_r}{2\sqrt{G}}\right)_r}{\sqrt{G}}\Rightarrow K=-\frac{\left(\sqrt{G}\right)_{rr}}{\sqrt{G}}.$$

A equação diferencial pode ser resolvida se K for constante negativa com as condições "iniciais" $\lim\limits_{r\to0}\sqrt{G}\left(r,\theta\right)=0$ e $\lim\limits_{r\to0}\dfrac{\partial}{\partial r}\sqrt{G\left(r,\theta\right)}=1$:

$$K=-\frac{\left(\sqrt{G}\right)_{rr}}{\sqrt{G}}\Rightarrow\left(\sqrt{G}\right)_{rr}+K\sqrt{G}=0\Rightarrow\sqrt{G}\left(r,\theta\right)=\frac{\operatorname{senh}\left(\sqrt{-K}r\right)}{\sqrt{-K}}\Rightarrow$$
$$\Rightarrow\boxed{G\left(r,\theta\right)=\frac{\operatorname{senh}^2\left(\sqrt{-K}r\right)}{-K}}$$

Observemos que, em coordenadas cartesianas, nossa conclusão é:

$$E\left(x,y\right)=1,\ F\left(x,y\right)=0\text{ e }G\left(x,y\right)=\frac{\operatorname{senh}^2\left(\sqrt{-K}\sqrt{x^2+y^2}\right)}{-K}.$$

Concluímos que o plano polar P pode ser munido de uma métrica riemanniana de curvatura gaussiana constante negativa K tal que

$$ds_P^2=E\left(r,\theta\right)dr^2+F\left(r,\theta\right)drd\theta+G\left(r,\theta\right)d\theta^2\Rightarrow ds_P^2=dr^2+\frac{\operatorname{senh}^2\left(\sqrt{-K}r\right)}{-K}d\theta^2$$

em coordenadas polares geodésicas.

Consideremos o plano polar P com a métrica de curvatura gaussiana constante negativa K com coeficientes da primeira forma quadrática em coordenadas polares geodésicas dados conforme acima. Retornemos à aplicação $\sigma:P\to D_R$.

Precisamos de uma função bijetiva diferenciável de uma variável $f : (0, R) \to (0, +\infty)$ que transforme o ponto (ρ, θ) de D_R no ponto $(r, \theta) = (f(\rho), \theta)$ de P (ambos os pontos em coordenadas polares) de tal modo que $\sigma^{-1}(\rho, \theta) = (f(\rho), \theta)$ (também em coordenadas polares). Logo, a métrica induzida por σ em D_R será de curvatura gaussiana constante K. Mais precisamente: sendo D_R parametrizado por coordenadas polares geodésicas $X(\rho, \theta) = (\rho\cos(\theta), \rho\,\mathrm{sen}(\theta))$, temos a métrica riemanniana

$$ds^2_{D_R} = E(\rho,\theta)\,d\rho^2 + F(\rho,\theta)\,d\rho d\theta + G(\rho,\theta)\,d\theta \Rightarrow ds^2_{D_R} = d\rho^2 + \rho^2 d\theta^2.$$

Como $r = f(\rho)$ temos $\dfrac{dr}{d\rho} = f'(\rho)$ e, portanto,

$$dr = f'(\rho)\,d\rho.$$

Por fim, a última imposição: queremos que a métrica induzida em D_R seja conforme (preserve ângulos euclidianos, o que significa $E = G$ e $F = 0$). Notemos que ds_P não é conforme, mas $F = 0$ significa que os raios geodésicos $\theta = \textit{constante}$ e os círculos geodésicos $r = \textit{constante}$ intersectam-se ortogonalmente.

Deste modo, a ação de σ em D_R é tal que

$$f'(\rho)\,ds_{D_R} = ds_P.$$

Pois

$$\begin{aligned}(f'(\rho))^2\,ds^2_{D_R} = ds^2_p &\iff (f'(\rho))^2\left(d\rho^2 + \rho^2 d\theta^2\right) = dr^2 + \frac{\mathrm{senh}^2\left(\sqrt{-K}r\right)}{-K}d\theta^2 \\ &\iff (f'(\rho)\,\rho)^2 = \frac{\mathrm{senh}^2\left(\sqrt{-K}f(\rho)\right)}{-K}\end{aligned}$$

e temos a EDO

$$\boxed{f'(\rho) = \frac{\mathrm{senh}\left(\sqrt{-K}f(\rho)\right)}{\sqrt{-K}\rho}}$$

que pode ser resolvida por separação de variáveis.

Façamos $y = f(\rho)$ e $x = \rho$. Como $r = f(\rho) > 0$ e $0 < \rho < R$ temos $y > 0$ e $0 < x < R$. Logo a EDO acima fica

$$\begin{aligned}\frac{dy}{dx} = \frac{\mathrm{senh}\left(\sqrt{-K}y\right)}{\sqrt{-K}x} \Rightarrow \int \frac{1}{\mathrm{senh}\left(\sqrt{-K}y\right)}dy = \int \frac{1}{\sqrt{-K}x}dx \Rightarrow \\ \int \frac{1}{2\,\mathrm{senh}\left(\frac{\sqrt{-K}y}{2}\right)\cosh\left(\frac{\sqrt{-K}y}{2}\right)}dy = \int \frac{1}{\sqrt{-K}x}dx \Rightarrow \\ \int \frac{1}{\frac{2\,\mathrm{senh}\left(\frac{\sqrt{-K}y}{2}\right)\cosh^2\left(\frac{\sqrt{-K}y}{2}\right)}{\cosh\left(\frac{\sqrt{-K}y}{2}\right)}}dy = \int \frac{1}{\sqrt{-K}x}dx \Rightarrow\end{aligned}$$

$$\int \frac{\operatorname{sech}^2\left(\frac{\sqrt{-K}y}{2}\right)}{2\operatorname{tgh}\left(\frac{\sqrt{-K}y}{2}\right)}dy = \int \frac{1}{\sqrt{-K}x}dx \Rightarrow \frac{\ln\left(\operatorname{tgh}\left(\frac{\sqrt{-K}y}{2}\right)\right)}{\sqrt{-K}} = \frac{\ln\left(\sqrt{-K}x\right)}{\sqrt{-K}} + c \Rightarrow$$

$$\ln\left(\operatorname{tgh}\left(\frac{\sqrt{-K}y}{2}\right)\right) = \ln\left(\sqrt{-K}x\right) + \sqrt{-K}c \Rightarrow \operatorname{tgh}\left(\frac{\sqrt{-K}y}{2}\right) = \left(\sqrt{-K}x\right)e^{\sqrt{-K}c}$$

A desigualdade $-1 < \left(\sqrt{-K}x\right)e^{\sqrt{-K}c} < 1$ deve ser satisfeita para que possamos inverter a tangente hiperbólica. Como $0 < x < R$, temos

$$\frac{1}{\sqrt{-K}e^{\sqrt{-K}c}} = R \Longleftrightarrow c = \frac{\ln\left(\frac{1}{\sqrt{-K}R}\right)}{\sqrt{-K}}$$

Fazendo $c = \dfrac{\ln\left(\frac{1}{\sqrt{-K}R}\right)}{\sqrt{-K}}$ temos

$$\operatorname{tgh}\left(\frac{\sqrt{-K}y}{2}\right) = \frac{x}{R} \Rightarrow \frac{\sqrt{-K}y}{2} = \operatorname{tgh}^{-1}\left(\frac{x}{R}\right) \Rightarrow y = \frac{2\operatorname{tgh}^{-1}\left(\frac{x}{R}\right)}{\sqrt{-K}}$$

Desta forma,

$$f(\rho) = \frac{2}{\sqrt{-K}}\operatorname{tgh}^{-1}\left(\frac{1}{R}\rho\right) \Rightarrow f'(\rho) = \frac{\frac{2}{R}}{\sqrt{-K}\left(1 - \frac{1}{R^2}\rho^2\right)}$$

Logo, a métrica procurada é

$$ds_H^2 = \left(f'(\rho)\right)^2 ds_{D_R}^2 = \left(\frac{\frac{2}{R}}{\sqrt{-K}\left(1 - \frac{1}{R^2}\rho^2\right)}\right)^2 \left(d\rho^2 + \rho^2 d\theta^2\right)$$

que, em coordenadas cartesianas, é

$$ds_H^2 = \frac{\frac{4}{R^2}\left(du^2 + dv^2\right)}{|K|\left(1 - \frac{1}{R^2}\left(u^2 + v^2\right)\right)^2}.$$

Conclusão: O disco D_R com a métrica riemanniana acima é um Disco de Poincaré e, portanto, é um modelo para a Geometria Hiperbólica Plana. Em particular, para $K = 1$ e $R = 1$ temos $D_1 = \mathcal{D}^2$ já apresentado na Subseção 1.2.2.

Seja $\alpha : [a, b] \to D_R$, $\alpha(t) = (u(t), v(t))$ curva em D_R. Sendo $l(\alpha)$ o comprimento de α na métrica riemanniana acima, temos

$$\begin{aligned} l(\alpha) &= \int_a^b \sqrt{I_{\alpha(t)}\left(\alpha'(t)\right)}dt \\ &= \int_a^b \sqrt{u'(t)^2 E(\alpha(t)) + 2u'(t)v'(t)F(\alpha(t)) + v'(t)^2 G(\alpha(t))}dt \end{aligned}$$

$$= \int_a^b \sqrt{u'(t)^2 \frac{\frac{4}{R^2}}{|K|\left(1-\frac{1}{R^2}\left(u(t)^2+v(t)^2\right)\right)^2} + v'(t)^2 \frac{\frac{4}{R^2}}{|K|\left(1-\frac{1}{R^2}\left(u(t)^2+v(t)^2\right)\right)^2}}dt$$

$$= \int_a^b \sqrt{\frac{4}{R^2}\frac{u'(t)^2+v'(t)^2}{|K|\left(1-\frac{1}{R^2}\left(u(t)^2+v(t)^2\right)\right)^2}}dt$$

$$= \int_a^b \frac{2\sqrt{u'(t)^2+v'(t)^2}}{R\sqrt{-K}\left(1-\frac{1}{R^2}\left(u(t)^2+v(t)^2\right)\right)}dt$$

Suponhamos que α seja a curva que liga os pontos $A = (0,0) = \alpha(a)$ e $B = (x,0) = \alpha(b)$, $0 < x < R$, de D_R. Na métrica ds_H^2 temos

$$l(\alpha) = \int_a^b \frac{2\sqrt{u'(t)^2+v'(t)^2}}{R\sqrt{-K}\left(1-\frac{1}{R^2}\left(u(t)^2+v(t)^2\right)\right)}dt \geqslant \int_a^b \frac{2|u'(t)|}{R\sqrt{-K}\left(1-\frac{1}{R^2}u(t)^2\right)}dt \geqslant$$

$$\geqslant \frac{1}{\sqrt{-K}}\int_a^b \frac{\frac{2u'(t)}{R}}{1-\left(\frac{u(t)}{R}\right)^2}dt = \frac{1}{\sqrt{-K}}\int_a^b \left(\frac{\frac{u'(t)}{R}}{1+\frac{u(t)}{R}} + \frac{\frac{u'(t)}{R}}{1-\frac{u(t)}{R}}\right)dt =$$

$$= \frac{1}{\sqrt{-K}}\left(\ln\left(1+\frac{u(t)}{R}\right) - \ln\left(1-\frac{u(t)}{R}\right)\right)\Bigg|_a^b$$

$$= \frac{1}{\sqrt{-K}}\left(\ln\left(\frac{1+\frac{u(t)}{R}}{1-\frac{u(t)}{R}}\right)\right)\Bigg|_a^b = \frac{1}{\sqrt{-K}}\ln\left(\frac{R+x}{R-x}\right)$$

Portanto, o menor comprimento possível para uma curva α (geodésica) ligando $A = (0,0)$ e $B = (x,0)$, $x > 0$, na métrica do disco é

$$l(\alpha) = \frac{1}{\sqrt{-K}}\ln\left(\frac{R+x}{R-x}\right).$$

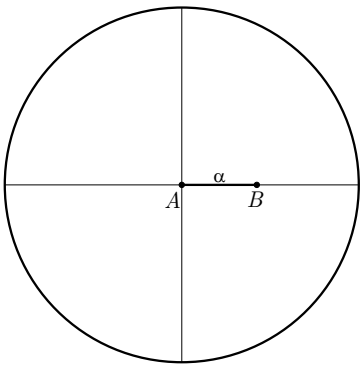


Figura 1.6: Curva que determina a distância entre A e B.

Ora, mas se α for o segmento de extremos A e B sobre o diâmetro horizontal de D_R (Figura 1.6), parametrizado por $\alpha(t) = (xt, 0)$ com $t \in [0,1]$, então α é uma curva minimizante, ou seja, segmento geodésico no disco com a métrica ds_H^2, pois as duas desigualdades no desenvolvimento acima tornam-se igualdades.

Conclusão: se $A = (0,0)$ e $B = (x,0)$ no disco de Poincaré de raio R com métrica de curvatura constante negativa K, dada por $ds_H^2 = \frac{\frac{4}{R^2}\left(du^2+dv^2\right)}{|K|\left(1-\frac{1}{R^2}\left(u^2+v^2\right)\right)^2}$, então

$$\boxed{\rho^*(A,B) = \frac{1}{\sqrt{-K}} \ln\left(\frac{R+x}{R-x}\right)}$$

Observação: Para nossos propósitos, a fórmula da distância deduzida acima é suficiente. Entretanto, fazendo uso das transformações de Möbius é possível continuar o trabalho acima e obter a fórmula para a distância entre dois pontos quaisquer do Disco de Poncaré de raio R e curvatura K, além do estudo sobre o formato euclidiano de suas geodésicas.

1.3 A métrica de Fisher

Agora que fizemos uma revisão dos conceitos de geometria hiperbólica podemos relacionar a distância entre distribuições normais e a distância do modelo do disco. Esta relação será muito utilizada no Capítulo 3.

Já vimos que uma distribuição normal univariada $N\left(\mu,\sigma^2\right)$ é dada pela distribuição

$$p(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right].$$

Além disso, um espaço paramétrico para essa família de distribuições é

$$\mathcal{H}_F = \left\{(\mu,\sigma) \in \mathbb{R}^2 \mid \sigma > 0\right\}$$

que coincide com $\mathcal{H}^2$. Veremos como a métrica dos dois espaços se relacionam.

Na Figura 1.7 (retirada de [17]) podemos ver uma comparação entre algumas distribuições. Observamos que não faremos distinção entre o ponto do plano (μ,σ) e a distribuição $N\left(\mu,\sigma^2\right)$.

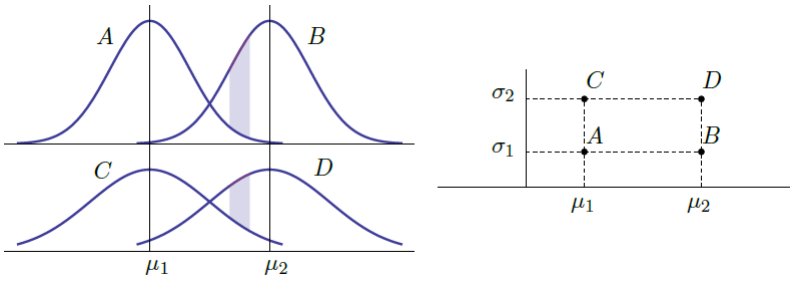


Figura 1.7: Comparação entre distribuições normais.

Conforme observado em [17], a distância entre dois pontos do espaço paramétrico deve condizer com a medida de dissimilaridade entre as duas distribuições correspondentes.

Dessa forma, claramente a distância entre os pontos de $\mathcal{H}_F$ não pode ser euclidiana e a métrica de Fisher para distribuições normais corrobora essa afirmação.

Consideremos a aplicação $T : \mathcal{H}_F \to \mathcal{H}^2$ tal que $T(\mu,\sigma) = \left(\frac{\mu}{\sqrt{2}},\sigma\right)$ e as métricas (1.1) e (1.2). Dada uma curva α no modelo $\mathcal{H}_F$ tal que $\alpha(t) = (\mu(t),\sigma(t))$, $t \in [a,b]$ e uma curva β no modelo $\mathcal{H}^2$ tal que $\beta(t) = T(\alpha(t)) = \left(\frac{\mu(t)}{\sqrt{2}},\sigma(t)\right)$, $t \in [a,b]$, temos:

$$\begin{aligned}\|\alpha\|_{H_F} &= \int_a^b \|\alpha'(t)\|_{H_F}\, dt = \int_a^b \sqrt{\mu'(t)^2 \frac{1}{\sigma'(t)^2} + 2\mu'(t)\sigma'(t).0 + \sigma'(t)^2 \frac{2}{\sigma'(t)^2}}dt \\ &= \int_a^b \frac{\sqrt{\mu'(t)^2 + 2\sigma'(t)^2}}{\sigma'(t)}dt.\end{aligned}$$

Além disso,

$$\begin{aligned}\|\beta\|_H &= \int_a^b \|\beta'(t)\|_H\, dt = \int_a^b \sqrt{\left(\frac{\mu'(t)}{\sqrt{2}}\right)^2 \frac{1}{\sigma'(t)^2} + 2\frac{\mu'(t)}{\sqrt{2}}y'(t).0 + \sigma'(t)^2 \frac{1}{\sigma'(t)^2}}dt \\ &= \int_a^b \frac{\sqrt{\frac{\mu'(t)^2}{2} + \sigma'(t)^2}}{\sigma'(t)}dt = \int_a^b \frac{\frac{1}{\sqrt{2}}\sqrt{\mu'(t)^2 + 2\sigma'(t)^2}}{\sigma'(t)}dt = \frac{1}{\sqrt{2}}\int_a^b \frac{\sqrt{\mu'(t)^2 + 2\sigma'(t)^2}}{\sigma'(t)} \\ &= \frac{1}{\sqrt{2}}\|\alpha\|_{H_F}\end{aligned}$$

Donde, $\|\alpha\|_{H_F} = \sqrt{2}\,\|\beta\|_H$. Portanto,

$$d_F((\mu_1,\sigma_1),(\mu_2,\sigma_2)) = \sqrt{2}\rho\left(\left(\frac{\mu_1}{\sqrt{2}},\sigma_1\right),\left(\frac{\mu_2}{\sqrt{2}},\sigma_2\right)\right)$$

onde d_F é a distância de Fisher de $\mathcal{H}_F$ e ρ, como vimos, é a distância hiperbólica do modelo do semiplano de Poincaré $\mathcal{H}^2$.

Podemos ver em [17] e [44], que as geodésicas do modelo $\mathcal{H}_F$ com a métrica de Fisher são as semielipses de excentricidade $\frac{1}{\sqrt{2}}$ e as semirretas ortogonais ao bordo $\mathbb{R}$.

Sobre a métrica α-deformada de Burbea-Rao

O artigo [7] apresenta algumas considerações sobre a estrutura geométrica de Burbea e Rao para variedades estatísticas. Considerando o caso particular das distribuições normais gaussianas univariadas temos, para um $\alpha > 0$ e um ponto $z = x + iy \in \mathcal{H}^2$, a métrica

$$\begin{cases} x = [A(\alpha)]^{-\frac{1}{2}}\mu, \quad y = \sigma; \\ ds_\alpha = B(\alpha)\, y^{-(\alpha+1)}\left(dx^2 + dy^2\right) \end{cases},$$

onde

$$A(\alpha) = \left(\alpha^{\frac{1}{2}} - \alpha^{-\frac{1}{2}}\right)^2 + 2\alpha^{-1}; \quad B(\alpha) = \alpha^{-\frac{3}{2}} (2\pi)^{\frac{1-\alpha}{2}} A(\alpha).$$

Essa métrica é conhecida como métrica de Kähler.

Quando $\alpha = 1$ temos a métrica hiperbólica de Poincaré. A curvatura gaussiana desse modelo é sempre negativa, ou seja, trata-se sempre de uma geometria hiperbólica.

Ao considerar uma parametrização polar para os pontos de $\mathcal{H}^2$ o artigo [7] apresenta as equações que devem ser resolvidas para encontrar as geodésicas do modelo e a distância entre dois pontos utilizando tal geodésica. Entretanto, para o cálculo da distância é preciso resolver um sistema não linear com três equações. Um exemplo (retirado de [7]) de geodésica para um caso particular é exibido na Figura 1.8. Observamos que, diferentes valores de α geram distintas geodésicas.

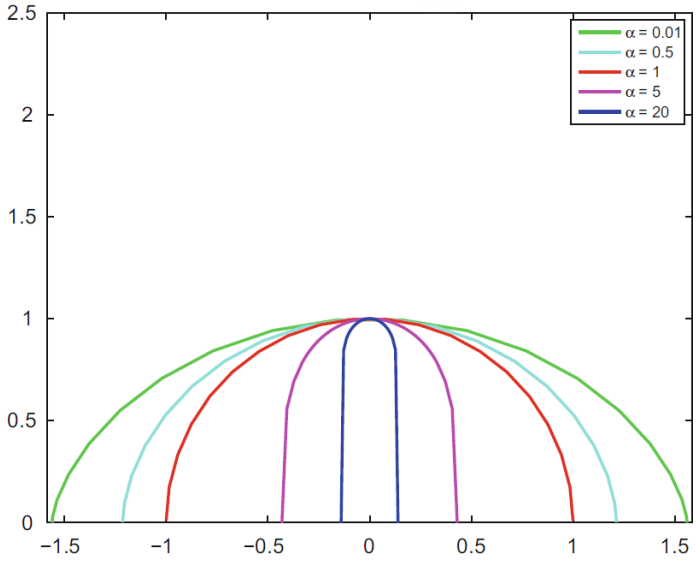


Figura 1.8: Exemplos de geodésicas com a métrica α-deformada de Burbea-Rao.

Sobre a distância α-ordem de Hellinger

O artigo [7] apresenta ainda uma solução alternativa para o cálculo da distância entre duas distribuições $N\left(\mu_1, \sigma^2\right)$ e $N\left(\mu_2, \sigma^2\right)$. A distância $\alpha-$ordem de Hellinger é dada por:

$$d_{He}\left(\left(\mu_1, \sigma_1^2\right), \left(\mu_2, \sigma_2^2\right); \alpha\right) =$$

$$= \frac{2(2\pi)^{\frac{(1-\alpha)}{4}}}{\alpha^{5/4}} \left[\left(\sigma_1^{\frac{1-\sigma}{2}} - \sigma_2^{\frac{1-\sigma}{2}}\right)^2 + 2(\sigma_1\sigma_2)^{\frac{1-\alpha}{2}} \left(1 - \left(\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}\right)^{\frac{1}{2}} \exp\left(\frac{-\alpha(\mu_1 - \mu_2)^2}{4(\sigma_1^2 + \sigma_2^2)}\right)\right)\right]^{\frac{1}{2}}$$

2 Introdução ao Processamento de Imagens

Neste capítulo apresentamos de forma resumida a teoria concernente ao processamento de imagens digitais, especialmente no que diz respeito aos aspectos matemáticos. As principais referências utilizadas nesta abordagem são [59], [29], [58] e [38].

O chamado processamento de imagens digitais consiste de qualquer manipulação de dados no qual a entrada e a saída são imagens (seção 2.2). O objetivo do processamento é a melhoria da imagem para visualização humana e/ou para análises subsequentes via computador.

As primeiras técnicas de processamento de imagens surgiram no começo do século XX para melhorar imagens digitalizadas de jornais que eram enviadas através de cabos submarinos entre Londres e Nova York. Com o advento do computador, o processamento de imagens, que era analógico, através de dispositivos óticos, passou, gradualmente, a ser digital tornando o processamento mais rápido, preciso e confiável, além da possibilidade de utilização de códigos corretores de erros na transmissão das imagens.

Figura 2.1: Reconhecimento de placas.

Atualmente, as técnicas de processamento de imagens são empregadas em diversas áreas do conhecimento. A medicina, por exemplo, se beneficia dessas técnicas já que o uso de diagnósticos por imagem se tornou rotineiro. Imagens de raios X, tomografias e outras imagens biométricas podem ser melhoradas para que a interpretação dos resultados seja mais satisfatória. Os geógrafos puderam aprimorar o estudo de padrões de poluição em imagens aéreas ou de satélite. A biologia pode contar com o processamento digital de imagens para processar automaticamente imagens obtidas em microscópios para, por exemplo, contar quantas unidades de uma determinada célula contém uma amostra. Podemos ainda citar diversas outras áreas que se beneficiaram com o avanço do processamento de imagens: astronomia, segurança (leitura biométrica), física, fotografia, sensoriamento remoto, geoprocessamento, metereologia, publicidade, reconhecimento mi-

litar, dentre outras. Vemos na Figura 2.1, retirada de [29], um exemplo da aplicação de processamento de imagens para reconhecimento de placa de carro.

Um sistema geral de processamento de imagens pode ser esquematizado conforme Figura 2.2 ([38], [29]):

Figura 2.2: Sistema de processamento de imagens.

A aquisição, que será discutida com mais detalhes adiante, transforma a imagem real em uma representação numérica compatível com as etapas subsequentes. Para isso, faz-se necessário um dispositivo físico para imageamento sensível a espectro de energia eletromagnética (como raios X, espectro visível da luz, ultravioleta, infravermelho, etc) e um digitalizador para a conversão da energia elétrica em sinal digital.

O armazenamento de imagens é uma parte importante do sistema de processamento de imagens devido à grande quantidade de bytes das imagens. No nosso contexto o armazenamento de imagens pode ser dividido em 3 categorias: (1) armazenamento durante o processamento e, por isso, por curto tempo, (2) armazenamento online de recuperação relativamente rápida e (3) armazenamento em arquivo.

O processamento da imagem é feito normalmente sob forma algorítmica. Técnicas de processamento que funcionam bem em uma área podem não funcionar em outras, ou seja, as soluções de melhoria buscadas são específicas ([29]). Dentre as técnicas de processamento de imagens podemos citar: redução de ruído, aumento de contraste, suavização de imagem, segmentação, reconhecimento de padrões, morfologia matemática, etc.

A comunicação ou transmissão de imagens ou vídeos enfrenta um grande desafio no que diz respeito a grandes volumes de dados para serem transmitidos no menor tempo possível.

A exibição de imagens é feita principalmente por monitores de vídeo. As resoluções espaciais dos monitores foram melhoradas consideravelmente nos últimos anos com o avanço da tecnologia. A resolução espacial *máxima* de um monitor normalmente é dada pela quantidade de pontos coloridos, ou *pixels* presentes nele. Normalmente a resolução é escrita na forma $A \times B$ significando A pontos na horizontal e B pontos na vertical do monitor. Entretanto, um monitor pode se configurado para apresentar uma resolução mais baixa agrupando pontos para representar o mesmo elemento da imagem. As resoluções mais comuns de monitores de computador são 1024×768 e 1920×1080, embora já seja possível encontrar monitores com resoluções maiores com, por exemplo, 3840×2160 pixels. O tamanho físico de cada um desses pontos coloridos depende do

tamanho da tela e da resolução. Por exemplo, em um monitor de $21,5$ polegadas com resolução de 1920×1080 um pixel tem $0,02479 \times 0,02479\ cm^2$ de área.

Existem diversas tecnologias de hardware utilizadas para exibição de imagens em monitores de vídeos. Dentre elas temos: CRT, LCD, Plasma e OLED [32].

Os monitores CRT (*Cathode Ray Tube*), desenvolvidos por Karl Braun, foram usados em todas as televisões até o final do século XX. Entretanto, com o avanço da tecnologia surgiram opções que são digitais, menos volumosas e consomem menos energia. Atualmente, os monitores CRT raramente são utilizados.

Presente na maioria dos monitores de vídeo na atualidade, as telas de LCD (*Liquid Cristal Display*) são compostas por arranjos de pixels (com construção similar aos chips) constituídos de cristais líquidos que irão determinar qual cor é exibida. As células da tela não emitem luz própria sendo necessária uma fonte externa. Porém, cada pixel tem seu próprio transistor e pode ser ligado individualmente o que não acontecia nos monitores CRT. As telas LCD apresentam baixo consumo de energia, melhor eficiência do que os monitores CRT e menor desgaste. Entretanto, o ângulo de visão é reduzido e não se consegue reproduzir o preto absoluto.

As telas de plasma são formadas por câmaras seladas com gases nobres (xeon e neon) em seu interior, entre duas placas de vidro. Elas apresentam boa luminosidade e bom nível de contraste, além de melhor ângulo de visão. Entretanto, além desse tipo de tela ser mais sucetível a defeitos, as células de gás que a compõe são muito grandes, dificultando a produção de monitores pequenos com boa resolução.

As telas OLED (*Organic Light Emitting Diode*) são baseadas no uso de substâncias orgânicas que brilham ao receber um impulso elétrico. Cada célula da tela emite, portanto, sua própria luz, fazendo com que as telas sejam finas. As telas OLED tendem a ser mais compactas e econômicas. Entretanto, ainda é uma tecnologia nova e por isso, os custos são elevados. Além disso, a vida útil é reduzida devido ao material orgânico. Inicialmente essa tecnologia tem sido usada em telas pequenas, como as de celulares. Uma característica muito interessante das telas OLED é que, como utiliza compostos líquidos, podem ser construídos em diversos tipos de superfícies, podendo, por exemplo, ser utilizado para a construção de telas flexíveis.

Em todos os tipos de telas que vimos, cada pixel é composto por subpixels com divisão das cores. As telas de CRT, LCD e Plasma apresentam o padrão RGB (Red, Green, Blue - Vermelho, Verde, Azul) para cada pixel com os três subpixels de mesmo tamanho. Podemos ver na Figura 2.23[1] como é feita a distribuição dos pixels em um monitor CRT e LCD. A Figura 2.3 à esquerda representa o padrão de distribuição de pixels de uma TV CRT enquanto o a Figura 2.3 ao meio representa um monitor de computador CRT. A

[1] Imagem retirada de: http://www.dsource.in/course/digital-typography-2/screentypes.html

Figura 2.3 à direita é o padrão de distribuição de pixels de um monitor LCD.

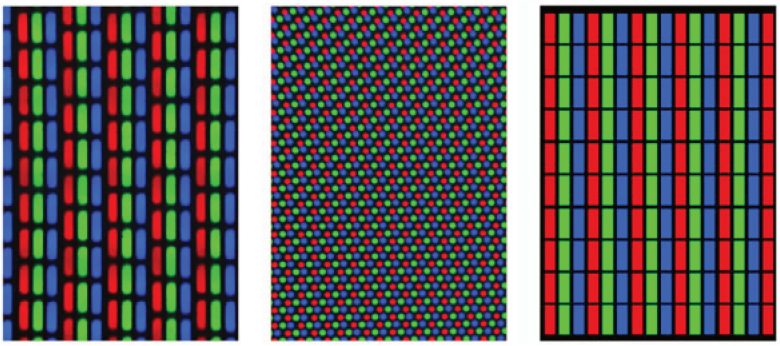


Figura 2.3: Diferentes padrões de subpixels em monitores CRT e LCD.

Os monitores OLED já estão presentes a algum tempo em celulares da Samsung com o padrão de pixels Pentile[2]. Na Figura 2.4 vemos alguns padrões de Pentile: o padrão à esquerda[3] está presente, por exemplo, no celular Samsung Galaxy S III, o tablet Samsung Galaxy Tab Pro 10.1 apresenta o segundo padrão[4] e o padrão diamond, apresentado na terceira imagem[5] está presente, por exemplo, no Samsung Galaxy S4. Nesse caso, os subpixels são agrupados aos pares. Além disso, os tamanhos dos subpixels são diferentes.

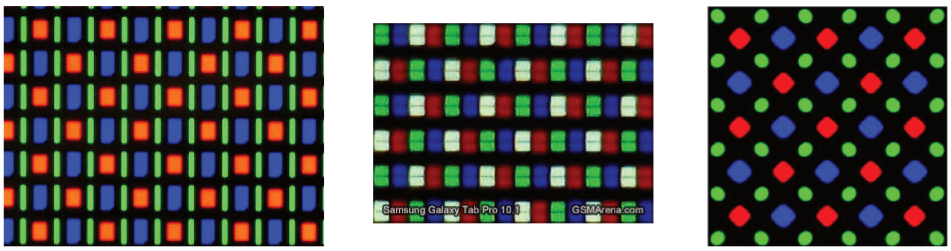


Figura 2.4: Padrões Pentile de distribuição de subpixels.

A outra forma bastante comum de exibição de imagens é a reprodução de imagens em papel. As primeiras câmeras Kodak geravam imagens em formato circular como podemos ver em algumas fotografias antigas. Por exemplo, a Figura 2.5 é uma fotografia de 1890 de George Eastman, fundador da Kodak e inventor do filme fotográfico que permitiu a popularização da fotografia. Atualmente existem diversas formas de reprodução de imagens em papel. A melhor e mais cara é a reprodução fotográfica. Temos também o uso da técnica de *halftoning* que é o método usado por jornais e pelas impressoras convencionais.

[2] Para mais informações: http://www.nouvoyance.com/
[3] Imagem disponível em: http://www.oled-info.com/pentile
[4] Imagem disponível em: http://www.gsmarena.com/samsung_galaxy_tab_pro_10_1-review-1044p2.php
[5] Imagem disponível em: http://abertoatedemadrugada.com/2013/05/galaxy-s4-tem-subpixeis-em-losango.html

Figura 2.5: Imagem circular gerada por uma câmera Kodak em 1890.

Neste trabalho abordamos apenas aspectos da aquisição e do processamento de imagens.

2.1 Dispositivos CCD

Normalmente sensores ópticos primeiro convertem uma imagem a ser adquirida em sinal elétrico analógico. Em seguida, circuitos eletrônicos chamados *frame grabbers* (dispositivo de captura de quadro) convertem esse sinal analógico em sinal digital a fim de tornar possível a interpretação por computador. Várias opções para a conversão de imagens em sinais elétricos estão disponíveis. Neste trabalho concentraremos nos dispositivos chamados CCD (dispositivo de carga acoplada - charged-couple device) que são dispositivos muito comuns utilizados atualmente. Na Figura 2.6 vemos uma representação de como uma imagem é adquirida.

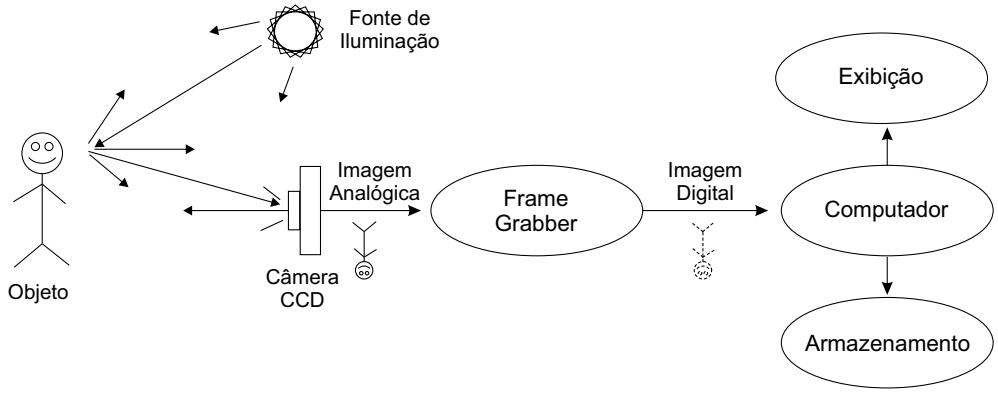


Figura 2.6: Esquema de aquisição de imagem.

Conforme [58] e [27], os dispositivos CCD consistem de uma lente e uma imagem plana (chip), com células semicondutoras fotossensíveis atuando como capacitores, que armazenam carga elétrica resultante da energia luminosa incidente. Quando um fóton incide sobre uma célula o efeito fotoelétrico faz com que o chip libere um elétron. Dessa forma, as células funcionam como um balde armazenando água da chuva. Quanto mais

fótons/chuva mais elétrons/água. Após o fim do tempo de exposição basta contar quantos fótons temos em cada balde. O tempo de exposição está diretamente ligado à nitidez e ao nível de detalhamento da imagem, pois quanto maior a exposição mais luz é coletada. É claro que podemos controlar o tempo de exposição da maneira que nos for mais conveniente. Curiosamente, com o olho humano não temos esse controle. Independente do tempo que ficarmos olhando uma imagem sem piscar, a imagem que se forma na retina não se altera já que a retina acumula luz apenas por uma fração de segundos e passa a descartar o restante após esse intervalo. A Figura 2.7 esquematiza a captura de imagem por um dispositivo CCD.

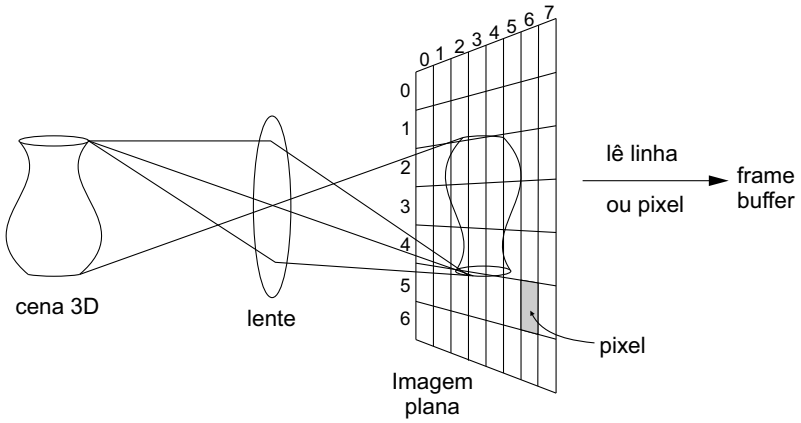


Figura 2.7: Esquema de câmera CCD.

Quanto maior o tamanho da imagem plana, maior é a área que pode ser imageada. Além disso, a resolução da imagem final está diretamente ligada ao tamanho de cada célula do chip.

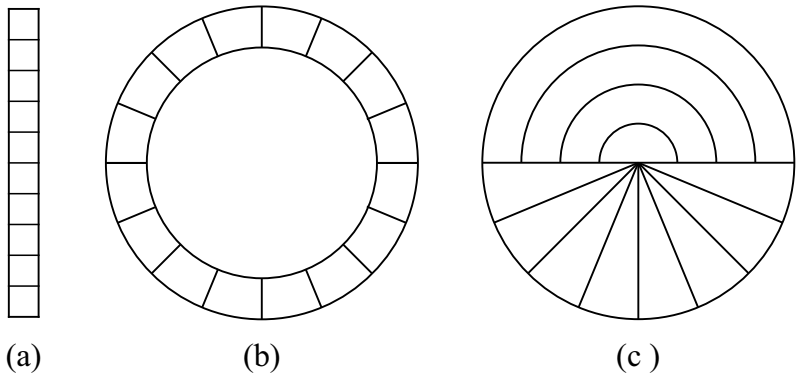


Figura 2.8: Exemplos de geometria do chip CCD.

Usualmente a imagem plana é uma malha retangular, mas existem outras geometrias úteis para a imagem plana [58]. Quando é preciso, por exemplo, medir apenas a largura da imagem podemos utilizar a geometria linear (Figura 2.8 (a)). Para a inspeção de mostradores analógicos, por exemplo o relógio e o velocímetro, a geometria circular (Figura 2.8 (b)) pode ser conveniente. A geometria "ROSA" (Figura 2.8 (c)) foi uma solução para encontrar o espectro de potência de uma imagem.

Sensores CCD de formato circulares foram desenvolvidos inspirados na retina humana. A resolução é maior no centro da imagem (ou FOV- Field of View - Campo de visão) em detrimento das bordas (ou periferia). Nesses tipo de sensores, os pixels são distribuídos em círculos concêntricos e o raio de cada círculo aumenta exponencialmente com o número da circunferência. Essa distribuição de pixels tem a interessante característica de reduzir seletivamente a quantidade de dados da imagem permitindo um processamento mais rápido ou o envio dessas imagens por redes com baixas taxas de transferência. Esses sensores foram inicialmente desenvolvidos para uso em robótica. Entretanto, sua estrutura também é interessante para o processamento de imagens em tempo real, active vision (visão ativa), transmissão de imagens comprimidas, etc [42].

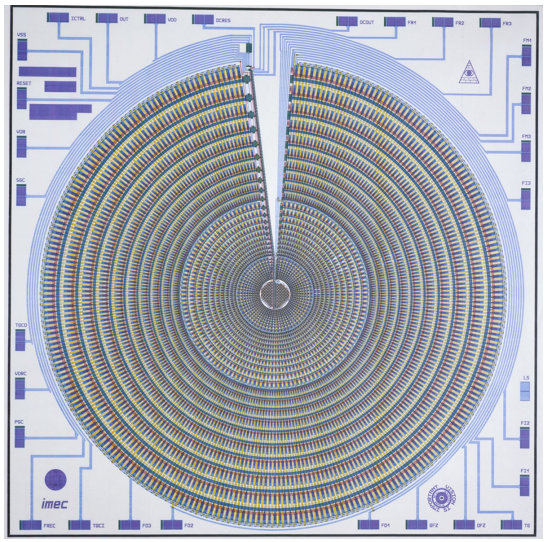


Figura 2.9: CCD de geometria circular fabricado em 1989.

Na Figura 2.9 um exemplo[6] de geometria circular de CCD criado por Debusschere and Kreider e Van der Spiegel e fabricado por IMEC e a Universidade da Pensilvânia em 1989 [19] com aplicação em robótica. Temos ainda o artigo [42] de 1998 que relata a construção de um chip CMOS log-polar (falaremos sobre esse tipo de chip após a Figura 2.10) exibindo, inclusive, uma imagem (Figura 2.10) capturada com o chip construído. Outras referências citando CCDs de geometrias circulares são [69], [67], [64] e [55]. O avanço na tecnologia de fabricação de chips possibilita a implementação de formatos personalizados. Essa possibilidade inspirou o estudo do tema apresentado do Capítulo 3.

Como vimos, o tempo de exposição à luz está intrinsecamente ligado à qualidade da imagem. Dessa forma, para melhorar a relação sinal-ruído (fidelidade entre a cena real e a imagem) aumenta-se o tempo de exposição de luz [7]. Essa abordagem é equivalente a fazer múltiplas aquisições, cada uma com o mesmo tempo de exposição à luz, e considerar a soma das aquisições. Também é comum considerar a média das aquisições. Entretanto,

[6] Imagem retirada de: http://image-sensors-world.blogspot.com.br/2013/01/exhibition-features-art-of-ccd-layout.html

não é comum levar em consideração a variância dos dados que é um estimador básico de ruído utilizado em processamento probabilístico de imagem [7]. O artigo [12] utiliza um modelo em que a média e o desvio padrão de uma vizinhança é considerada para determinar o tom de cinza de cada ponto da imagem.

Figura 2.10: Imagem capturada com um chip CMOS log-polar.

Outro sensor de aquisição de imagens bastante comum é o CMOS (Complementary Metal-Oxide Semiconductor) [54]. A conversão de luz em elétrons é feita da mesma maneira que nos chip CCD. Entretanto, as diferenças começam na maneira como os elétrons são lidos. Nos sensores CCD as cargas das células são transportadas ao longo do chip, lidas em um canto da malha e em seguida convertidas para um valor digital em um conversor separado. Já nos sensores CMOS cada célula é lida e convertida individualmente. Por utilizar um conversor externo o processo de geração de imagens dos sensores CCD é mais lento do que dos CMOS. Os sensores CCD tem uma manufatura especial para que seja possível transportar a carga sem distorção, o que leva a uma qualidade elevada em termos de fidelidade a sensibilidade à luz, e, por isso, são sensores mais caros. Os sensores CMOS são produzidos com a mesma tecnologia da maioria dos microprocessadores, o que os torna mais baratos. Entretanto, esses sensores são mais suscetíveis a ruídos. Além disso, os sensores CMOS gastam menos energia do que os sensores CCD. Por serem mais antigos os sensores CCD foram mais desenvolvidos, propiciando qualidades melhores.

As câmeras CCD e CMOS comuns utilizam chips feitos de silício. Entretanto, o silício não consegue capturar alguns comprimentos de ondas como, por exemplo, o infravermelho. O imageamento de tais comprimentos de onda é consideravelmente mais complicado, volumoso e caro. Esse problema motivou um grupo de pesquisadores da Universidade de Rice a desenvolver um protótipo de câmera mais simples, menor e mais barata do que as câmeras digitais e que pode operar de maneira eficiente através de uma faixa espectral muito mais ampla do que as câmeras convencionais com chips de silício.

A "single-pixel" (assim chamada por possuir um único detector de fótons) é baseada em uma nova e crescente área chamada "Compressive Sensing" (ou Compressed Sampling). O objetivo é o de utilizar um pequeno número de amostras posicionadas aleatoriamente para reconstruir todos os valores de um sinal esparso. Ao invés de capturar amostras de pixels através da média de uma matriz de detectores de fótons, a "single-pixel" mede produtos internos entre a cena em questão e um conjunto de funções teste. Apesar de ser uma tecnologia muito promissora, o desing dessa câmera ainda precisa passar por muitos ajustes até que se torne comercial. As referências [61], [23], [28], [33], [63], [14], [24] e [46] abordam com detalhes os assuntos deste parágrafo.

2.2 Definição de Imagem

Seja $\Omega \subset \mathbb{R}^2$ não vazio. Definimos como *imagem* um modelo $f : \Omega \to \mathbb{R}$ que associa a cada coordenada $(x, y) \in \Omega$ um valor real $t = f(x, y)$ que é proporcional à intensidade luminosa em (x, y). Cada ponto $(x, y) \in \Omega$ é chamado de *pixel* (ou *elemento da imagem*) e Ω é chamado de *espaço suporte de pixels*. Usualmente Ω é um subconjunto de $\mathbb{Z}^2$. Observamos que aqui a noção de pixel é diferente da apresentada anteriormente para tela de monitor.

A luz é uma fonte de energia e por isso t deve ser não negativo. Neste trabalho chamaremos t de tom de cinza (ou nível de cinza). A intensidade da luz de uma imagem de uma cena qualquer percebida pelo olho humano pode ser modelada segundo duas componentes:

(1) A iluminância $i(x, y)$ que é a quantidade (em lúmens) de luz incidindo nos objetos observados;

(2) A reflectância $r(x, y)$ que é a taxa de luz refletida pelos objetos observados.

Dessa forma, um modelo matemático usual é dado por $t = f(x, y) = i(x, y)r(x, y)$ onde $0 < i(x, y) < \infty$ e $0 < r(x, y) < 1$, ou seja, t é fração de luz refletida pelo objeto. Observamos que a iluminância é determinada pela fonte de luz e a reflectância pelas características dos objetos. Por exemplo [38], $i(x, y) = 900$ em um dia ensolarado e $i(x, y) = 10$ em um escritório iluminado. Por outro lado, $r(x, y) = 0,8$ para uma parede branco-fosca e $r(x, y) = 0,01$ para um veludo preto. Quando Ω é compacto e, sendo f contínua, t pertence a um intervalo limitado.

Quando trabalhamos com imagens é importante diferenciar os conceitos de imagem analógica, imagem digital e modelo matemático de imagem.

Uma *imagem analógica* é um modelo $f(x, y)$ que tem, teoricamente, precisão tão boa quanto se queira em intensidade em cada coordenada (x, y).

Uma *imagem digital ou imagem discreta* é um modelo $f(x, y)$ com

$$\Omega = \left\{(x, y) \in \mathbb{Z}^2 : 0 \leqslant x < m \text{ e } 0 \leqslant y < n\right\} \tag{2.1}$$

que pode ser representada por uma matriz $M_{m\times n}$ finita de amostras de intensidades luminosas, ou seja, com entradas $t_{i,j} = f(i, j)$. As intensidades máxima e mínima são finitas. Observemos que as coordenadas de cada elemento da matriz é um pixel. Um pixel é, portanto, a representação abstrata da imagem que pode ser trabalhada em um ambiente digital. Iremos considerar o espaço suporte Ω associado a imagens digitais muitas vezes neste trabalho e, por isso, iremos chamá-lo de *espaço suporte tradicional*, que indicaremos por Ω_T.

Na Figura 2.6 temos representado em qual parte do processo de aquisição surgem as imagens analógica e digital.

Um *modelo matemático de imagem monocromática* (ou em tons de cinza) é simplesmente um modelo f no qual é permitido que f assuma valores negativos. Modelar matematicamente a imagem como uma função é muito útil para descrever a imagem e também para definir operações sobre a imagem.

Existe diferença entre a informação visual e a informação descritiva. A informação vista na tela de um computador, por exemplo, é visual. A informação descritiva refere-se ao modelo matemático de imagem que representa os objetos visualizados.

Para criarmos uma imagem digital a partir de uma imagem analógica devemos discretizá-la por meio de um processo chamado de *amostragem* (na Figura 2.6 esta etapa é realizada pelo dispositivo "frame grabber"). Esta discretização também pode ocorrer nos tons de cinza, que pode ser aproximado para um número inteiro. Neste caso, o procedimento é chamado de *quantização em níveis de cinza*. Tal procedimento é necessário para que a imagem possa ser processada pelo computador.

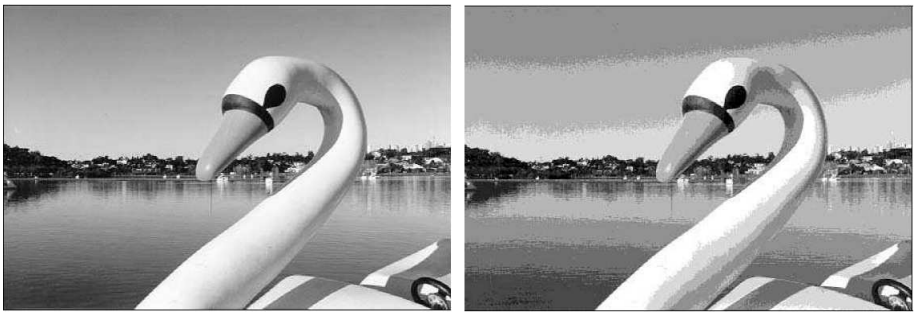


Figura 2.11: Imagens com 256 e 8 tons de cinza.

O processo mais comum de quantização faz com que $t = f(x, y)$ assuma valores inteiros entre 0 e $2^k - 1$. Chamaremos o intervalo $[0, 2^k - 1]$ de escala de cinza. Esse formato da escala de cinza está intrinsicamente ligado ao sistema binário do computador.

O valor de k está associado à quantidade de tons de cinza que t pode assumir que, neste exemplo, é de 2^k tons. Para o olho humano é suficiente trabalharmos com 64 tons de cinza [38]. Entretanto, o valor mais comum para k é 8 resultando em 256 valores de tons de cinza possíveis para t. Um caso particular importante é quando $k = 1$. Nesse caso, os pixels assumem apenas os valores 0 e 1, a imagem é dita binária e é visualizada em preto e branco. Na Figura 2.11 (retirada de [38]) vemos a mesma imagem representada com 256 tons de cinza (à esquerda) e com 8 tons de cinza (à direita).

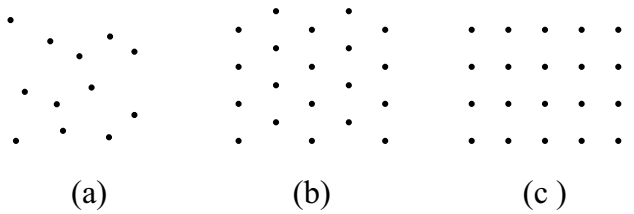


Figura 2.12: Diferentes redes de pontos.

Dada uma imagem analógica, que é "contínua" em $\mathbb{R}^2$, para obter uma imagem digitalizada, ou seja, discretizada em $\mathbb{Z}^2$, é feita uma amostragem de $\mathbb{R}^2$. Este processo consiste em considerar uma rede de pontos (pixels). Usualmente, essa rede de pontos é uniformemente distribuída, entretanto isso não é uma condição necessária. Na Figura 2.12 algumas redes de pontos são consideradas. A Figura 2.12 (a) é um exemplo de rede de pontos que não é uniformemente distribuída, a Figura 2.12 (b) é uma rede uniforme triangular e a Figura 2.12 (c) é uma rede uniforme quadrada.

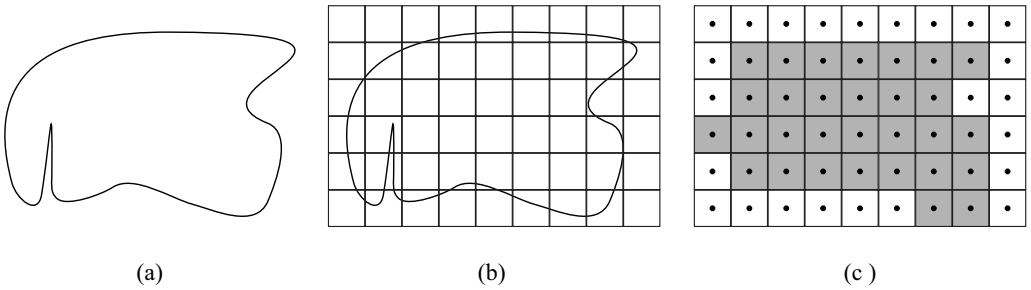


Figura 2.13: Exemplo de digitalização de imagem.

O processo mais comum de amostragem consiste em dividir o plano xy em uma grade retangular de pixels e considerar o centro de cada elemento da grade como sendo um par ordenado de $\mathbb{Z}^2$. A Figura 2.13 é um exemplo do processo de digitalização de uma imagem. Dada a imagem analógica, que é "contínua" (Figura 2.13 (a)), o plano é dividido em uma grade (Figura 2.13 (b)). A amostragem pode ser vista na Figura 2.13 (c) juntamente com os pontos da rede de pixels. Neste exemplo, um elemento da grade recebe um tom de cinza se o ponto central da grade intersecta a imagem contínua. Observamos que as propriedades geométricas como, por exemplo, a convexidade, das imagens analógica e digital podem ser diferentes. Observamos ainda que o formato da imagem digital depende

do posicionamento e do tamanho da grade utilizada na amostragem. Quanto mais divisões tiver a grade melhor a imagem digitalizada se aproximará da imagem "contínua".

Vejamos, na Figura 2.14, uma representação de uma imagem digital plana. Observamos que o sentido de orientação dos eixos é diferente do sistema cartesiano e optamos por utilizar a orientação da Figura 2.14 em todo nosso trabalho, pois essa é a orientação utilizada nos programas computacionais.

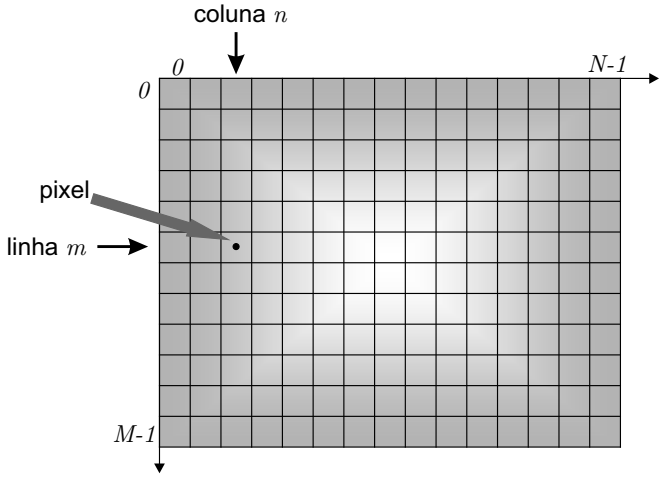


Figura 2.14: Representação de imagem digital.

Ressaltamos que resolução de imagens é um conceito que pode ser utilizado em diversos contextos.

No processo de aquisição de uma imagem digital sua resolução é dada em pixels que, como vimos, independe de unidade de medida estabelecida. Nessa etapa, a resolução está ligada à quantidade de células do sensor CCD. Se compararmos dois sensores de mesmo tamanho, aquele com maior número de células terá maior resolução, pois assim as células serão menores e, dessa forma, conseguirão captar fragmentos menores da cena e a cópia digital da imagem será mais fiel à original.

Figura 2.15: Diferentes resoluções de imagem.

Ao armazenar a imagem, diferenças de formatos, espaço em disco entre outros fatores faz com que alguns pixels adquiridos sejam descartados, podendo diminuir a

resolução da imagem. Na Figura 2.15 podemos ver a mesma imagem representadas com o mesmo tamanho físico e com resoluções de 512×512 pixels (à esquerda), 128×128 (no meio) e 64×64 (à direita).

Já a resolução de uma imagem exibida em um monitor depende do tamanho e da quantidade de pixels da tela. Se o monitor estiver na resolução máxima temos uma correspondência de um para um entre os pixels da imagem digital e os pontos do monitor. Assim, dada uma imagem digital, ou seja, uma quantidade fixa de pixels, considerando monitores de mesmo tamanho, quanto maior for a resolução do monitor, menor será a imagem apresentada na tela. Na Figura 2.16 vemos um exemplo do logotipo da Unicamp (460×442 pixels) exibida em monitores de mesmo tamanho (21,5 polegadas) com resoluções diferentes: na esquerda a resolução é de 1980×1080 pixels enquanto na direita é de 1280×720.

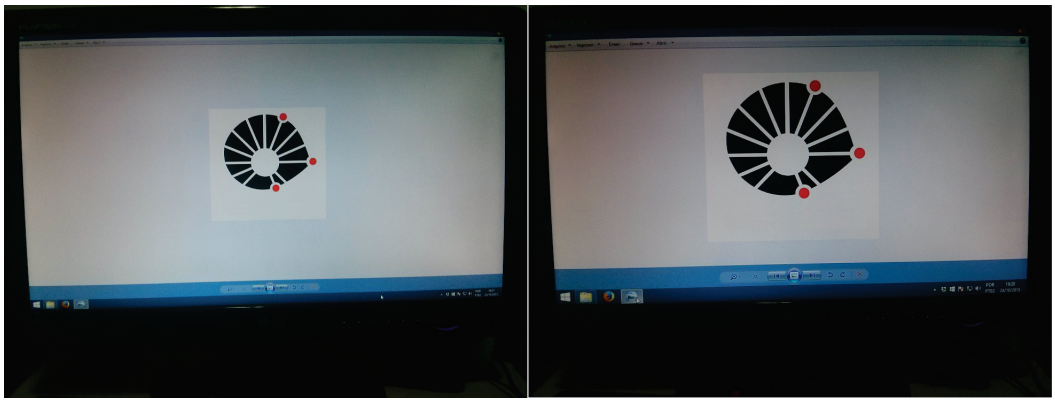


Figura 2.16: Mesma imagem em dois monitores de mesmo tamanho com resoluções diferentes.

A quantidade de pixels que a imagem possui, juntamente com a resolução do monitor é o que determina a resolução da imagem na tela. Por exemplo, uma imagem com 1920×1080 pixels preencherá completamente a tela de um monitor de resolução 1920×1080, ocupará somente parte de um monitor de 2560×1440 e não caberá em um monitor de 1024×768 de forma que apenas um trecho da imagem será mostrado.

A ferramenta que conhecemos como "zoom" é útil quando a resolução da imagem digital não coincide com a resolução do monitor (o que é extremamente comum). Matematicamente o zoom é uma reamostragem da imagem para exibição e, via de regra, não altera a resolução original da imagem digital. Se o objetivo for a alteração das dimensões da imagem existem processos denominados "scaling" e "resizing" [38].

Técnicas adaptativas podem ser utilizadas para aprimorar os processos de amostragem e quantização [38]. Na amostragem a ideia é utilizar mais pontos em regiões de grande detalhe em detrimento de regiões homogêneas de grandes dimensões, pois essas regiões poderiam ser amostradas com um menor número de pixels. Como o olho humano

não é capaz de perceber pequenas diferenças nos tons de cinza perto das variações bruscas de intensidade a quantização procura utilizar poucos tons de cinza nessas regiões. A principal dificuldade para implementar essas técnicas é a necessidade de identificação antecipada das regiões da imagem e das fronteiras entre elas. Na Figura 2.17, retirada de [26], temos um exemplo de amostragem adaptativa utilizada para obter efeito mosaico em uma imagem.

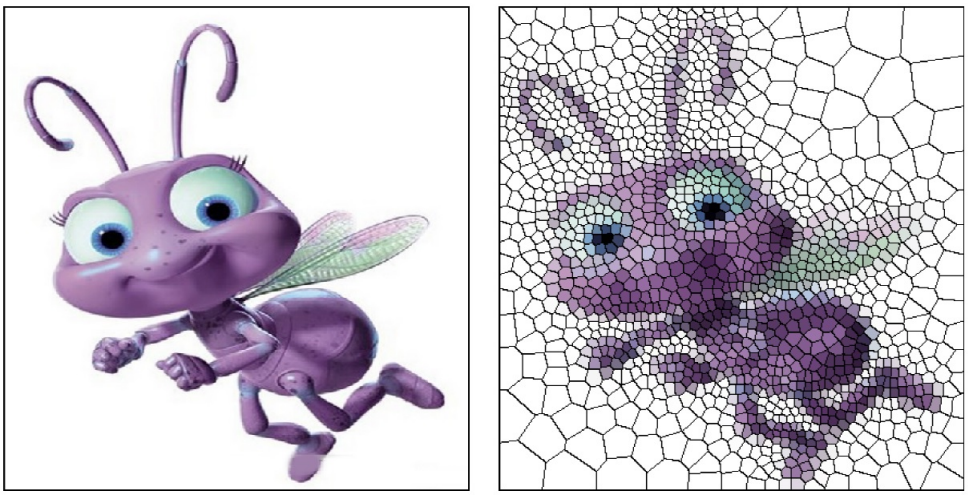


Figura 2.17: Amostragem adaptativa para efeito mosaico em imagem.

O processo de digitalização envolve decisões com relação aos valores de m, n e k. É comum, em computação, considerar que esses valores são potências de 2. Em princípio, quanto maiores esses valores, melhores os resultados, entretanto, enfrentamos um problema quanto ao armazenamento da imagem já que, para ser armazenada, a imagem necessita de kmn bits. A Figura 2.18, por exemplo, necessita de 2.097.152 bits ou 262.144 bytes (1 byte = 8 bits) para ser armazenada sem qualquer técnica de compressão ou compactação. Além disso, é pertinente citar que "qualidade de imagem" é um conceito relativo, pois depende dos requisitos da aplicação dada.

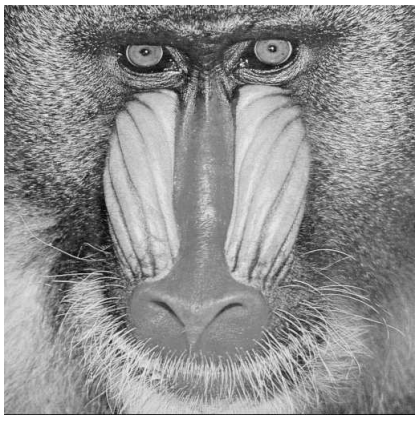


Figura 2.18: Imagem de 512×512 pixels e 8 bits.

2.3 Topologia da Imagem

Nesta seção apresentaremos técnicas que são úteis em processamento de imagens digitais.

Se uma imagem for amostrada de maneira tradicional, em forma de grade retangular (ou matriz $M_{m\times n}$), podemos definir quem são os vizinhos de um pixel apenas observando as arestas e vértices do retângulo que o delimita. Entretanto, como vimos, outras amostragens são possíveis. Para esses casos (especialmente quando a amostragem não for uniformemente distribuída) pode ser complexo definir quem são os vizinhos de um pixel e muitas operações sobre as imagens necessitam desse conceito. Assim, introduziremos a noção de grafo para nos auxiliar nas definições de vizinhança e distância entre pixels.

Um grafo $\mathcal{G}$ não orientado associado a uma rede de pixels é um par (V, E) de vértices V e arestas E onde $V = \{v_1, ..., v_r\}$ é o conjunto de pontos da rede e $E = \{e_1, ..., e_s\}$ é uma família de pares (v_i, v_j) não ordenados representados pelas arestas.

Um grafo é dito simples se não contém "loops" (ou seja, arestas do tipo (v_i, v_i)) e se não existe mais de uma aresta conectando dois vértices distintos. Grafos usados em processamento de imagens são simples. Os tipos mais comuns de grafos usados em análise de imagens são os grafos 4-conexos e 8-conexos que podem ser visualizados na Figura 2.19.

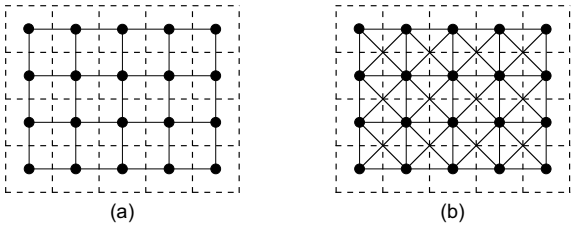


Figura 2.19: Grafos (a) 4−conexo e (b) 8−conexo.

Os vizinhos de um pixel v em um grafo $\mathcal{G} = (V, E)$ são dados por [59]:

$$N_{\mathcal{G}}(v) = \{v' \in V \mid (v, v') \in E\}.$$

Observemos que em um grafo 4-conexo dois pixels são vizinhos quando possuem uma aresta comum dos quadrados que os delimitam (pontilhado na Figura 2.19). No caso 8-conexo temos que dois pixels são vizinhos quando possuem uma aresta comum dos quadrados pontilhados que os delimitam ou um vértice comum desses mesmos quadrados pontilhados.

Uma sequência $(v_0, v_1, ..., v_k)$ de vértices distintos de um grafo é um caminho $\mathcal{P}$ de comprimento $l(\mathcal{P}) = k$ quando v_i e v_{i+1} são vizinhos para todo $i \in \{0, 1, ..., k-1\}$.

Vejamos agora o conceito de distância que é amplamente usado em análise de imagens, pois fornece uma medida de separação entre os pontos da imagem digital.

Como já citado no Capítulo 1, uma métrica d em um conjunto $\mathcal{E}$ é uma função $d : \mathcal{E} \times \mathcal{E} \longrightarrow \mathbb{R}$ que associa um elemento de $\mathcal{E} \times \mathcal{E}$ a um número real não negativo e que satisfaz as seguintes condições para quaisquer $p, q, r \in \mathcal{E}$:

(i) $d(p,q) \geqslant 0$ e $d(p,q) = 0 \Longleftrightarrow p = q$.

(ii) $d(p,q) = d(q,p)$ (simetria).

(iii) $d(p,q) \leqslant d(p,r) + d(r,q)$ (desigualdade triangular)

Existem muitas distâncias discretas que satisfazem esses axiomas. A escolha da distância para um determinado processamento de imagens depende, por exemplo, da precisão pretendida, da disponibilidade de memória e da velocidade da aplicação.

Para o processamento de imagens digitais, consideramos que $\Omega \subset V \times V$ sendo $\mathcal{G}(V, E)$ grafo, ou seja, cada um dos pixels de Ω pode ser considerado como vértice de um grafo.

Em um grafo $\mathcal{G}$, a distância discreta $d_{\mathcal{G}}$ entre os pixels p e q é dada pelo comprimento do menor caminho ligando p e q:

$$d_{\mathcal{G}}(p,q) = \min \{ l(\mathcal{P}) \mid \mathcal{P} \text{ é um caminho ligando } p \text{ e } q \}.$$

Esta definição implica que a distância depende fortemente do tipo de grafo escolhido. Suponhamos que $p = (x_1, y_1)$ e $q = (x_2, y_2)$. Se o grafo for 4-conexo, sua métrica d_4 é conhecida como métrica da soma e a definição acima se torna:

$$d_4((x_1, y_1), (x_2, y_2)) = |x_2 - x_1| + |y_2 - y_1|.$$

Por outro lado, se o grafo for 8-conexo sua métrica d_8 é conhecida por métrica do máximo e a definição acima fica:

$$d_8((x_1, y_1), (x_2, y_2)) = \max \{ |x_2 - x_1|, |y_2 - y_1| \}$$

Podemos ainda, considerar os pixels como pontos de $\mathbb{R}^2$ e utilizar a distância euclidiana d_ε:

$$d_\varepsilon((x_1, y_1), (x_2, y_2)) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Porém, essa abordagem métrica pode não levar em conta as relações de vizinhanças entre os pontos da imagem.

Normalmente, em aplicações práticas, a distância euclidiana é arredondada para o inteiro mais próximo. Entretanto, esse arredondamento faz com que a função não seja métrica, pois não satisfaz a desigualdade triangular.

Observamos na Figura 2.20 que a distância entre os pontos p e q depende da métrica utilizada. Na Figura 2.20 (a) $d_4(p,q) = 5$. Na Figura 2.20 (b) $d_8(p,q) = 3$. E,

na Figura 2.20 (c) $d_{\varepsilon}(p,q) = \sqrt{13}$. Os caminhos pontilhados nas Figuras 2.20 (a) e (b) também representam a distância entre os pontos p e q.

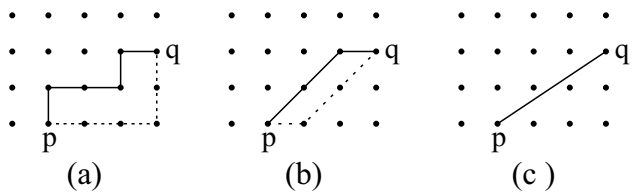


Figura 2.20: Diferentes distâncias entre os mesmos pontos.

Diremos que uma vizinhança $N_{\mathcal{G}}^{r}(p)$ em um grafo $\mathcal{G} = (V, E)$ é uma bola de raio r centrado em p se

$$N_{\mathcal{G}}^{r}(p) = \{q \in V \mid d_{\mathcal{G}}(p,q) \leqslant r\},$$

onde $d_{\mathcal{G}}$ é a distância definida sobre o grafo $\mathcal{G}$.

Vejamos, na Figura 2.21, uma bola $N_{\mathcal{G}}^{4}(p)$ para um grafo 4-conexo e um grafo 8-conexo (os elementos de $N_{\mathcal{G}}^{4}(p)$ são os pontos centrais dos quadrados hachurados).

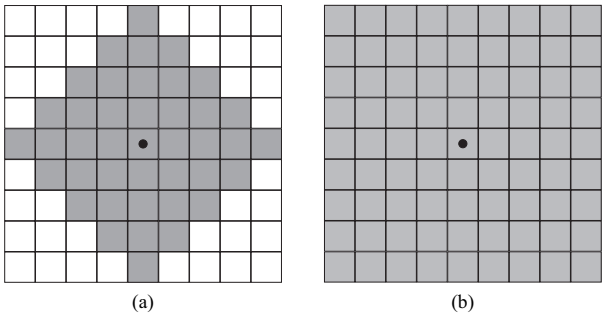


Figura 2.21: Bolas de raio 4 em grafo 4-conexo e 8-conexo.

2.4 Morfologia Matemática

Morfologia matemática, como vimos anteriormente, é um dos processamentos que podem ser feitos em uma imagem. A palavra morfologia é utilizada na biologia para designar o estudo da forma ou da estrutura das plantas e dos animais. A morfologia matemática é um ramo do processamento não linear de imagens que tem por objetivo extrair e identificar propriedades importantes de uma imagem baseadas na estrutura geométrica de elementos da imagem. Na década de 60, Jean Serra e George Matheron [38] deram início ao estudo da Morfologia Matemática ao analisar imagens binárias com o auxílio da teoria de conjuntos. Posteriormente, o estudo foi estendido para imagens em tons de cinza utilizando a teoria de reticulados. Atualmente, a morfologia matemática ainda é amplamente estudada e utilizada, possuindo, assim, uma rica literatura, além de linguagem e notação própria. Existem inúmeros métodos morfológicos tais como *melhoramento*, *segmentação*, *restauração*, *esqueletonização*, dentre outros. E diversos desses métodos são

utilizados, com sucesso, pela biologia, visão robótica, microscopia, metalurgia, imageamento médico e muitas outras áreas.

Conforme [21], os operadores morfológicos servem com uma linguagem universal para o processamento de imagens, e, por isso, suas aplicações são limitadas apenas pela elaboração de algoritmos efetivos e implementação computacional eficientes.

A ideia básica da morfologia matemática consiste em examinar a imagem, utilizando um conjunto conhecido e bem-definido chamado de *elemento estrutural*. As operações básicas da morfologia são a *erosão* e a *dilatação*. Essas operações são relativamente simples e muitas operações morfológicas sofisticadas são reduzidas a sequências de dilatação e erosão [60]. As definições destas operações dependem do tipo de imagem a ser processada: binárias, em tons de cinza ou coloridas. Seguindo a evolução histórica abordaremos primeiro as operações em imagens binárias e, em seguida, as em tons de cinza. Ressaltamos que as imagens coloridas não serão abordadas neste trabalho.

2.4.1 Elemento estrutural

O elemento estrutural (SE - *structuring element*) é um determinado conjunto de pixels utilizado para examinar a imagem a ser estudada. Geralmente, a quantidade de pixels do SE é menor do que a quantidade da imagem. O formato do SE é decidido com base em conhecimento prévio da geometria dos objetos relevantes e irrelevantes da imagem. Dessa forma, o êxito de uma operação morfológica está na escolha do SE de acordo com uma dada aplicação ou objetivo.

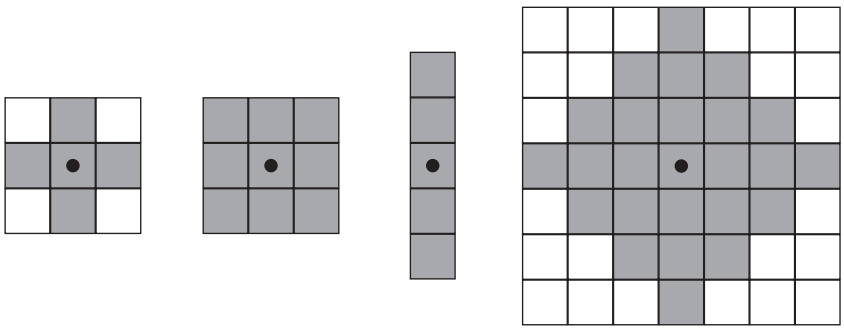


Figura 2.22: Exemplos de elementos estruturais.

As operações fundamentais da morfologia matemática necessitam da definição de uma origem para cada SE. Na Figura 2.22 temos alguns SEs, os dois primeiros são os mais comumente utilizados. Chamaremos o primeiro SE de $SE-4$ e o segundo de $SE-8$.

Observamos, entretanto, que ao sobrepor SEs à imagem, o formato destes pode se alterar dependendo da posição da origem p. A Figura 2.23 ilustra como é o formato dos SEs $SE-4$ e $SE-8$ com origem centrada em $(0,0)$.

O processamento de imagens no qual SEs são consideradas como na Figura 2.23 é o que enfocaremos neste trabalho. Entretanto, há processamentos nos quais o tratamento de pixels próximos da borda é diferente do que propomos. Vejamos alguns exemplos.

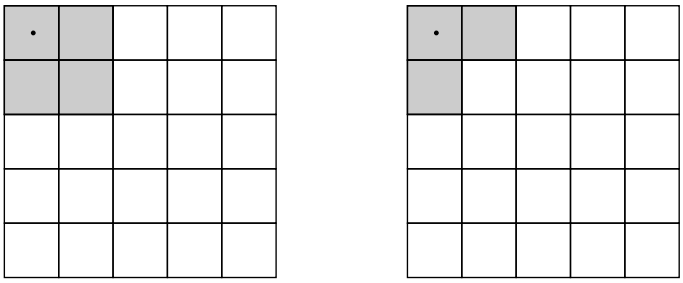


Figura 2.23: Elementos estruturais com origem nas bordas da imagem para (a) 4−conexo e (b) 8−conexo.

Em termos computacionais, a imagem de entrada deve ter o mesmo domínio, ou seja o mesmo tamanho, da imagem de saída. Por esse motivo, [21] opta por uma abordagem análoga a que propomos, ou seja, na Figura 2.24, onde podemos ver um SE centrado em p, apenas 6 de seus pixels fazem parte da imagem e são considerados no processamento. Por outro lado, [38], cita algumas estratégias alternativas. São elas:

(1) os pixels do SE que não pertencem à imagem podem ter tons de cinza definidos como sendo zero (preto), o que é equivalente a considerar uma camada externa extra de pixels de tons zero em torno da figura;

(2) os tons de cinza dos 3 pixels do SE externos à imagem são os mesmos tons dos três pixels da borda direita (hachurados na figura), consequentemente, também estamos acrescentando uma camada externa extra de pixels à imagem. Essa abordagem considera a imagem como se fosse um *toro*, pois as arestas exteriores paralelas da imagem são identificadas no processamento;

(3) os SEs que contêm pixels que não pertecem à imagem são desconsideradas no processamento, ou seja, não há processamento, sendo que o pixel p permanece inalterado.

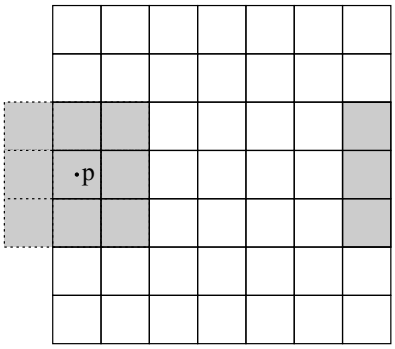


Figura 2.24: Processamento do pixel p utilizando a bola 8− conexa de raio 1.

2.4.2 Morfologia binária

Em uma imagem binária $f : \Omega \subset \mathbb{Z}^2 \longrightarrow \{0, 1\}$ os pixels brancos (de valor 1) definem completamente a imagem e os pixels pretos (de valor 0) são o fundo da imagem. Na Figura 2.25[7] temos um exemplo de imagem binária. Ressaltamos, porém, que algumas literaturas definem os pixels pretos como sendo os objetos da imagem. Diremos que os pixels brancos formam o primeiro plano da imagem ou *foreground* enquanto os pixels pretos formam o plano de fundo da imagem ou *background* (veja Figura 2.25).

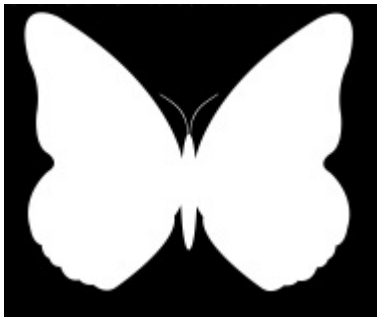


Figura 2.25: Exemplo de imagem binária.

Operações morfológicas em imagens binárias produzem imagens binárias. Assim, um processo morfológico se reduz a decidir quais pixels serão desligados e farão parte do fundo (valor 0) e quais serão ligados e farão parte dos objetos da imagem (valor 1). Nesta linha de raciocínio, dado um pixel, o SE é que determina quais dos pixels ao redor do dado pixel devem ser considerados na decisão de alterar ou não seu valor [60]. Nesse sentido, a escolha do SE é central no processamento morfológico.

A linguagem da morfologia matemática em imagens binárias é a teoria de conjuntos. Os conjuntos representam os objetos em uma imagem. Recordemos algumas definições que nos auxiliarão nessa seção.

A translação de um conjunto B por um ponto (vetor) x, denotada por B_x e definida por:

$$B_x = \{b + x \mid b \in B\}\,.$$

A translação será utilizada para posicionar o SE em local conveniente.

Uma reflexão de um conjunto B será denotada por $\breve{B}$ e definida por

$$\breve{B} = \{-b \mid b \in B\}\,.$$

Esta reflexão também pode ser vista como uma rotação de 180^o em torno da origem.

O complemento de um conjunto B será denotado por B^c e definido por

$$B^c = \{x \mid x \notin B\}\,.$$

[7] Imagem retirada de: http://br.stockfresh.com/image/1041280/silhouettes-of-butterflies

Agora podemos definir as operações de erosão e dilatação.

Chamaremos de $A \subset \Omega$ o conjunto das coordenadas dos pixels de valor 1 da imagem.

2.4.2.1 Erosão e Dilatação

A erosão de um conjunto A por um SE B, de origem $(0,0)$, denotada por $A \ominus B$, é definida por

$$A \ominus B = \{x \in A \mid B_x \subset A\}.$$

Isso significa que, para realizar a erosão em uma imagem binária, devemos examinar todos os pixels da imagem transladando o SE B para cada pixel de A e em seguida verificar quais são os pixels tais que B está contido em A. Se o SE estiver contido em A (valores 1) então essas posições continuarão valendo 1 na imagem final, caso contrário o pixel deve ter seu valor alterado para 0.

A dilatação de um conjunto A por um SE B, de origem $(0,0)$, denotada por $A \oplus B$, pode ser definida em termos de erosão da seguinte forma:

$$A \oplus B = \left(A^c \ominus \breve{B}\right)^c.$$

Observamos, entretanto, que

$$A \oplus B = \left(A^c \ominus \breve{B}\right)^c = \left(\left\{x \in A^c \mid \left(\breve{B}\right)_x \subset A^c\right\}\right)^c = \left\{x \in \Omega \mid \left(\breve{B}\right)_x \cap A \neq \varnothing\right\},$$

que é uma forma equivalente da definição de dilatação bastante útil para a implementação computacional.

Assim, para realizar a dilatação devemos inicialmente posicionar a origem do SE B rotacionado em cada pixel de Ω. As posições em que $\breve{B}$ intersectam A passarão a ter valor 1. Observamos que se posicionarmos a origem de $\breve{B}$ em qualquer ponto de A a interseção não será vazia. Dessa forma, todos os pontos de A farão parte da imagem dilatada bastando examinar apenas os pixels do *background*.

Se a origem do SE é um dos pixels do SE, então a erosão tem o efeito de "encolher" a imagem enquanto a dilatação tem o efeito de "expandir" a imagem. Com isso, a erosão apaga pixels que não atendem a um determinado padrão, enquanto a dilatação altera uma pequena vizinhança dos pixels para um dado padrão.

Observamos que, caso o SE B tenha um ponto central e seja simétrico em relação a ele (o que é comum), então $\breve{B} = B$.

Vejamos um exemplo básico de erosão binária e um de dilatação binária afim de compreendermos as operações.

Exemplo 1: Erosão binária

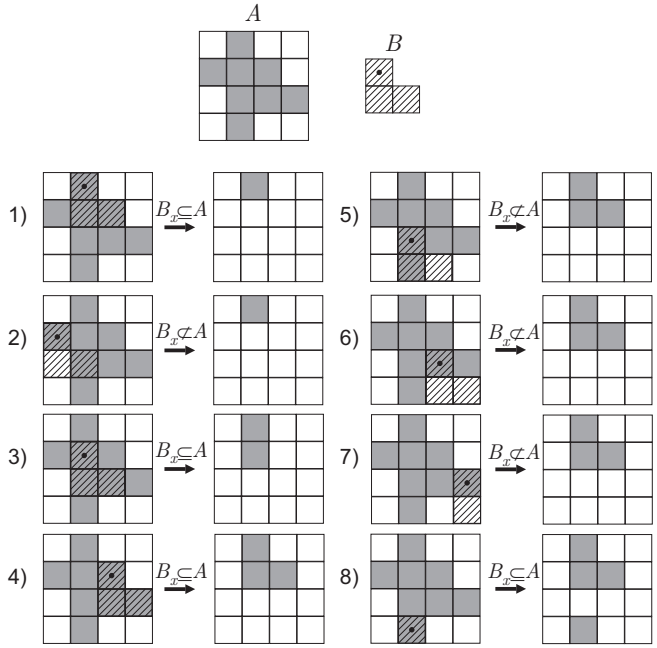


Figura 2.26: Exemplo de erosão binária.

A Figura 2.26 é um exemplo simples de erosão binária. Nessa figura, para uma melhor vizualização, os pixels de valor 1 estão pintados de cinza, os pixels de valor 0 estão pintados de branco e, o SE está hachurado para que possamos enxergar a sobreposição do SE B nos pixels de A. Como vimos, precisamos verificar apenas se os elementos que estão em A permanecerão em A. O pixel x permanecerá em A caso $B_x \subseteq A$. Observemos que com o elemento estrutural acima definido, a última linha de pixels da imagem sempre permanece inalterada.

Exemplo 2: Dilatação binária.

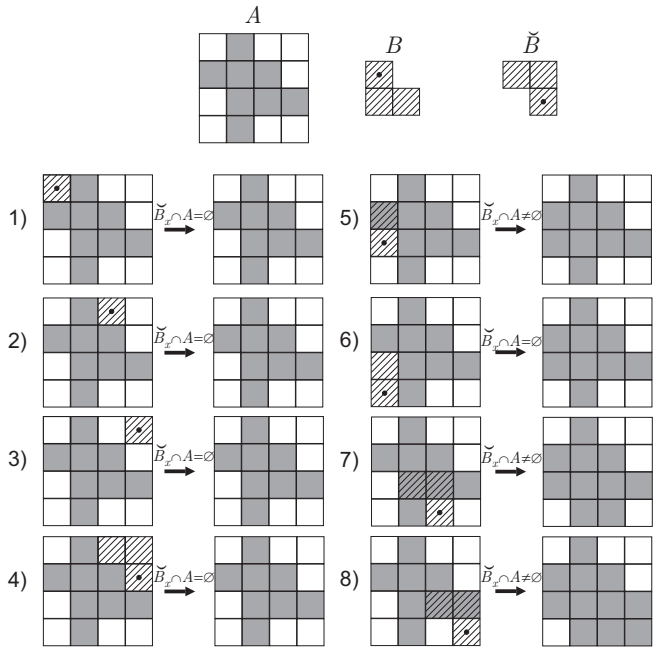


Figura 2.27: Exemplo de dilatação binária.

A Figura 2.27 nos mostra um exemplo de dilatação binária. Aqui, mais uma vez, os pixels de valor 1 estão pintados de cinza e os de valor 0 estão em branco. Como vimos, todos os pixels de A continuarão na imagem dilatada. Precisamos examinar apenas os pixels do *background* para verificar quais deles farão parte da imagem dilatada. E essa verificação é feita com base na interseção do SE B rotacionado com A.

2.4.2.2 Dualidade

Dado um operador Ψ o seu operador dual é denotado por Ψ^* e é definido por $\Psi^*(A) = (\Psi(A^c))^c$.

Dado o SE B, a erosão é dual da dilatação, ou seja, $\Psi^*(A) = A \ominus B$ e $\Psi(A') = A' \oplus \breve{B}$. De fato: tomemos a dilatação do conjunto A^c pelo SE $\breve{B}$ então temos, por definição, que $A^c \oplus \breve{B} = (A \ominus B)^c$. Tomando o complementar dos dois lados temos que $A \ominus B = \left(A^c \oplus \breve{B}\right)^c$.

A dualidade nos mostra que a erosão e a dilatação não são operações inversas. A erosão "encolhe" os objetos da imagem mas "expande" o background enquanto a dilatação faz o contrário. Conforme [59], não existe transformações inversas para a erosão e dilatação.

2.4.2.3 Abertura e Fechamento

Existem duas operações secundárias, porém não menos importantes, na morfologia matemática. Essas operações são definidas pela composição das operações básicas.

A erosão remove objetos da imagem que não contém o SE. Entretanto, algumas vezes é preciso recuperar algumas estruturas perdidas e, como vimos, não existe transformação inversa da erosão. Essa necessidade motivou a definição de abertura.

A abertura de um conjunto A por um SE B, denotada por $A \circ B$, é dada por

$$A \circ B = (A \ominus B) \oplus B.$$

Quando uma imagem é dilatada muitas vezes perde seu formato original. Para suprir essa necessidade temos a definição de fechamento. O fechamento de um conjunto A por um SE B, denotado por $A \bullet B$, é dado por

$$A \bullet B = (A \oplus B) \ominus B.$$

Na Figura 2.28 podemos ver um exemplo básico das operações de abertura e fechamento em uma imagem binária.

Geralmente a abertura suaviza o contorno das imagens, elimina saliências finas e quebra ligações estreitas. Por outro lado, o fechamento além de também suavizar contornos, funde descontinuidades estreitas, alonga golfos finos, elimina pequenos buracos

e preenche lacunas em um contorno [29]. As operações de abertura e fechamento são duais, assim como ocorre com a erosão e a dilatação.

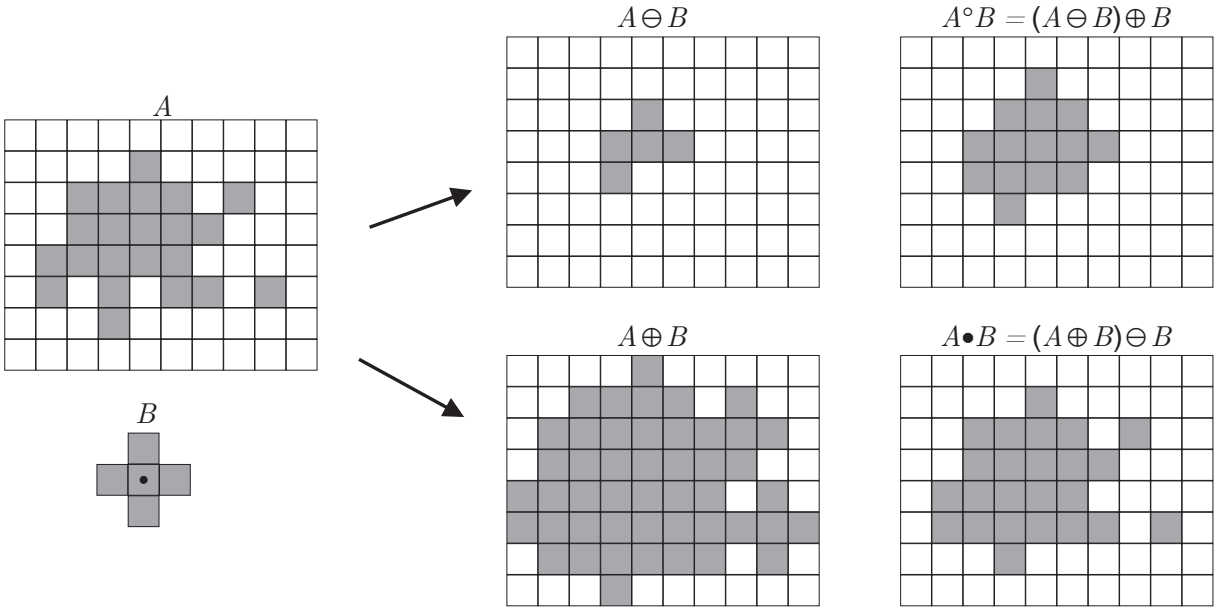


Figura 2.28: Exemplo de abertura e fechamento em uma imagem binária.

A Figura 2.29 ilustra um exemplo prático do uso da sequência abertura - fechamento para a redução de ruído na imagem de uma impressão digital.

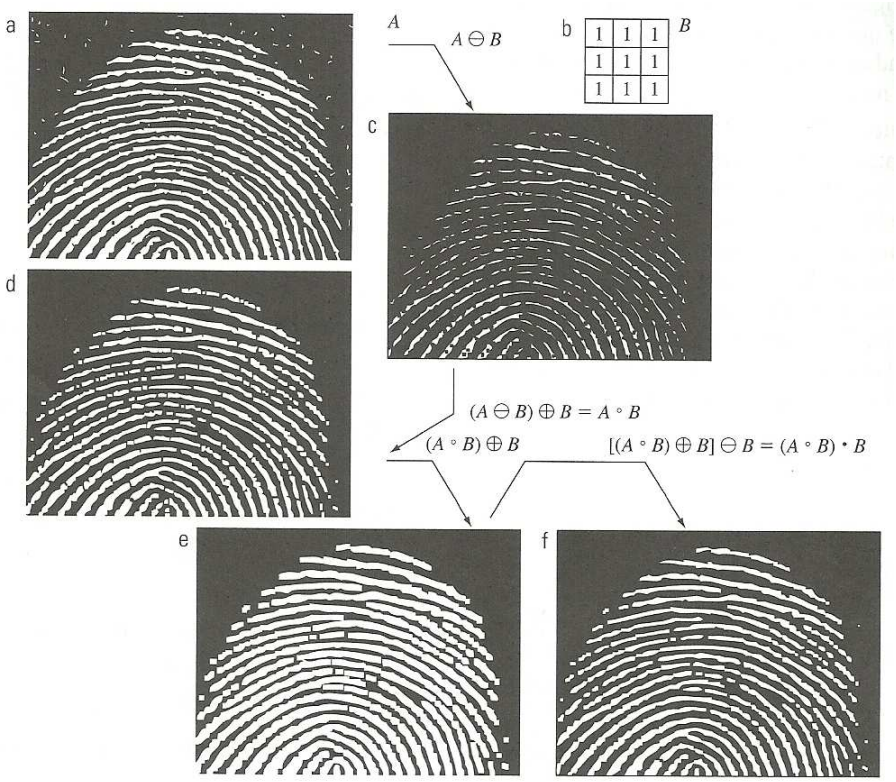


Figura 2.29: Redução de ruído de uma impressão digital [8].

[8] Imagem retirada de [29].

Alguns algoritmos morfológicos básicos de uso prático são: *extração de fronteiras*, *preenchimento de buracos*, *extração de componentes conexos*, *fecho convexos*, *afinamento*, *espessamento*, *esqueletos*, *poda* e *reconstrução morfológica*. Tais algoritmos podem ser encontrados em [29], [45], [21] ou [38].

2.4.3 Morfologia em tons de cinza

Vejamos agora como são definidos os operadores morfológicos para uma imagem f em tons de cinza. Os mesmos princípios básicos da morfologia binária podem ser utilizados. Neste trabalho, abordaremos apenas as operações de erosão e dilatação por elemento estrutural. Utilizaremos a mesma definição de SE vista acima.

Consideraremos, primeiramente, os casos em que a amostragem foi feita utilizando uma rede de pontos uniformemente distribuída.

Neste caso, o domínio da imagem é um subconjunto de $\mathbb{Z}^2$. Assim, definimos a translação da imagem f por um vetor b, denotada por f_b, como sendo $f_b(x) = f(x-b)$, desde que f esteja definida para $x-b$, ou seja, o valor da imagem transladada em um pixel x é igual ao valor da imagem original na posição transladada pelo vetor oposto $-b$.

Observamos que, por definição, uma imagem só pode ser transladada caso a rede de pontos da imagem transladada coincida com rede de pontos da imagem original [59]. Dessa forma, as definições que utilizam a translação só podem ser aplicadas em imagens com rede de pontos uniformemente distribuídas.

A erosão de uma imagem f por um SE B é denotada por $\varepsilon_B(f)$ e definida por

$$[\varepsilon_B(f)](x) = \min_{b \in B} f(x+b). \tag{2.2}$$

Outra forma equivalente de interpretar a definição acima é que a erosão pode ser dada pelo valor mínimo na região coincidente com B quando a origem de B está em x. Como toma-se o mínimo de cada região então espera-se que a imagem erodida fique mais escura do que a imagem original. Além disso, espera-se que o tamanho dos objetos claros sejam reduzidos e o tamanho dos objetos escuros sejam aumentados.

A dilatação de uma imagem f por um SE B, denotada por $\delta_B(f)$, é definida por

$$[\delta_B(f)](x) = \max_{b \in B} f(x-b). \tag{2.3}$$

As observações com relação à dilatação são parecidas com as da erosão. Como toma-se o máximo espera-se que a imagem fique mais clara. Espera-se ainda que o tamanho dos objetos claros aumentem enquanto o tamanho dos objetos escuros diminuem.

Análogo a funções binárias temos que a erosão e a dilatação são operadores duais.

As definições de abertura e fechamento para imagens em tons de cinza são inteiramente análogas às para imagem binárias.

A abertura de uma imagem f por um SE B, denotada por $\gamma_B(f)$, é definida por

$$\gamma_B(f) = \delta_B[\varepsilon_B(f)].$$

O fechamento de uma imagem f por um SE B, denotado pro $\varphi_B(f)$, é definido por

$$\varphi_B(f) = \varepsilon_B[\delta_B(f)].$$

Vejamos um exemplo básico de erosão e dilatação em tons de cinza. A Figura 2.30 nos mostra uma imagem f de 5×5 pixels (aumentada) e ao seu lado um gráfico com o tom de cinza de cada pixel na escala $[0, 255]$. Temos ainda representado o formato do SE B. Na segunda linha da Figura 2.30 temos o gráfico resultado da erosão $\varepsilon_B(f)$ e a imagem erodida de 5×5 pixels (aumentada). Na terceira linha temos, novamente, o gráfico e a imagem dilatada $\delta_B(f)$. Observamos que a imagem erodida tende a ficar mais escura enquanto a dilatada tende a ficar mais clara.

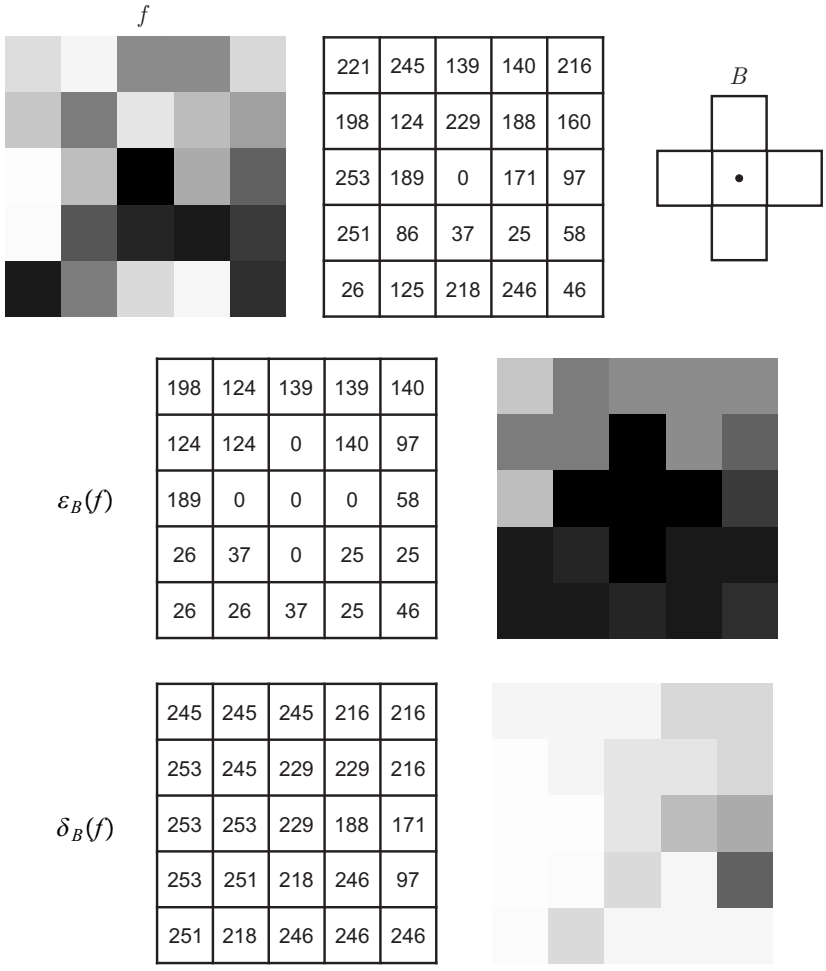


Figura 2.30: Exemplo simples de erosão e dilatação em tons de cinza.

A Figura 2.31 nos mostra um exemplo de abertura (b) e fechamento (c) de uma imagem radiográfica (a) de 448×425 pixels utilizando um SE 4−conexo de tamanho 3×3.

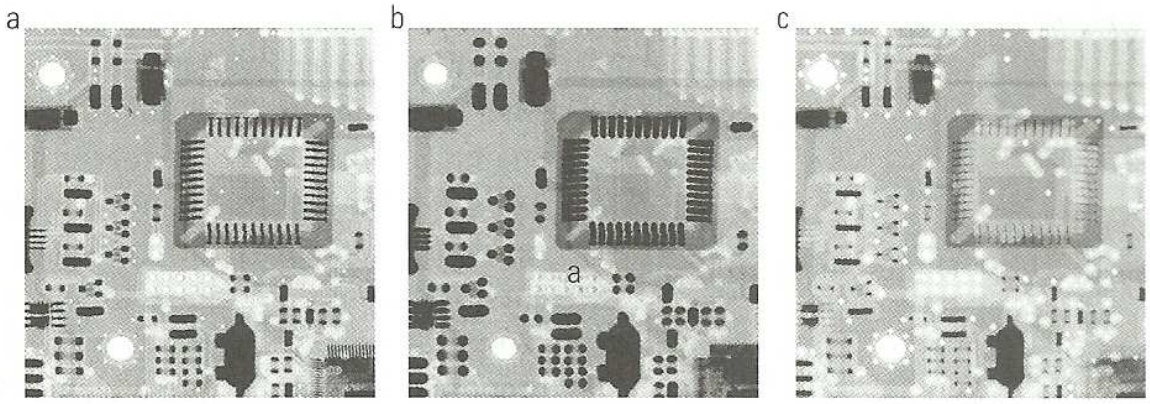


Figura 2.31: Exemplo de abertura e fechamento em uma imagem radiográfica[9].

As equações (2.2) e (2.3) que definem, respectivamente, a erosão e a dilatação em tons de cinza não valem quando a rede de pontos não é uniformemente distribuída. Entretanto, as definições de erosão e dilatação para esse caso podem ser obtidas considerando o grafo associado à rede de pixels [59]. Recordemos que $N_{\mathcal{G}}^{r}(v)$ é a notação para bolas centradas em v e raio r sobre o grafo $\mathcal{G}$.

Para cada vértice v de uma imagem f definida sobre um grafo $\mathcal{G}$ a definição de erosão de raio r é dada por

$$[\varepsilon_{\mathcal{G}}(f)](v) = \min\left\{f(v') \mid v' \in N_{\mathcal{G}}^{r}(v)\right\},$$

ou seja, toma-se o mínimo entre os valores dos tons de cinza dos elementos da bola.

Analogamente a dilatação de raio r é obtida considerando-se o máximo

$$[\delta_{\mathcal{G}}(f)](v) = \max\left\{f(v') \mid v' \in N_{\mathcal{G}}^{r}(v)\right\}.$$

Já vimos que podemos escolher o SE mais conveniente para cada aplicação. Entretanto, para imagens deste tipo, não temos mais a noção de formato e orientação, de forma que a vizinhança considerada nessas operações é simplesmente a vizinhança dada pelo grafo, ou seja, não existe a escolha do formato do SE.

2.4.4 Reticulados Completos

Como estamos interessados em trabalhar com espaços suporte de pixels com formatos diferentes dos apresentados nas seções acima e, também, queremos estudar

[9] Imagem retirada de [29].

métricas distintas nesses espaços, generalizaremos as definições de erosão e dilatação para os chamados reticulados completos.

Detalhes sobre essa subseção podem ser encontrados nas referências [57], [47] e [30].

Dado um conjunto não vazio $\mathcal{L}$, uma relação binária $\leqslant$ em $\mathcal{L}$ é chamada de ordenamento parcial quando para todo $x, y, z \in \mathcal{L}$ valem as seguintes propriedades:

$i)$ $x \leqslant x$

$ii)$ $x \leqslant y$ e $y \leqslant x \Longrightarrow x = y$

$iii)$ $x \leqslant y$ e $y \leqslant z \Longrightarrow x \leqslant z$

Um conjunto $\mathcal{L}$ munido da ordem parcial $\leqslant$ é chamado de conjunto parcialmente ordenado ou simplesmente *poset* e denotado por $(\mathcal{L}, \leqslant)$. Caso $x \leqslant y$ ou $y \leqslant x$ para quaisquer $x, y \in \mathcal{L}$ dizemos que $\mathcal{L}$ é totalmente ordenado.

Todo ordenamento parcial $(\mathcal{L}, \leqslant)$ tem um ordenamento dual correspondente $(\mathcal{L}, \leqslant')$, chamado poset dual, definido por $x \leqslant' y$ quando $x \geqslant y$. Observamos que $(\leqslant')'$ coincide com $\leqslant$.

Dado um poset $(\mathcal{L}, \leqslant)$ e um subconjunto $\mathcal{K}$ de $\mathcal{L}$:

- $a \in \mathcal{K}$ é chamado de menor elemento de $\mathcal{K}$ quando $a \leqslant x$ para todo $x \in \mathcal{K}$;

- $b \in \mathcal{K}$ é chamado de maior elemento de $\mathcal{K}$ quando $b \geqslant x$ para todo $x \in \mathcal{K}$;

- $a \in \mathcal{L}$ é chamado de limitante inferior de $\mathcal{K}$ quando $a \leqslant x$ para todo $x \in \mathcal{K}$

(note que a não precisa pertencer a $\mathcal{K}$)

O maior limitante inferior a_0 de $\mathcal{K}$ é chamado de ínfimo de $\mathcal{K}$, denotado por $\inf \mathcal{K}$ ou $\wedge \mathcal{K}$, satisfaz:

(i) $a_0 \leqslant x$ para todo $x \in \mathcal{K}$;

(ii) $a \leqslant a_0$ para qualquer outro limitante inferior a de $\mathcal{K}$.

As noções de limitante superior e supremo, denotado por $\sup \mathcal{K}$ ou $\vee \mathcal{K}$, são definidas de forma análoga. Na verdade, ínfimo e supremo são noções duais pelo princípio de dualidade [30].

Caso o supremo ou o ínfimo sejam calculados para uma família de índices podemos utilizar as notações:

$$\bigvee_{i \in I} x_i \text{ ou } \bigvee \{x_i \mid i \in I\} \text{ e } \bigwedge_{i \in I} x_i \text{ ou } \bigwedge \{x_i \mid i \in I\}$$

Um poset $(\mathcal{L}, \leqslant)$ é chamado de **reticulado** se todo subconjunto finito de $\mathcal{L}$ tem um ínfimo e um supremo. Um reticulado é dito **completo** se todo subconjunto de $\mathcal{L}$ tem um ínfimo e um supremo.

Todo poset totalmente ordenado é um reticulado, porém esse reticulado pode não ser completo.

Por definição, qualquer reticulado completo não vazio é limitado pois contém seu ínfimo, denotado por $O = \wedge\mathcal{L}$, e seu supremo, denotado por $I = \vee\mathcal{L}$.

Dados dois reticulados completos $\mathcal{L}$, $\mathcal{M}$, os operadores $\varepsilon : \mathcal{L} \to \mathcal{M}$ e $\delta : \mathcal{L} \to \mathcal{M}$ são chamados, respectivamente, de erosão e dilatação quando

$$\varepsilon\left(\wedge x_i\right) = \wedge\varepsilon\left(x_i\right), \ \forall x_i \in \mathcal{L}$$
$$\delta\left(\vee x_i\right) = \vee\delta\left(x_i\right), \ \forall x_i \in \mathcal{L}.$$

Os operadores de erosão e dilatação são crescentes, isto é, $x \leqslant y \Longrightarrow \varepsilon\left(x\right) \leqslant \varepsilon\left(y\right)$ e $\delta\left(x\right) \leqslant \delta\left(y\right)$.

Proposição: [57] Dados dois reticulados completos $\mathcal{L}$ e $\mathcal{M}$ a erosão $\varepsilon : \mathcal{L} \to \mathcal{M}$ e a dilatação $\delta : \mathcal{L} \to \mathcal{M}$ estão relacionadas pela chamada adjunção de Gallois dada por

$$\delta\left(x\right) \leqslant y \Longleftrightarrow x \leqslant \varepsilon\left(y\right), \quad x, y \in \mathcal{L}.$$

Cada dilatação δ define uma única erosão dada por

$$\varepsilon\left(x\right) = \vee\left\{b \in \mathcal{L} : \delta\left(b\right) \leqslant x\right\}$$

e cada erosão ε define uma única dilatação dada por

$$\delta\left(x\right) = \wedge\left\{b \in \mathcal{L} : \varepsilon\left(b\right) \geqslant x\right\}.$$

Dados os operadores ε e δ os operadores de abertura e fechamento são definidos, respectivamente, como $\gamma\left(x\right) = \delta\left(\varepsilon\left(x\right)\right)$ e $\varphi\left(x\right) = \varepsilon\left(\delta\left(x\right)\right)$.

Um poset $(\mathcal{L}, \leqslant)$ no qual todo subconjunto admite um ínfimo, mas não necessariamente um supremo, é chamado de inf semirreticulado completo. Para esse caso temos os seguintes resultados [7]:

(i) dada uma erosão ε é sempre possível definir uma abertura γ por $\gamma\left(x\right) = \wedge\left\{y \in \mathcal{L} : \varepsilon\left(x\right) \leqslant \varepsilon\left(y\right)\right\}$;

(ii) mesmo que a dilatação δ não esteja bem definida para todo $x \in \mathcal{L}$, δ sempre está bem definida em $\varepsilon\left(\mathcal{L}\right)$;

(iii) $\gamma = \delta(\varepsilon)$

O fechamento $\varphi = \varepsilon(\delta)$ está apenas parcialmente definido.

Exemplos:

(a) A reta $\mathbb{R}$ com a ordenação $\leqslant$ usual é um ordenamento completo. Se considerarmos o ínfimo e o supremo usuais na reta temos que $\mathbb{R}$ é um reticulado, porém não é completo. Enquanto, $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ com as mesmas operações é um reticulado completo.

(b) Dado um conjunto E qualquer, o conjunto das partes $\mathcal{P}(E)$ com a relação usual de inclusão de conjuntos é parcialmente ordenado. Considerando como ínfimo a interseção de conjuntos e como supremo a união de conjuntos temos que $\mathcal{P}(E)$ é um reticulado completo.

3 Espaços Suporte Circulares

Neste capítulo desenvolveremos duas propostas originais de espaços suporte que podem ser utilizadas com qualquer escala de tons de cinza ou espaços de cor. Parte do desenvolvimento deste capítulo foi previamente publicada no artigo [50]. Os algoritmos por nós desenvolvidos e apresentados neste trabalho foram implementados em linguagem *Python* (referências [22], [39] e [37]). Dentre os inúmeros comandos e notações do *Python* antecipamos algumas que serão utilizadas no decorrer desse capítulo:

- $round\,(x)$ é o inteiro mais próximo do número real x.

- $len(A)$ é o número de colunas do vetor linha A;

- $A[i]$ é a *i-ésima* entrada do vetor linha A (a numeração dos vetores e matrizes se inicia no 0).

3.1 Propostas de espaços suporte

Sabemos da Geometria Diferencial que superfícies esféricas não podem ser planificadas isometricamente com a métrica euclidiana induzida. Intuitivamente, quando tentamos imaginar tal planificação, é como se "faltasse área" na esfera ou "sobrasse área no plano". Entretanto, podemos pensar em planificações de superfícies esféricas em planos por meio de projeções, naturalmente não isométricas, como por exemplo, projeções cilíndricas ou cônicas. Visualmente, em tais projeções, algumas partes da imagem plana tem maior nitidez enquanto outras concentram muita informação em pouca área. Observamos, por exemplo, uma imagem da Lua na Figura 3.1[1].

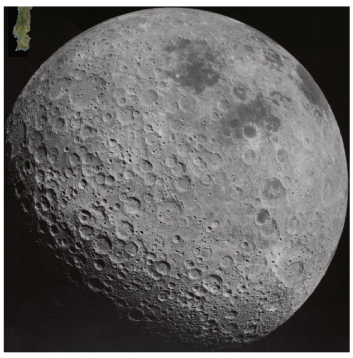


Figura 3.1: A Lua é um exemplo da classe de imagens que estamos trabalhando.

Nesta imagem vemos que o meio da Lua está nítido, podemos inclusive contar a quantidade de crateras. Entretanto, as bordas da imagem da Lua não estão nítidas,

[1] Imagem retirada de http://www.vox.com/2015/8/5/9100625/far-side-moon-nasa.

pois uma área grande da superfície da Lua foi planificada em pouco espaço, compactando informação.

A nossa proposta de trabalho consiste em apresentar modelos matemáticos para imagens de objetos esféricos que permitam melhorar a qualidade das imagens nos pontos próximos aos bordos circulares, bem como diminuir seu tamanho em termos de bytes para armazenagem. Contemplando esse objetivo, todas as imagens consideradas serão quadradas com o círculo que representa a esfera tangenciando as bordas da imagem (como na Figura 3.1). Como nosso interesse é a superfície esférica, vamos desprezar as partes da imagem que não correspondem à superfície e nos concentraremos no círculo inscrito à imagem.

Como já existe a perda de informação na planificação da imagem não desejamos permitir mais perdas utilizando baixa resolução para representar a imagem. Porém, ao utilizar muitos pixels para representar a imagem estamos sendo redundantes no meio da mesma, pois ali não precisamos de muito refinamento. Dessa forma, estamos em um impasse. Para abordar essa situação, estamos propondo novos espaços suporte circulares de pixels mantendo o aspecto visual da borda da imagem e utilizando menos pixels para representar a parte central da imagem.

Temos duas possibilidades justificadas pelas informações citadas no Capítulo 2. A primeira seria a construção de um chip CCD com a geometria que estamos propondo, pois, como vimos, a tecnologia atual contempla novos modelos já existindo, inclusive, alguns chips com características parecidas, porém com maior refinamento no centro, e não nos bordos como estamos propondo. Além disso, vimos que a evolução das telas de monitor permite a exibição de distribuições diferenciadas de pixels. Entretanto, esse tipo de engenharia foge ao escopo deste trabalho. A segunda opção, que será por nós desenvolvida, é propor uma nova amostragem adaptativa, pois de um modo geral, os chips de captura e monitores de exibição de imagens não são circulares.

Conforme [35], quando se trabalha com objetos redondos os resultados podem não ser satisfatórios se utilizarmos métricas e amostragens discretas, parecidas com a euclidiana. Nessa linha de raciocínio, já foi sugerido inclusive, que tais imagens fossem transformadas para outros espaços suporte que se adaptem melhor a natureza do objeto. Dessa forma, neste trabalho utilizaremos uma amostragem adaptativa a partir de uma imagem plana que suporemos ter, inicialmente, uma quantidade $P \times P$ ($P \geqslant 7$) suficiente (grande) de pixels para preservar as bordas da imagem.

A nossa proposta é construir modelos que utilizem coordenadas polares, que são naturais em um disco. Com esse objetivo, consideraremos que o centro dos nossos modelos encontram-se na posição $\mathcal{O} = \left(\frac{P-1}{2}, \frac{P-1}{2}\right)$, ou seja, exatamente no meio da imagem (recordemos, na Figura 2.14, como estamos considerando as coordenadas dos

pixels no modelo Ω_T, definido no Capítulo 2). Além disso, consideraremos, a partir de $\mathcal{O}$, círculos concêntricos e segmentos de raios que dividirão esses círculos para formar os *pixels polares* dos nossos modelos, que serão chamados apenas de *pixels* quando não houver confusão. Esses segmentos de raios começarão em alguma das circunferências concêntricas e terminarão no bordo do disco. Chamaremos a circunferência de raio $\frac{P}{2}$, que delimita a imagem, de C. Além disso, chamaremos de primeira coroa o espaço entre a primeira circunferência (sem divisões) e a segunda circunferência, a segunda coroa será o espaço entre a segunda e a terceira circunferência, e assim por diante.

Para fazer a quantização dos espaços suporte que proporemos, iremos sobrepor o espaço Ω_T e os que estamos propondo e definiremos os tons de cinza de cada região delimitada pelos pixels dos novos espaços suporte com a média aritmética dos tons de cinza dos pixels de Ω_T presentes em seu interior (Figura 3.2). Nesse sentido é interessante que os pixels dos novos espaços suporte não sejam demasiadamente estreitos (distância radial pequena) nem finos (distância angular pequena), pois nesses casos utilizaríamos o mesmo tom de cinza em vários pixels dos novos espaços suporte já que eles sobreporiam o mesmo pixel de Ω_T.

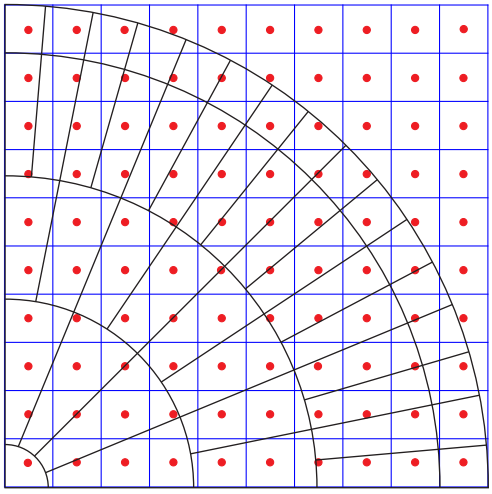


Figura 3.2: Proposta de quantização.

Em princípio, como nosso interesse é apenas na esfera, desprezaremos os bordos da imagem que correspondem a aproximadamente 21.5% da quantidade total de pixels da imagem, pois a área da circunferência corresponde a $\frac{\pi\left(\frac{P}{2}\right)^2}{P^2} = \frac{\pi}{4} \approx 0.785$ da área do quadrado. Além disso, gostaríamos de reduzir a quantidade de pixels no meio, onde a imagem é mais nítida. Dessa forma, trabalharemos com uma parcela da quantidade $0.785P^2$ de pixels iniciais. Chamaremos de p_r (onde $p_r \in [0,1]$) o parâmetro que fornece a porcentagem dos pixels iniciais com os quais trabalharemos. *Propomos, nos nossos algoritmos, utilizar* $p_r = 0.8$*, entretanto, ressaltamos que tal parâmetro pode ser alterado nos algoritmos.* Para o parâmetro escolhido, geraremos espaços suporte com, aproximadamente

$(0.785)\,(0.8)\,P^2 = (0.628)\,P^2$ pixels polares. Assim, a imagem nos modelos propostos terão, aproximadamente, apenas 62.8% dos pixels iniciais.

Outra observação importante é que, como não queremos perder informação do bordo, é importante que a diferença entre os raios da circunferência C e a penúltima circunferência (c_j na Figura 3.3) seja 1, pois na representação que estamos utilizando em Ω_T o pixel tem "comprimento" euclidiano 1. Além disso, gostaríamos que a divisão radial na última coroa fosse de tal forma que os pixels polares dessa coroa se assemelhem aos pixels em Ω_T. Para isso, consideremos o ângulo central α conforme Figura 3.3. Gostaríamos que o arco de circunferência determinado por α na circunferência C seja, aproximadamente 1. Dessa forma, se tivermos x pixels polares em um quarto da coroa (ou em um dos quadrantes da imagem), então cada pixel será determinado por um arco em C de tamanho $\frac{1}{x}\frac{2\pi\left(\frac{P}{2}\right)}{4} = \frac{\pi P}{4x}$. Como gostaríamos que esse tamanho fosse aproximadamente 1, então devem haver aproximadamente $x \approx \frac{\pi P}{4}$ pixels em um quarto de coroa, ou seja, aproximadamente πP pixels na última coroa. Observamos que, com as imposições feitas, independente de como ficará a divisão no meio da imagem, a última coroa já está definida, pois sua largura e suas divisões já estão determinadas. Além disso, optamos por não fazer divisões na circunferência de menor raio para evitar os pixels finos.

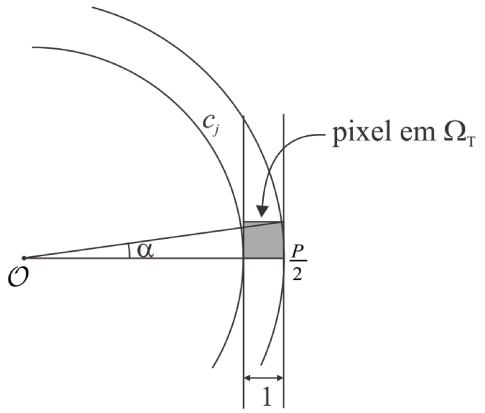


Figura 3.3: Determinação da última coroa.

Após diversas tentativas de formato de distribuição de pixels no disco (ver Apêndice A) encontramos duas distribuições que se mostraram satisfatórias. Durante essas tentativas percebemos que a quantidade de pixels deve aumentar gradativamente à medida que nos aproximamos do bordo. Por isso, criamos um espaço suporte constituído de k camadas, onde cada camada tem várias coroas. A divisão radial é a mesma em todas as coroas da camada e, ao passar de uma camada interna para uma camada imediatamente externa, a quantidade de pixels por coroa dobra. A primeira camada terá coroas de n pixels cada, onde $n > n_{min}$. O parâmetro n_{min} pode ser escolhido conforme a aplicação. Além disso, a quantidade de coroas em cada camada aumenta à medida que nos aproximamos do bordo conforme o vetor $A = [a_1, a_2, ..., a_k]$ (com $a_i < a_j$ se $i < j$). Dessa forma, a primeira

camada será constituída de a_1 coroas de $2^0 n$ pixels. A segunda camada terá a_2 coroas de $2^1 n$ pixels. A terceira camada terá a_3 coroas de $2^2 n$ pixels, a quarta camada terá a_4 coroas de $2^3 n$ pixels, e assim por diante.

Nos algoritmos deste trabalho optamos por tomar o vetor A como sendo $A = [1, mc_1, 2mc_1, ..., (k-1)mc_1]$, onde m é o parâmetro que permitirá controlar o aumento de coroas por camadas e poderá ser alterado conforme a necessidade da aplicação, enquanto c_1 será calculado de forma a aproximar quantidade inicial de pixels.

O algoritmo que determina os parâmetros das distribuições de pixels para nossos modelos é dado por:

Algoritmo de Subdivisões:

Entrada: Imagem em Ω_T de $P \times P$ pixels ($P \geqslant 7$), p_r, m e n_{min}.

1^o) Calcularemos, conforme explicamos anteriormente, a quantidade de pixels de cada coroa da última camada: $x = round\,(\pi P)$.

2^o) De acordo com o padrão citado anteriormente, a cada mudança de coroa subsequente, devemos duplicar a quantidade de pixels radiais. Assim, precisamos reverter o processo. Para isso, dividimos x por 2. Tomamos o resultado e dividimos por 2. Repetimos o processo enquanto o resultado da divisão for maior do que n_{min}. Chamaremos de k_1 a quantidade de divisões até satisfazer a condição, ou seja, k_1 é o maior inteiro tal que $\dfrac{x}{2^{k_1}} > n_{min}$.

3^o) O número $\dfrac{x}{2^{k_1}}$ é, então, o número de pixels que deverá ter a primeira coroa. Como esse número pode não ser inteiro tomamos $n = round\left(\dfrac{x}{2^{k_1}}\right)$.

4^o) Tomemos o vetor $B = \left[n, 2n, 2^2 n, ..., 2^{k_1} n\right]$. A quantidade $k = k_1 + 1$ de entradas significa quantas camadas terá o espaço suporte (exclui-se a circunferência central sem divisões).

5^o) A quantidade de pixels da imagem será

$$1 + AB^t = 1 + n + (2n)\, mc_1 + \left(2^2 n\right)(2mc_1) + \cdots + \left(2^{k-1} n\right)(k-1)\, mc_1.$$

Gostaríamos, conforme proposta inicial, que esse valor se aproximasse de $0.785 p_r P^2$. Assim, $c_1 = round\left(\dfrac{0.785 p_r P^2 - 1 - n}{\sum_{i=1}^{k-1} 2^i mni}\right)$.

As regiões dadas pela distribuição acima serão os *pixels polares* dos espaços suporte. Assim, temos um algoritmo que determina as variáveis para a distribuição de pixels em uma imagem, com exceção do comprimento dos raios.

Para facilitar a escrita dos demais algoritmos consideremos o vetor

$$k_{aux} = \{n, \underbrace{2n, \cdots, 2n}_{mc_1 \text{ vezes}}, \underbrace{4n, \cdots, 4n}_{2mc_1 \text{ vezes}}, \cdots, \underbrace{2^{k-1}, \cdots, 2^{k-1}}_{(k-1)mc_1 \text{ vezes}}\}, \tag{3.1}$$

onde a entrada $2n$ se repete mc_1 vezes, a entrada $4n$ se repete $2mc_1$ vezes, e assim por diante até que a entrada $2^{k_1}n$ se repita $(k-1)\,mc_1$ vezes. Dessa forma, $len(k_{aux})$ é a quantidade de coroas que a imagem possui. Denotaremos por $c = len(k_{aux}) + 1$.

Nos nossos algoritmos optamos por utilizar $p_r = 0.8$, $n_{min} = 10$ e $m = 1$. Na subseção 3.1.4 mostraremos alguns exemplos abordando a influência desses parâmetros nos espaços suportes propostos.

Tomemos como exemplo uma imagem que em Ω_T tenha 50×50 pixels, ou seja, $P = 50$:

$1^o)$ $x = round\,(\pi 50) = 157$

$2^o)$ $\dfrac{157}{2} = 78.5 > 10 \Longrightarrow \dfrac{78.5}{2} = 39.25 > 10 \Longrightarrow \dfrac{39.5}{2} = 19.625 > 10 \Longrightarrow \dfrac{19.625}{2} = 9.8125 \ngtr 10$. Logo, $k_1 = 3$ e $k = 4$.

$3^o)$ $n = round\left(\dfrac{157}{2^3}\right) = 20$

$4^o)$ $\left[20, 2\,(20)\,, 2^2\,(20)\,, 2^3\,(20)\right] = [20, 40, 80, 160]$

$5^o)$ $c_1 = round\left(\dfrac{0.628\,(50)^2 - 1 - 20}{\sum\limits_{i=1}^{k-1} 2^i\,(20i)}\right) = 2$

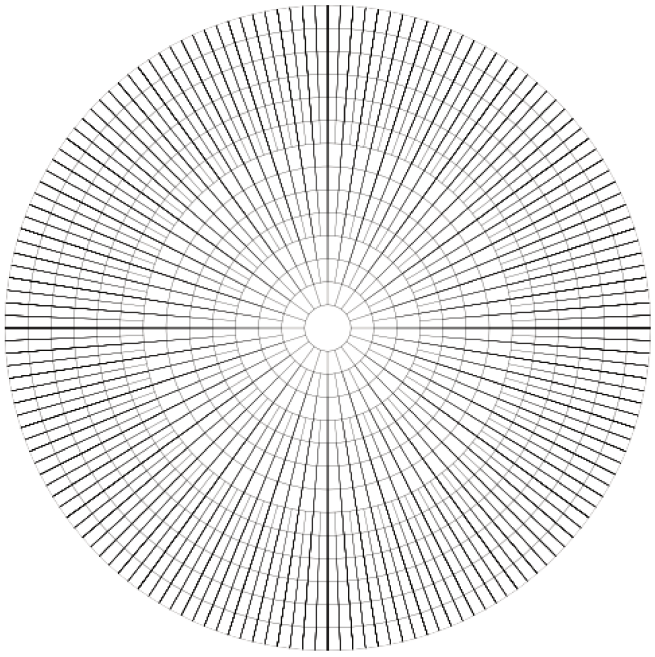


Figura 3.4: Resultado do Algoritmo de Subdivisões para $P = 50$ utilizando a métrica euclidiana.

Assim, temos uma circunferência inicial sem divisões e 4 camadas sendo a primeira com uma coroa de 20 pixels, a segunda camada com 2 coroas de 40 pixels cada,

a terceira camada com 4 coroas de 80 pixels cada e a quarta camada com 6 coroas de 160 pixels cada. Na Figura 3.4 podemos verificar a divisão que propomos. Entretanto, utilizamos os raios igualmente espaçados euclidianamente (o que não satisfaz as hipóteses que propomos para os nossos modelos) a fim de ilustrar o exemplo.

3.1.1 Notação

Denotaremos os pixels por $Q_{i,j}$, onde i é a coroa no qual se encontra o pixel e $j+1$ é a posição na coroa, girando no sentido anti-horário (ver Figura 3.5 (a)). Observamos ainda que a rede de pontos determinada pelos centros das regiões que delimitam $Q_{i,j}$ não é uma rede uniformemente distribuída.

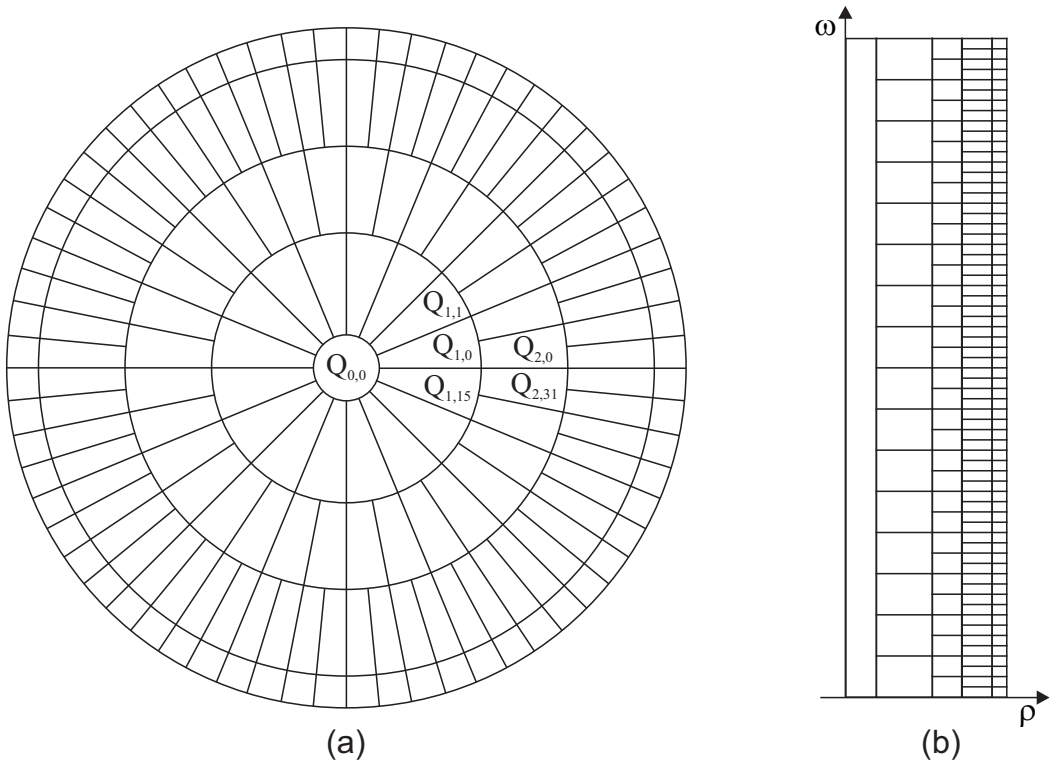


Figura 3.5: (a) Localização dos pixels $Q_{i,j}$ nos nossos modelos, (b) representação do espaço suporte de (a) no plano $\rho\omega$.

As referências [2], [36], [35], [56] e [42] apresentam uma representação de imagem adequada para imagens que contêm algum tipo de simetria radial. As coordenadas cartesianas são transformadas em coordenadas polares e representadas no plano $\rho\omega$ de tal forma que a coordenada radial ρ é representada no eixo horizontal e a coordenada angular ω, no eixo vertical. Conforme [35], essa representação apresenta algumas características interessantes. A periodicidade da coordenada angular faz com que rotações no plano cartesiano se tornem translações verticais no plano $\rho\omega$. Além disso, mudanças de escala da imagem se tornam translações horizontais em $\rho\omega$. Porém, essa representação é bastante dependente da escolha do centro da imagem. A Figura 3.6 exemplifica esta representação.

Como os espaços suportes que estamos propondo se assemelham às coordenadas polares, utlizaremos esta idéia para representarmos os pixels. Assim, o espaço suporte respresentado na Figura 3.5 (a) se transformaria, no plano $\rho\omega$, na Figura 3.5 (b).

Trabalharemos agora no sentido de definir uma maneira de calcular os raios das circunferências dadas pelo Algoritmo de Subdivisões de forma a satisfazer as hipóteses propostas.

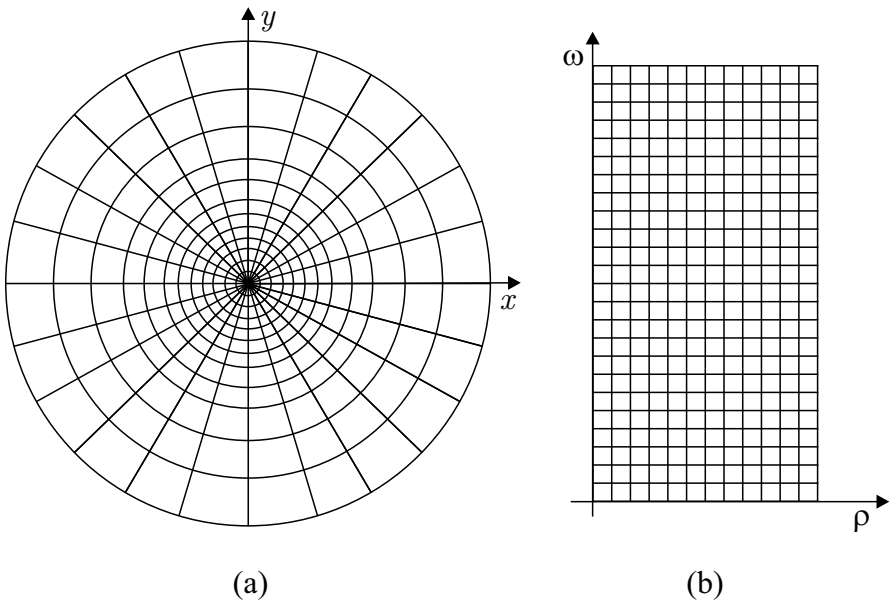


Figura 3.6: Representação dos pixels no plano euclidiano xy e no plano $\rho\omega$.

3.1.2 Espaço Suporte Hiperbólico Ω_H

Nesta abordagem utilizaremos a métrica hiperbólica no modelo do Disco de Poincaré $\mathcal{D}_R$ de raio R e curvatura gaussiana K apresentada no Capítulo 1. Recordemos que, nessa métrica, a distância entre os pontos $(0,0)$ e $(x,0)$ é dada por

$$\rho^*\left((0,0),(x,0)\right) = \frac{1}{\sqrt{-K}}\ln\left(\frac{R+x}{R-x}\right).$$

Fazendo $z = \rho^*\left((0,0),(x,0)\right)$ então $z\sqrt{-K} = \ln\left(\frac{R+x}{R-x}\right) \Longrightarrow x = R\frac{e^{z\sqrt{-K}}-1}{e^{z\sqrt{-K}}+1}$. Dessa forma, supondo que a medida hiperbólica do raio da circunferência C é c (obtido no Algoritmo de Subdivisões) então para fixar a distância 1 na última coroa podemos trabalhar com as variáveis R e K a fim de satisfazer essa condição, ou seja, gostaríamos de encontrar R e K tal que

$$\begin{cases} R\dfrac{e^{c\sqrt{-K}}-1}{e^{c\sqrt{-K}}+1} = \dfrac{P}{2} \\ R\dfrac{e^{(c-1)\sqrt{-K}}-1}{e^{(c-1)\sqrt{-K}}+1} = \dfrac{P}{2}-1 \end{cases} \tag{3.2}$$

Isolando R na primeira equação temos que $R = \frac{P}{2}\,\frac{e^{c\sqrt{-K}}+1}{e^{c\sqrt{-K}}-1}$. Fazendo $X = \sqrt{-K}$ e substituindo na segunda equação temos que $e^{X(2c-1)} - (P-1)\,e^{cX} + (P-1)\,e^{(c-1)X} - 1 = 0$. Fazendo $y = e^X$ temos $y^{2c-1} - (P-1)\,y^c + (P-1)\,y^{c-1} - 1 = 0$. Assim, encontrar R e K se torna um problema de encontrar soluções de um polinômio. Encontrada a solução do polinômio teremos que as medidas $r_i = r_h\,(i)$ dos raios das circunferências serão dadas por

$$r_h\,(i) = R\frac{e^{i\sqrt{-K}}-1}{e^{i\sqrt{-K}}+1}$$

Chamaremos o espaço suporte com a distribuição de pixels dada pelo Algoritmo de Subdivisões e com as medidas dos raios dadas por r_h de Espaço Suporte Hiperbólico e denotaremos por Ω_H.

Há vantagens na utilização da métrica hiperbólica na determinação dos raios em nosso modelo, pois como vimos no Capítulo 1, a imersão do plano polar em um disco de raio R conduz naturalmente à métrica hiperbólica, ou seja, estamos enxergando um plano ilimitado em todas as possíveis direções radiais em um disco de raio euclidiano finito, de tal modo que eixos polares rotacionados são mapeados em segmentos euclidianos que tornam-se semirretas com a métrica hiperbólica. Uma das vantagens de tal abordagem é a possibilidade de considerar distribuições radiais com quantidades infinitas de pixels, cada um deles com coordenadas inteiras, o que facilita muito a implementação computacional do modelo. Além disso, como a imersão do plano polar no disco é conforme, não há distorções angulares da imagem. *Ressalta-se também que devido ao fato da métrica hiperbólica tratar o bordo do disco como infinito, é possivel fazer distribuições radiais de quantidades tão grande quanto se queira de pixels de modo bastante homogêneo à medida que nos aproximamos do bordo, bastando para isso considerar curvaturas negativas adequadas.* Na seção 5.2, por exemplo, utilizamos $K = -5.760867571133922 \times 10^{-7}$.

No caso de $P = 50$ temos que $R = 32.16023452492$ e $K = -0.022016533691046$. E, com esses valores, temos que os raios $r_h\,(i)$ das 14 circunferências que compõe o espaço são:

i	1	2	3	4	5	6	7
$r_h\,(i)$	2.38	4.73	7.04	9.27	11.41	13.43	15.34

i	8	9	10	11	12	13	14
$r_h\,(i)$	17.12	18.76	20.27	21.64	22.88	24	25

onde nas primeiras linhas são dados os valores de i e nas segundas linhas os valores de $r_h\,(i)$.

Observamos que, nesse caso, a largura da última coroa é 1 conforme as hipóteses que impomos. O espaço suporte para este exemplo é apresentado na Figura 3.7.

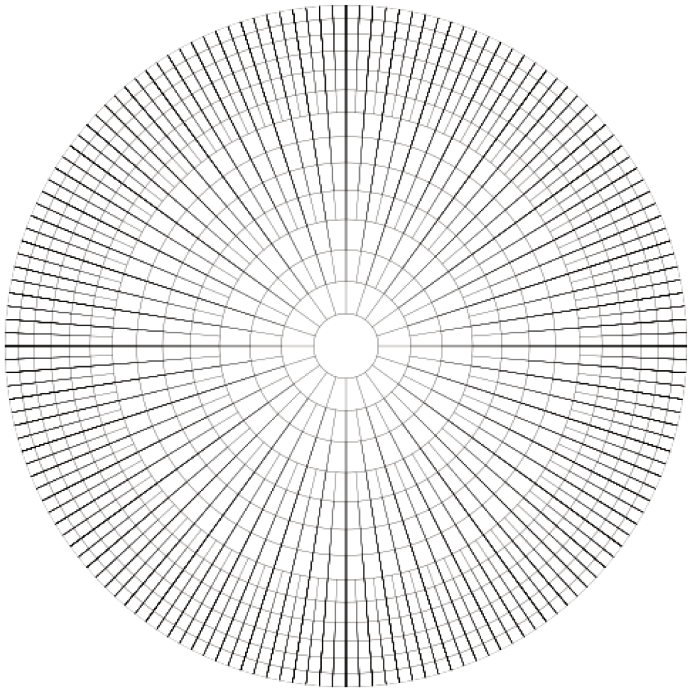


Figura 3.7: Espaço suporte Ω_H para $P = 50$.

3.1.3 Espaço Suporte Elíptico Ω_E

Como estamos tratando de imagens que são oriundas de objetos esféricos é interessante levar em consideração a relação entre a área dos pixels na imagem e a área desses pixels projetados no objeto esférico ao se determinar os raios. Fizemos alguns testes de projeções que podem ser encontrados no Apêndice A.

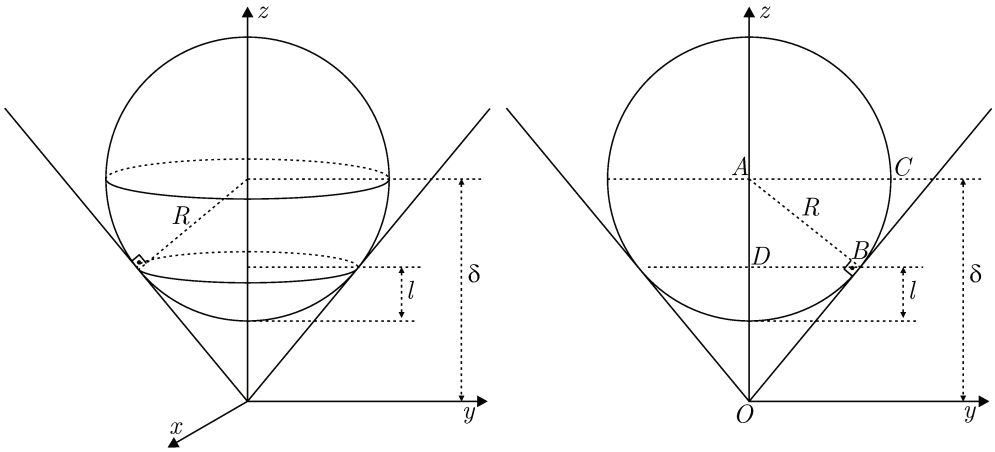


Figura 3.8: Esquematização da calota esférica visualizada pelo observador.

Observamos que, quando vemos (ou fotografamos) uma esfera nós nunca conseguimos visualizar um hemisfério inteiro da esfera. Dessa forma, para que as medidas

de área sejam fiéis ao original devemos saber qual o tamanho da calota que estamos visualizando e essa informação está instrinsicamente ligada à distância entre o objeto e o observador e ao raio do objeto.

Consideremos uma esfera de raio R e um observador situado em um ponto O distante δ do centro A dessa esfera. Queremos determinar a altura l da calota visível ao observador. Observamos, na Figura 3.8, que os triângulos ADB e ABO são semelhantes, donde $\frac{R}{\delta} = \frac{AD}{R}$. Daí, $AD = \frac{R^2}{\delta}$. Portanto, $l = R - AD = \frac{R}{\delta}(\delta - R)$.

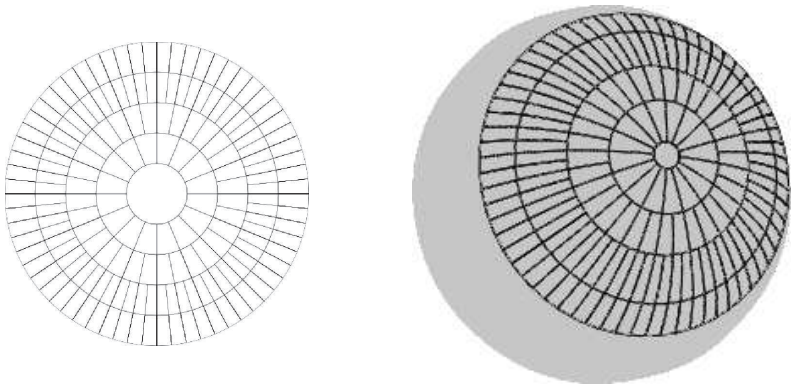


Figura 3.9: (a) Distribuição dada pelo Algoritmo de Subdivisões (b) Distribuição na calota esférica de altura l em regiões de áreas iguais.

Gostaríamos de encontrar uma projeção T que distribua os pixels de acordo com o Algoritmo de Subdivisões na calota (em cima da esfera) que é visualizada de tal forma que a área de todos os pixels sobre a esfera sejam iguais (Figura 3.9). Cada coroa será projetada em uma zona esférica. Da expressão para a área de uma superfície de revolução, podemos deduzir que a área de uma zona esférica (Figura 3.10) é $2\pi Rh$ onde R é o raio da esfera e h é a altura da zona esférica.

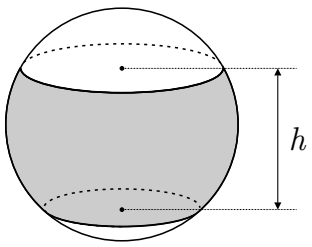


Figura 3.10: Zona esférica de altura h.

A quantidade de pixels em cada coroa é dada pelo vetor k_{aux}. Diremos que h_0 é a altura referente a zona esférica determinada por $T(Q_{0,0})$ na esfera, h_i, $i = 1, ..., c-1$ será a altura da zona esférica dada pela projeção da coroa i na esfera. Assim (Figura 3.11),

$$h_0 + h_1 + \cdots + h_{c-1} = l. \tag{3.3}$$

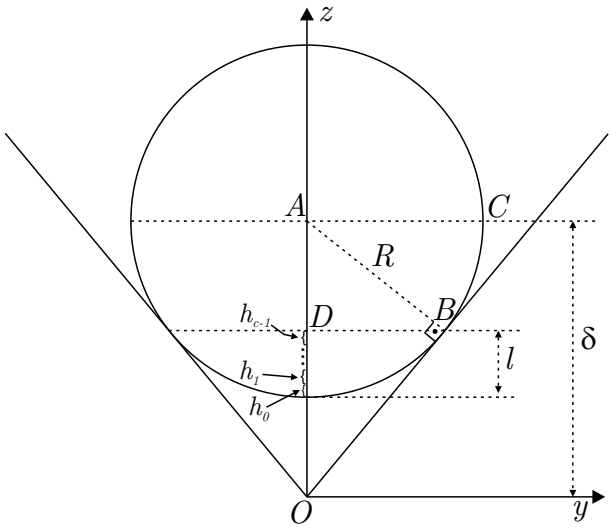


Figura 3.11: Alturas $h_i's$ distribuídas sobre o comprimento l.

Queremos que os pixels representem áreas iguais na esfera. A área A_0 de $T(Q_{0,0})$ é dada por $2\pi Rh_0$. A área da $i-ésima$ coroa é dada por $2\pi Rh_i$. Como a entrada $k_{aux}[i-1]$ (3.1) fornece a quantidade de pixels da $i-ésima$ coroa então $A_i = \dfrac{2\pi Rh_i}{k_{aux}[i-1]}$, onde A_i é a área de $T(Q_{i,j})$ de qualquer um dos pixels da coroa i. Como $A_i = A_0$ para todo i então $2\pi Rh_0 = \dfrac{2\pi Rh_i}{k_{aux}[i-1]}$ donde $h_i = h_0 k_{aux}[i-1]$. De (3.3) vem que $h_0 + h_0 k_{aux}[0] + h_0 k_{aux}[1] + ... + h_0 k_{aux}[c-2] = l$, donde, $h_0 = \dfrac{l}{\left(1 + \sum_{i=1}^{c-1} k_{aux}[i-1]\right)}$. E assim temos determinadas todas as alturas h_i, ou seja:

$$A_p = 2\pi Rh_0 = 2\pi R \frac{l}{\left(1 + \sum_{i=1}^{c-1} k_{aux}[i-1]\right)} = \frac{2\pi R^2}{\delta}\left(\frac{\delta - R}{1 + \sum_{i=1}^{c-1} k_{aux}[i-1]}\right).$$

Resumidamente, demonstramos o seguinte teorema (de contribuição própria):

Teorema:

(i) Consideremos uma esfera S de raio R, um ponto O que dista δ do centro da esfera e uma imagem $I \in \Omega_T$ de $P \times P$ pixels capturada por um observador em O retratando parte da esfera S. Então, a parte visível de S em I é a calota de altura $l = \dfrac{R}{\delta}(\delta - R)$.

(ii) Consideremos ainda a calota na esfera S de altura l dividida em uma calota de altura $h_0 = \dfrac{l}{\left(1 + \sum_{i=1}^{c-1} k_{aux}[i-1]\right)}$ e $c-1$ zonas esféricas de altura $h_i = h_0 k_{aux}[i-1]$ divididas em $k_{aux}[i-1]$ partes, onde $i = 1, ..., c$ e de tal forma que o vetor k_{aux} e o valor de c são dados pelo Algoritmo de Subdivisões. Então, a área A_p de cada uma dessas regiões da

calota de altura l é:

$$A_p = \frac{2\pi R^2}{\delta}\left(\frac{\delta - R}{1 + \sum_{i=1}^{c-1} k_{aux}[i-1]}\right).$$

Observação: No teorema acima o vetor k_{aux}, que fornece a quantidade de pixels por coroa, pode ser qualquer, não necessariamente o que adotamos por simplicidade em (3.1) do Algoritmo de Subdivisões.

Agora, supondo que $r_e(i)$ seja o raio das circunferências dadas pelo Algoritmo de Subdivisões, gostaríamos ainda que T fosse de tal forma que $r_e(c) - r_e(c-1) = 1$ de forma a satisfazer as hipóteses impostas acima. Assim, considerando os mesmos sistemas de coordenadas da Figura 3.8, gostaríamos de encontrar um ponto N, centro da projeção cônica T (Figura 3.12 (a)), de tal forma que a projeção satisfaça $r_e(c) - r_e(c-1) = 1$.

Faremos agora algumas observações referentes à Figura 3.12 (a):

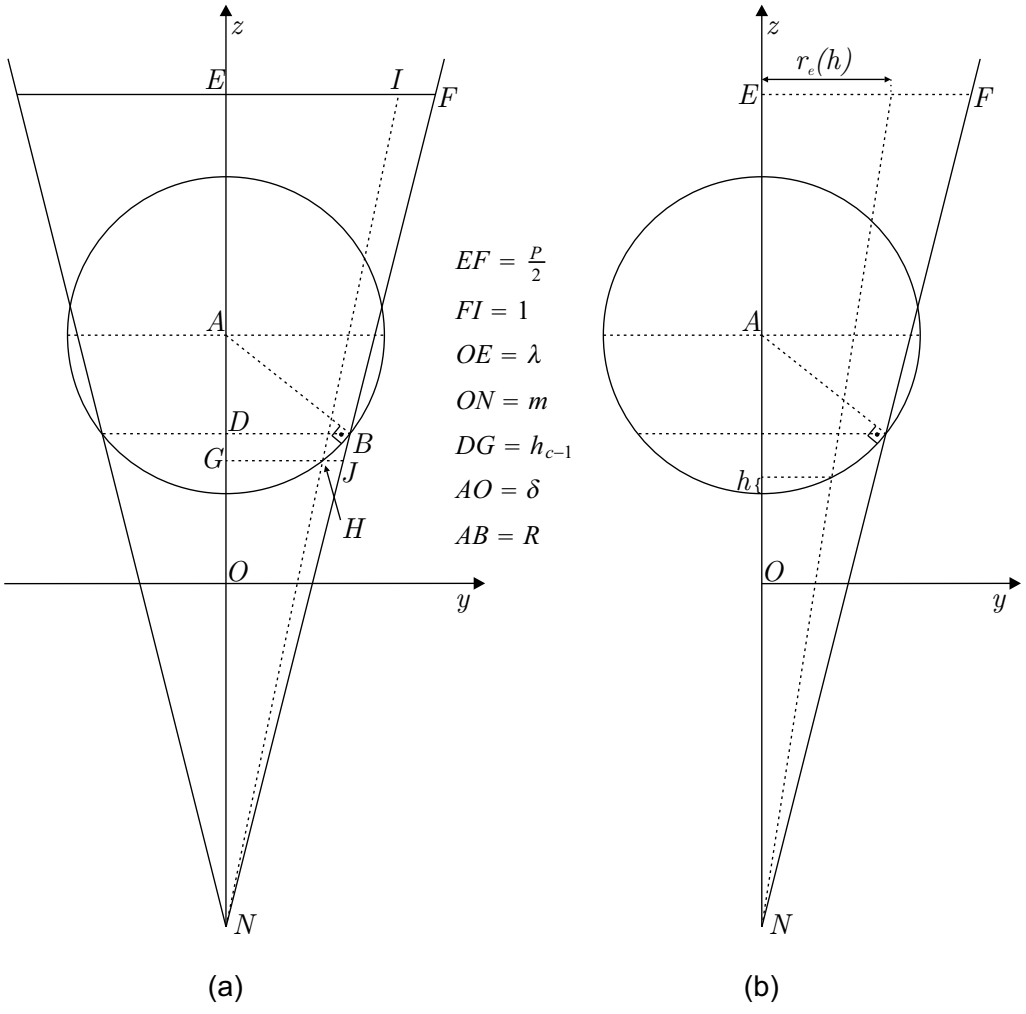


Figura 3.12: (a) Esquematização para o cálculo de N (b) Esquematização para o cálculo dos raios $r_e(h)$.

Os triângulos NJH e NFI são semelhantes donde $NF = \dfrac{NJ}{HJ}$. Os triângulos NGJ e NEF são semelhantes donde $NF = \dfrac{(NJ)(EF)}{GJ}$. Igualando temos $GJ = \dfrac{P}{2}HJ$, donde $GH + HJ = \dfrac{P}{2}HJ$. Daí, $HJ = \dfrac{2GH}{P-2}$. Agora, de acordo com o triângulo AGH temos que $GH = \sqrt{R^2 - (AD + h_{c-1})^2}$.

Denotemos $m = ON$ e $\lambda = OE$. Daí, como os triângulos NGJ e NDB são semelhantes temos que $\dfrac{NG}{GJ} = \dfrac{ND}{DB}$ donde,

$$\frac{m + \delta - AD - h_{c-1}}{\sqrt{R^2 - (AD + h_{c-1})^2\left(1 + \frac{2}{P-2}\right)}} = \frac{m + \delta - AD}{\sqrt{R^2 - AD}} \Longrightarrow$$

$$m = \frac{(P-2)(\delta - AD - h_{c-1})\sqrt{R^2 - AD^2} - P(\delta - AD)\sqrt{R^2 - (AD + h_{c-1})^2}}{P\sqrt{R^2 - (AD + h_{c-1})^2} - (P-2)\sqrt{R^2 - AD^2}}.$$

Além disso, como os triângulos NDB e NEF são semelhantes temos que $\dfrac{ND}{NE} = \dfrac{DB}{EF}$ donde $\dfrac{m + \delta - AD}{\lambda + m} = \dfrac{\sqrt{R^2 - AD^2}}{\frac{P}{2}}$. Daí, $\lambda = \dfrac{P}{2}\dfrac{m + \delta - AD}{\sqrt{R^2 - AD^2}} - m$.

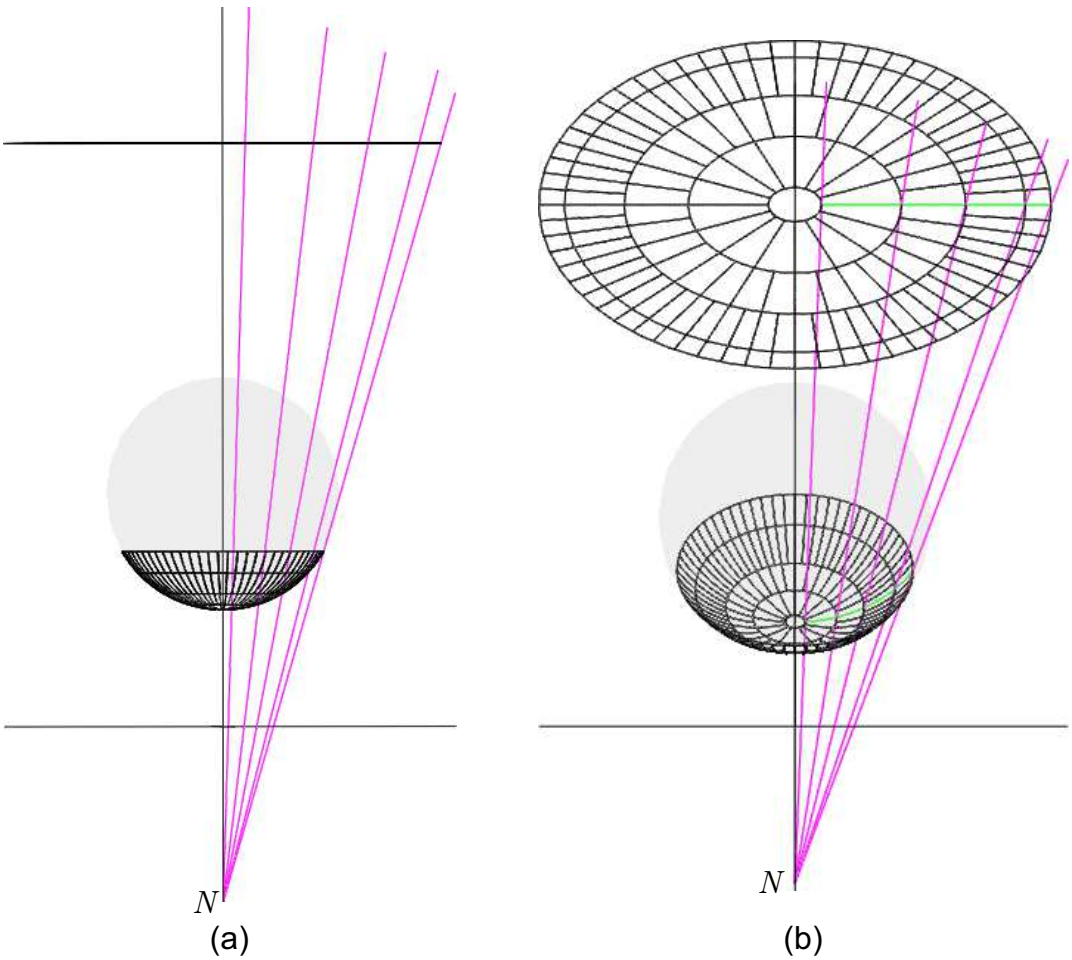


Figura 3.13: (a) Esquematização bidimensional da obtenção de $r_e(h)$ (b) Esquematização tridimensional da obtenção de $r_e(h)$.

Com m e λ determinados podemos encontrar uma expressão para T que leve uma altura h, conforme Figura 3.12 (b), em um raio $r_e(h)$. Temos então que $T(h) = r_e(h)$ onde

$$r_e(h) = \frac{(\lambda + m)\sqrt{2Rh - h^2}}{m + \delta - R + h}.$$

A Figura 3.13 ilustra, em um exemplo tridimensional, como os raios do espaço suporte elíptico são obtidos. Ressaltamos que os espaços Ω_H e Ω_E têm a mesma quantidade de *pixels polares*, dada pelo Algoritmo de Subdivisões. A diferença entre os espaços está na forma como são determinados os raios das circunferências. Na Figura 3.13 (b) podemos observar, na esfera embaixo, que as regiões delimitadas sobre a esfera têm a mesma área e é a partir desta distribuição que determinamos os raios das circunferências deste modelo.

Chamaremos de Espaço Suporte Elíptico, e denotaremos por Ω_E, o espaço que tem a distribuição do Algoritmo de Subdivisões com as medidas dos raios das circunferências dadas por $r_i = r_e(h_0 + h_1 + ... + h_i)$.

Suponhamos que a imagem seja exibida em um monitor em que cada pixel meça b cm de largura. Assim, para identificarmos um pixel do modelo Ω_E no monitor basta utilizarmos a fórmula $R_i = b\frac{P}{2}\frac{r_i(m + \delta - (R - l))}{(\lambda + m)\sqrt{R^2 - (R - l)^2}}$ para encontrarmos os raios equivalentes ao espaço Ω_E, já que o modelo é conforme.

Um caso particular interessante de cálculo de área de região S na esfera ocorre quando sua imagem no monitor está delimitada pelos raios s_1 e s_2 ($s_2 > s_1$) e pelos ângulos β_1 e β_2 ($\beta_2 > \beta_1$) (área em cinza na Figura 3.14). Nesta situação não precisamos fazer a tradicional contagem de pixels, mas sim utilizarmos a seguinte relação:

$$h(s) = \frac{\delta\lambda_1^2 + \lambda_1\sqrt{(R^2 - \delta^2)Y^2 + R^2\lambda_1^2}}{Y^2 + \lambda_1^2}$$

onde $\lambda_1 = \lambda + m$ e $Y = \frac{2s\lambda_1\sqrt{2Rl - l^2}}{bP(m + \delta - R + l)}$.

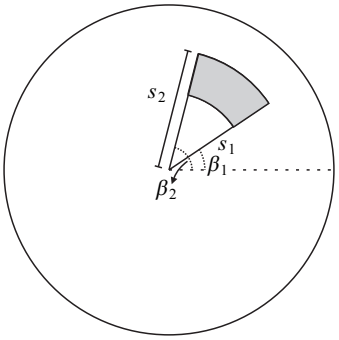


Figura 3.14: Área de uma região de uma imagem exibida em um monitor.

Logo, a área é dada por $2\pi R(h(s_2) - h(s_1))\frac{\beta_2 - \beta_1}{2\pi}$. Entretanto, ressaltamos que, no nosso modelo, só afirmamos que a área referente a quaisquer dois pixels na esfera

são iguais, sendo assim, se tomarmos outra região S' da esfera de tal modo que $s_2 - s_1$ e $\beta_2 - \beta_1$ sejam iguais às de S, podemos ter S e S' com áreas diferentes.

Novamente no caso de $P = 50$ e supondo que $\delta = 15$ e $R = 1$, temos que os raios $r_e(i)$ das 14 circunferências que compõe o espaço são:

i	1	2	3	4	5	6	7
$r_e(i)$	0.74	3.39	5.78	7.43	9.92	11.88	13.54

i	8	9	10	11	12	13	14
$r_e(i)$	14.99	17.46	19.52	21.27	22.76	24	25

onde nas primeiras linhas são dados os valores de i e nas segundas linhas os valores de $r_e(i)$.

Observamos novamente, que a largura da última coroa é 1 conforme as hipóteses que impomos.

O espaço suporte para este exemplo é dado pela Figura 3.15. É interessante observar que, enquanto a largura das coroas do modelo Ω_H sempre diminuem à medida que se aproximam do bordo (pois estão associadas às unidades da métrica hiperbólica), as coroas do modelo Ω_E não seguem tal padrão. Neste caso, o decrescimento das larguras das coroas muda de acordo com cada camada pois sua taxa de variação está relacionada com o vetor k_{aux}.

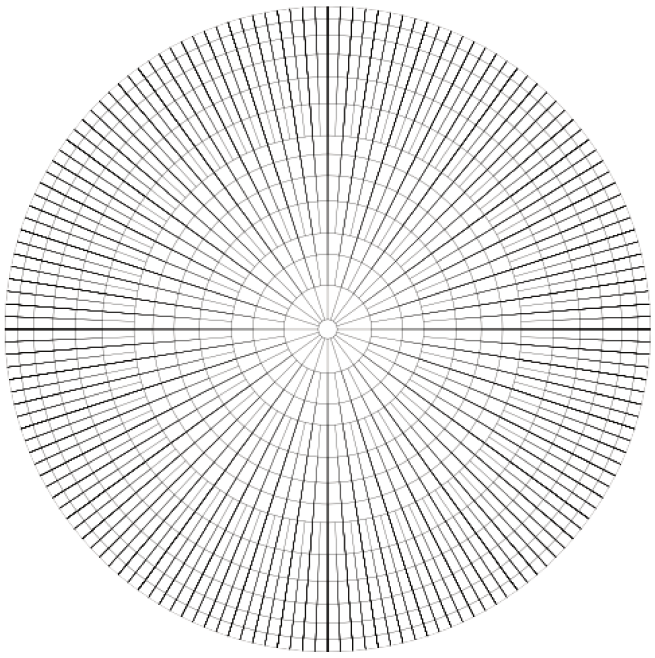


Figura 3.15: Espaço suporte Ω_E para $P = 50$.

3.1.3.1 Área em Ω_E

O espaço suporte elíptico possui a vantagem de, dada uma imagem em Ω_E, os pixels $Q_{i,j}$ possuem todos a mesma área sobre a esfera S_R que gerou a imagem. Dessa forma, se quiséssemos saber a área aproximada de determinada região na esfera bastaria contar quantos pixels pertencem à região em Ω_E e multiplicar pela área (dada pelo Teorema)

$$A_p = \frac{2\pi R^2}{\delta}\left(\frac{\delta - R}{1 + \sum_{i=1}^{c-1} k_{aux}[i-1]}\right)$$

de cada pixel na esfera e, então, teremos um valor aproximado para a área desejada.

Vejamos um exemplo prático desse algoritmo. Consideraremos duas imagens em Ω_T e, com o auxílio do software "*GIMP*[2]" tornamos a imagem binária de tal forma que, no círculo que representa a imagem, a região que desejamos calcular a área tenha valor 1. Em seguida, convertemos essa imagem binária para Ω_E, contamos quantos pixels têm valor 1 e multiplicamos pela área A_p.

(a) (b)

Figura 3.16: (a) Imagem de 864×864 pixels representando a esfera em estudo e (b) Imagem binária em Ω_E evidenciando região para o cálculo da área.

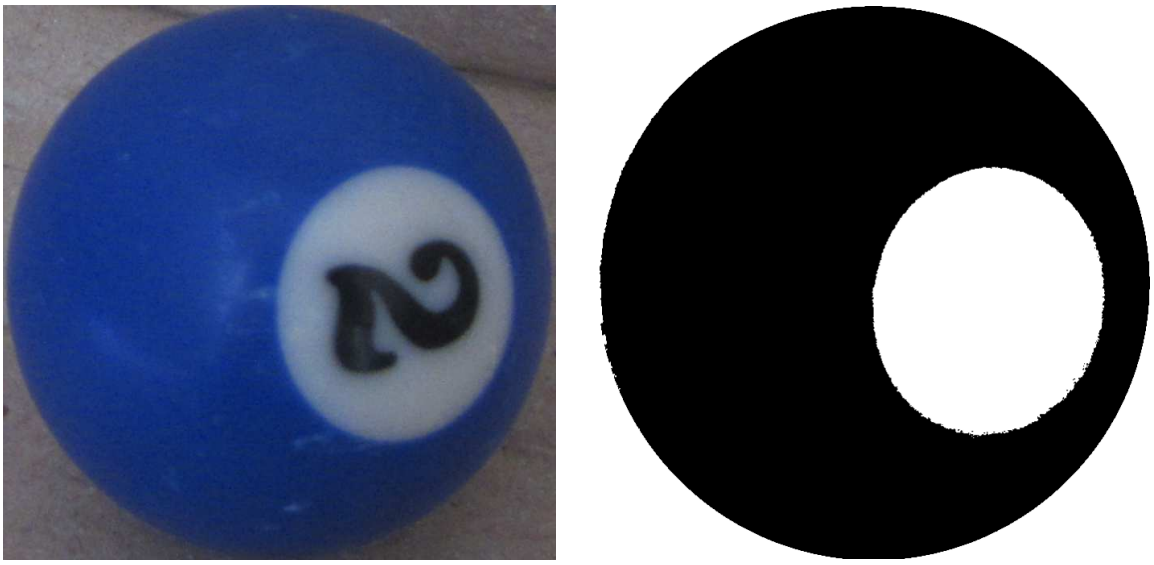
Consideremos a imagem dada pela Figura 3.16 (a) (de 864×864 pixels) em Ω_T e seu correspondente binário em Ω_E Figura 3.16 (b). Desejamos saber a área da região branca na Figura 3.16 (a). Para isso, precisamos saber o raio R da esfera e a distância δ entre o centro da esfera e o observador. O comprimento de um círculo máximo dessa

[2] https://www.gimp.org/

esfera é 8 cm, donde seu raio R é $\frac{8}{2\pi}$. A distância da superfície onde a esfera foi apoiada até a câmera fotográfica é de 17 cm, donde a $\delta = 17 - R$. A área de cada pixel na esfera é dada por $A_p = \frac{2\pi R^2}{\delta}\left(\frac{\delta - R}{473364}\right) = 1.9776036855420163 \times 10^{-5}$. A região desejada tem 87864 pixels em Ω_E de forma que a área é 1.737601 cm^2.

Para constatar que o valor obtido se aproxima do valor real da área desejada consideramos que a região se aproxima de uma calota da esfera e medimos a altura da calota. Assim, a medida física que obtivemos foi $2\pi R\,(2R - 2.33) = 1.731832$ cm^2.

De forma análoga consideremos a esfera dada pela Figura 3.17 (a) a seguir de 688 × 688 pixels. Gostaríamos de considerar o número 2 sobre a esfera como se estivesse em branco e calcular a área da parte branca. Novamente com o software "*GIMP*", retiramos o número 2 e tornamos a imagem binária em Ω_E (Figura 3.17 (b)). As medidas necessárias são: $R = \frac{10}{2\pi}$ e $\delta = 25 - R$. A área de cada pixel na esfera é dada por $A_p = 5.1572169971168264 \times 10^{-5}$. A região desejada tem 54939 pixels em Ω_E donde a área desejada é 2.833323 cm^2.



(a) (b)

Figura 3.17: (a) Imagem de 688 × 688 pixels representando a esfera em estudo e (b) Imagem binária em Ω_E evidenciando região para o cálculo da área.

A medida do valor físico da área que obtivemos nesse caso foi $2\pi R\,(2R - 2.9) = 2.830998$ cm^2.

Observamos que, devido a precariedade dos equipamentos de medição disponíveis tradicionalmente e a seleção da região de interesse na imagem ter sido feita manualmente, as medidas reais e as obtidas pelo algoritmo são iguais apenas até a segunda casa decimal.

O algoritmo que propomos calcula áreas aproximadas de quaisquer regiões sobre a esfera. Tomamos calotas nos nossos exemplos pela facilidade de comparação com o valor real. Em trabalhos futuros proporemos um algoritmo automático para o cálculo de áreas que torne a imagem binária demarcando a região de interesse e forneça a área desejada.

3.1.4 Influência dos parâmetros nos espaços suporte propostos

Conforme comentado anteriormente, alguns parâmetros do Algoritmo de Subdivisões podem ser alterados. Vejamos alguns exemplos de alterações e como tais alterações influenciam em um espaço suporte para o caso $P = 256$.

A entrada p_r do algoritmo impõe que haja, aproximadamente, $0.785 p_r P^2$ pixels nos modelos propostos. Entretanto, devido aos diversos arredondamentos impostos por tal algoritmo, essa quantidade pode ser bastante diferente da obtida.

Chamaremos de caso 1 as escolhas de $n_{min} = 10$ e $m = 1$. Caso 2, as escolhas de $n_{min} = 5$ e $m = 1$ e, de caso 3, $n_{min} = 15$ e $m = 2$. As tabelas a seguir relatam as diferenças de comportamento dos espaços suportes ao tomarmos, respectivamente, $p_r = 0.8$ e $p_r = 0.9$ em cada um dos três casos.

$p_r = 0.8$ (espera-se 62.8% dos pixels iniciais)

	c	n	c_1	k	% obtida
Caso 1	107	13	5	7	63.69
Caso 2	114	6	4	8	56.33
Caso 3	92	25	3	6	59.09

$p_r = 0.9$ (espera-se 70.65% dos pixels iniciais)

	c	n	c_1	k	% obtida
Caso 1	128	13	6	7	76.43
Caso 2	142	6	5	8	70.41
Caso 3	122	25	4	6	78.77

Observamos que a diferença entre a quantidade de pixels que se espera e a que se obtêm para $p_r = 0.8$ é menor no caso 1 e, para $p_r = 0.9$, no caso 2.

3.1.5 Vizinhança de um pixel

Já vimos que diversas aplicações necessitam da definição de vizinhança de pixel para um dado modelo. Vejamos como definiremos os vizinhos de um pixel nos nossos modelos. Recordemos que, como o que diferencia os dois modelos são as medidas dos raios,

a noção de vizinhança é a mesma pra os modelos Ω_H e Ω_E. Para facilitar a visualização utilizaremos ambos os espaços suporte e suas representações no plano $\rho\omega$.

Consideraremos, nos nossos modelos, dois tipos de vizinhanças. Chamaremos de V_1 a vizinhança análoga à vizinhança do tipo $4-$conexo, ou seja, os pixels vizinhos são definidos pelas arestas em comum. E, chamaremos de V_2 a vizinhança análoga à vizinhança do tipo $8-$conexo, onde as arestas e os vértices comuns definem os pixels vizinhos. Ao contrário do que ocorre no caso euclidiano, nos nossos modelos um pixel não tem uma quantidade fixa de vizinhos. A quantidade depende da posição do pixel na imagem. Além disso, devido à periodicidade da variável angular e a conveniência de haver uma "continuidade" na imagem, diremos que a aresta que define $\omega = 0$ é a mesma aresta que define $\omega = 2\pi$.

Na Figura 3.18 temos alguns exemplos. As vizinhanças dos pixels P_1, P_2, P_3 e P_4 são do tipo V_2 enquanto as vizinhanças dos pixels P_5, P_6 e P_7 são do tipo V_1. Observamos, na representação $\rho\omega$, que uma parte da vizinhança de P_3 está na parte de cima do gráfico. Além disso, observamos, na vizinhança do pixel P_4 como estamos tomando as vizinhanças dos pixels que estão na borda da imagem. Definimos ainda como vizinhos do pixel central (sem divisões) como todos os pixels da primeira coroa.

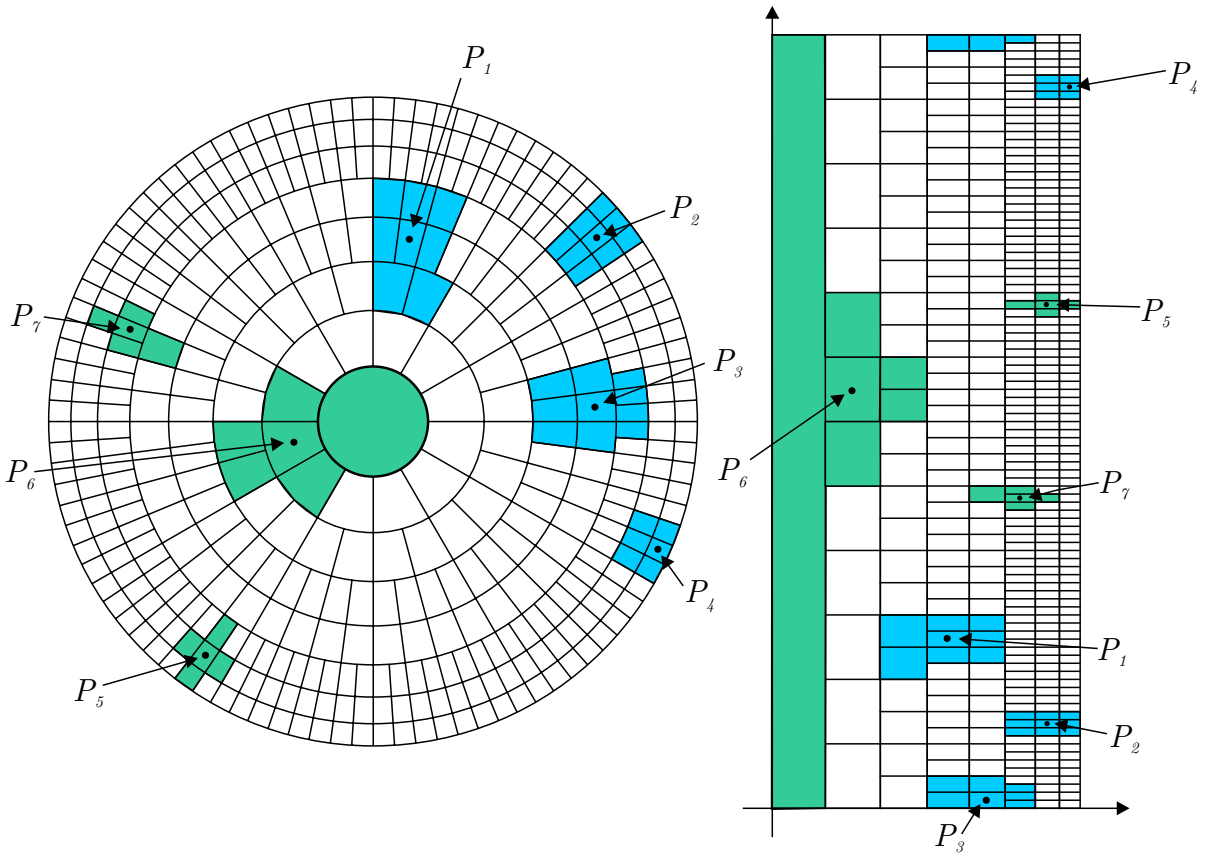


Figura 3.18: Vizinhanças do tipo V_1 em verde e do tipo V_2 em azul representadas no plano xy e no plano $\rho\omega$.

Neste trabalho utilizaremos apenas as vizinhanças do tipo V_2.

3.1.6 Quantização

Nesta subseção admitimos que o espaço suporte Ω pode ser Ω_H ou Ω_E. Chamaremos de $r_{eh}(i)$ a função que define os raios das circunferências que delimitam as coroas nos modelos. A quantização, conforme comentamos anteriormente, é feita sobrepondo os modelos Ω e Ω_T. Os pixels de Ω_T cujo ponto central (do pixel) pertençam à região delimitada por um pixel em Ω contribuirão para o tom de cinza deste pixel e o tom de cinza em Ω será tomado como a média aritmética dos tons de cinza dos pixels de Ω_T que pertencerem à região.

Para explicar, computacionalmente, como é feita a quantização utilizaremos algumas notações da linguagem *Python*.

- $M[i][j]$ é a entrada (i,j) da matriz M (a numeração começa no 0);

- $x//y$ é a divisão inteira de x por y;

- $M[i][j] += x$ significa que estamos adicionando x ao valor da entrada $M[i][j]$.

Seja M a matriz que é a representação digital da imagem $f \in \Omega_T$, ou seja, $M[i][j]$ é o tom de cinza do pixel $P_{i,j}$ de coordenadas (i,j). Chamaremos de Q o conjunto de sequências de tons de cinza $Q[i][j]$ dos pixels $Q_{i,j} \in \Omega$ dado por $Q = \left\{v_i = (Q[i][j])_j\right\}$ onde i e j variam de acordo com a posição do pixel Q_{ij} em Ω, ou seja, $Q = \{v_0 = Q[0][0]\} \cup \{v_i = (Q[i][j]) \mid i = 1,..,c \text{ e } j = 0,...,k_{aux}[i-1]\}$.

Utilizaremos uma matriz de pares ordenados de dimensão $P \times P$, que chamaremos de I, que relaciona os índices dos pixels $P_{i,j}$ de Ω_T com os pixels $Q_{i,j}$ de Ω de tal forma que se, por exemplo, $I[i][j] = (i_1, j_1)$ então $P_{i,j}$ fez parte da quantização de Q_{i_1,j_1}. O processo de determinação de i_1 e j_1 será descrito abaixo.

Para o cálculo computacional, primeiramente consideraremos todas as coordenadas cartesianas (i,j) em coordenadas polares $(r(i,j), \theta(i,j))$ tendo como centro $\mathcal{O} = \left(\frac{P-1}{2}, \frac{P-1}{2}\right)$ da imagem. Para o cálculo do raio definimos a função $r(i,j) = \|(i,j) - \mathcal{O}\|_e$, onde $\|.\|_e$ é a distância euclidiana. Para o cálculo do ângulo consideremos a função

$$\Phi(i,j) = \arctan\left(\frac{\frac{P-1}{2} - j}{i - \frac{P-1}{2}}\right).$$

$$\text{Daí, } \theta(i,j) = \begin{cases} \Phi(i,j) & \text{se } x > \frac{P-1}{2} \text{ e } y < \frac{P-1}{2} \\ \Phi(i,j) + 2\pi & \text{se } x > \frac{P-1}{2} \text{ e } y > \frac{P-1}{2} \\ \Phi(i,j) + \pi & \text{se } x < \frac{P-1}{2} \text{ e } y < \frac{P-1}{2} \\ \Phi(i,j) + \pi & \text{se } x < \frac{P-1}{2} \text{ e } y > \frac{P-1}{2} \\ \frac{\pi}{2} & \text{se } x = \frac{P-1}{2} \text{ e } y < \frac{P-1}{2} \\ \frac{3\pi}{2} & \text{se } x = \frac{P-1}{2} \text{ e } y > \frac{P-1}{2} \\ 0 & \text{se } x > \frac{P-1}{2} \text{ e } y = \frac{P-1}{2} \\ \pi & \text{se } x < \frac{P-1}{2} \text{ e } y = \frac{P-1}{2} \end{cases}, \ 0 \leqslant \theta < 2\pi.$$

Utilizaremos um vetor para armazenar os raios do modelo: $v = [r_{he}(1), ..., r_{he}(c)]$ e uma matriz M_{count} de dimensão $P \times P$ que será utilizada como contador.

Para $i = 0, ..., P-1$ e $j = 0, ..., P-1$ calculamos $r(i,j)$ e $\theta(i,j)$:

- Caso $r(i,j) > \dfrac{P}{2}$ então $I[i][j] = (0,1)$ para posteriormente descartar tais pixels.

- Caso $r(i,j) < r_{he}(1)$ então $I[i][j] = (0,0)$, ou seja, esse pixels está contribuindo para o tom de cinza de $Q_{0,0}$.

- Caso $r_{he}(1) \leqslant r(i,j) \leqslant \dfrac{P}{2}$:

- Verificamos em qual coroa do espaço está (i,j) (ou seja, verificamos para qual i_1 a desigualdade $r_{he}(i_1) \leqslant r(i,j) < r_{he}(i_1+1)$ é verdadeira), determinando assim, i_1.

- Com o auxílio de $\theta(i,j)$ calculamos $j_1 = \theta(i,j) // \dfrac{2\pi}{k_{aux}[i_1 - 1]}$.

Dessa forma, sabemos que $P_{i,j}$ contribuiu para o tom de cinza do pixel Q_{i_1,j_1}. Assim, fazemos $I[i][j] = (i_1, j_1)$ e $Q[i_1][j_1] += M[i][j]$.

Com o auxílio de uma matriz de contadores fazemos a média aritmética de todos os $M[i][j]$ que contribuíram para $Q[i_1][j_1]$.

Alguns pixels $Q_{i,j}$ (área delimitada pelo pixel) não contêm nenhuma coordenada (i,j) de $P_{i,j}$. Para estes pixels optamos por uma abordagem que toma como tom de cinza de $Q_{i,j}$ o tom de cinza do $P_{i,j}$ mais próximo de $Q_{i,j}$.

Determinamos, portanto, os tons de cinza de todos os pixels $Q_{i,j}$ do modelo. Além disso, determinamos que $Q[0][1]$, que não é um pixel do modelo, tenha tom de cinza como sendo o branco absoluto. Este artifício foi criado para identificar os pixels que serão irrelevantes para a visualização da imagem resultante desse processo em monitores retangulares tradicionais (são os pixels dos "quatro cantos da imagem quadrada exteriores ao disco").

Para visualizar a imagem F dada por Q transformamos os pixels em coordenadas cartesianas por não dispormos de um monitor que exiba a imagem nos modelos circulares que estamos propondo. A matriz I foi utilizada para nos auxiliar nesta tarefa. Para isso, basta fazermos $F[i][j] = Q[i_1][j_1]$ onde $I[i][j] = (i_1, j_1)$.

A implementação deste algoritmo em linguagem *Python* está disponível no Apêndice B.

4 Um Modelo Matemático de Imagem

Nos últimos anos, diversos modelos probabilísticos para imagens têm sido propostos por pesquisadores de processamento de imagens, computação visual, matemática aplicada e estatística [11]. Neste trabalho optamos por utilizar o modelo gaussiano. Apresentamos neste capítulo um interessante modelo matemático proposto em [7] que utiliza o espaço das distribuições gaussianas como espaço dos tons de cinza da imagem. Analisamos este modelo que pode ser utilizado juntamente com qualquer espaço suporte de imagem e desenvolvemos dois novos ordenamentos descritos nas subseções 4.1.3 e 4.1.7. As demais subseções introduzem os ordenamentos propostos em [7] que serão também utilizados na análise de imagens no Capítulo 5.

Recordemos que, ao se fazer a captura de uma imagem, vários frames são tomados. Normalmente, para gerar a imagem final, a média da luminosidade recebida por cada pixel é utilizada, entretanto, pode-se utilizar também o desvio padrão. Seguindo esta ideia, os autores propuseram um modelo onde considera-se uma vizinhança de cada pixel e toma-se a média e o desvio padrão dessa vizinhança, substituindo, assim, o tom de cinza $t \in \mathbb{R}$ inicial por uma distribuição gaussiana. Este modelo pode ser utilizado com qualquer espaço suporte.

Consideraremos, inicialmente, um espaço suporte Ω qualquer. Assim, o modelo transforma uma imagem $g : \Omega \longrightarrow \mathbb{R}$ em uma imagem $f : \Omega \longrightarrow \mathcal{N}$ com $f(p) = N(\mu, \sigma) \in \mathcal{N}$, onde $\mathcal{N}$ é o espaço das distribuições gaussianas.

As operações do modelo que transformam g em f são dadas por:

Item (1) a imagem é normalizada para ter média $\mu = 0$ e desvio padrão $\sigma = 1$:

$$g(p) \mapsto \hat{g}(p) = \frac{g(p) - \mu(g)}{\sigma(g)}$$

onde $\mu(g)$ é a média de todos os tons de cinza da imagem g e $\sigma(g)$ é o desvio padrão.

Item (2) as partes real e imaginária de $f(p)$ são obtidas calculando, respectivamente, a média e o desvio padrão de uma vizinhança de comprimento W de p:

$$\hat{g}(p) \mapsto f(p) = \mu_W(\hat{g})(p) + i\sigma_W(\hat{g})(p) = f_x(p) + if_y(p).$$

Em seguida, para utilizarmos a métrica de Fisher em $\mathcal{H}^2$, que mede a distância entre distribuições normais de probabilidade, fizemos a seguinte mudança de variáveis $(\mu, \sigma) = \left(\frac{\mu}{\sqrt{2}}, \sigma\right)$, conforme vimos na seção 1.3. Além disso, para a representação destas distribuições em $\mathcal{H}^2$ temos que garantir que $f_y(p) > 0$. Para isso, adicionamos $\epsilon > 0$ pequeno a $\sigma_W(\hat{g})(p)$.

Consideremos, neste momento, que $\Omega = \Omega_T$.

O conjunto de imagens $\mathcal{F}\left(\Omega, \mathcal{H}^2\right)$ será um reticulado completo quando $\left(\mathcal{H}^2, \leqslant\right)$ for um reticulado completo e, nesse caso, podemos considerar as operações de erosão $\varepsilon_B(f)$ e dilatação $\delta_B(f)$ definidas abaixo:

$$\varepsilon_B(f)(p) = \bigwedge_{q \in B(p)} f(p+q)$$
$$\delta_B(f)(p) = \bigvee_{q \in B(p)} f(q-p)$$

onde $B(p)$ é um elemento estrutural centrado em p. Nesse caso, essas operações satisfazem a lei da adjunção, ou seja,

$$\delta_B(f)(p) \leqslant g(p) \Longleftrightarrow f(p) \leqslant \varepsilon_B(g)(p); \quad \forall f, g \in \mathcal{F}\left(\Omega, \mathcal{H}^2\right).$$

As operações de abertura e fechamento são tais que

$$\gamma_B(f) = \delta_B(\varepsilon_B(f))$$
$$\varphi_B(f) = \varepsilon_B(\delta_B(f))$$

conforme vimos no Capítulo 2.

Ressaltamos que, no caso da captura de vários frames da mesma imagem, ao invés de tomarmos a média e o desvio padrão de uma vizinhança de p tomamos a média e o desvio padrão da mesma posição (i, j) nos diversos frames.

4.1 Ordenamentos

O cálculo da erosão e dilação necessita de definições de ordenamentos em $\mathcal{H}^2$. O artigo [7] apresenta seis ordenamentos que veremos a seguir. Além disso, propomos um ordenamento (o sétimo) que chamaremos de *ordenamento circular*. Utilizaremos conceitos apresentados no Capítulo 1.

Para ilustrar os ordenamentos consideremos um exemplo.

Exemplo: Consideremos o conjunto de pontos

$$\begin{aligned} Z = \{&(-3, 1.5), (-1.8, 5), (-1.3, 1), (0, 3.7), (1.6, 2.2), \\ &(2, 0.8), (2.8, 5.1), (3.5, 3.7), (4, 2.5), (7.2)\} \end{aligned} \tag{4.1}$$

contido em $\mathcal{H}^2$ exibido na Figura 4.1.

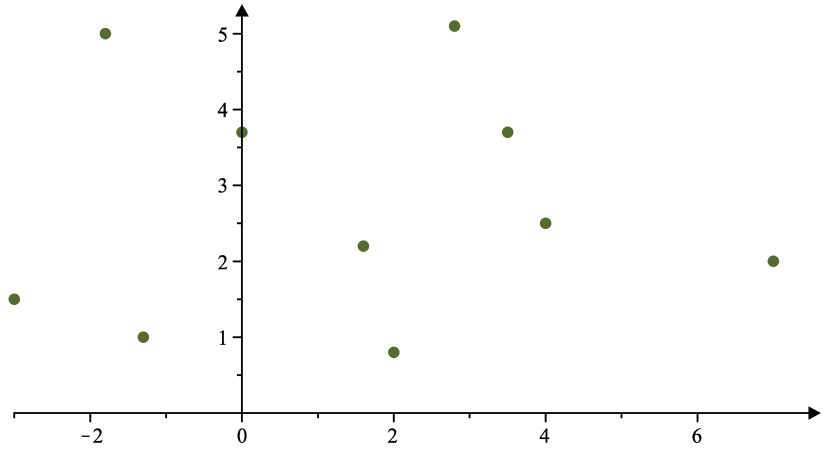


Figura 4.1: Pontos do conjunto Z representados em $\mathcal{H}^2$.

Veremos a seguir as definições dos ordenamentos e o supremo e o ínfimo de um conjunto genérico finito $Z \in \mathcal{H}^2$ para cada ordenamento.

4.1.1 Ordenamento 1: Ordenamento produto

Definamos $\mathcal{H}^2_+ = \left\{z \in \mathcal{H}^2 \mid x \geqslant 0 \text{ e } y \geqslant 1\right\}$. Assim,

$$z_1 \leqslant_{\mathcal{H}^2} z_2 \Longleftrightarrow z_2 \boxminus z_1 \in \mathcal{H}^2_+$$

onde $z_2 \boxminus z_1 = (x_2 - x_1) + i\left(y_2 y_1^{-1}\right)$, sendo z_1 e z_2 pontos de $\mathcal{H}^2$. De acordo com esse ordenamento parcial podemos definir, respectivamente, o ínfimo e o supremo entre z_1 e z_2:

$$z_1 \wedge_{\mathcal{H}^2} z_2 = \min\{x_1, x_2\} + i \min\{y_1, y_2\}$$
$$z_1 \vee_{\mathcal{H}^2} z_2 = \max\{x_1, x_2\} + i \max\{y_1, y_2\}.$$

Generalizando: Seja $Z \subset \mathcal{H}^2$ um conjunto finito de pontos. Sejam

$$\overline{z_1} = \min\{\mathrm{Re}(z) \mid z \in Z\} + i \min\{\mathrm{Im}(z) \mid z \in Z\}$$
$$\overline{z_2} = \max\{\mathrm{Re}(z) \mid z \in Z\} + i \max\{\mathrm{Im}(z) \mid z \in Z\}.$$

Logo, $\overline{z_1} \leqslant_{\mathcal{H}^2} z$ para todo $z \in Z$ e $z \leqslant_{\mathcal{H}^2} \overline{z_2}$ para todo $z \in Z$. Dizemos que $\overline{z_1}$ é o ínfimo de Z e $\overline{z_2}$ é o supremo de Z.

A Figura 4.2 ilustra o ordenamento produto.

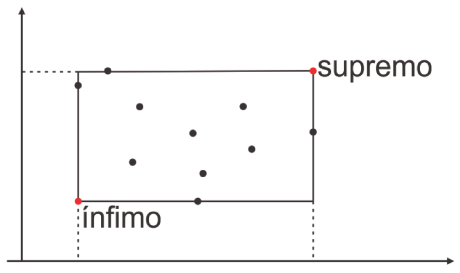


Figura 4.2: Ordenamento produto.

De forma análoga ao que acontece na morfologia tradicional (com tons de cinza em $\mathbb{R}$) o ordenamento produto definido em $\overline{\mathcal{H}^2}$ é um reticulado completo. Assim, $\wedge_{\mathcal{H}^2}$ e $\vee_{\mathcal{H}^2}$ podem ser utilizados para construir as operações de erosão e dilatação definidas no Capítulo 2 de forma que a lei da adjunção seja válida. Nesse caso, a abertura e o fechamento também estão bem definidos.

Consideremos a seguinte involução

$$\complement z = -x + iy^{-1}.$$

Com respeito a essa involução é fácil ver que

$$z_1 \vee_{\mathcal{H}^2} z_2 = \complement\left(\complement z_1 \wedge_{\mathcal{H}^2} \complement z_2\right) \text{ e } z_1 \wedge_{\mathcal{H}^2} z_2 = \complement\left(\complement z_1 \vee_{\mathcal{H}^2} \complement z_2\right),$$

ou seja, temos uma dualidade entre o ínfimo e o supremo.

Exemplo: O ínfimo do conjunto Z dado em (4.1) é o ponto $(-3, 0.8)$ (em vermelho na Figura 4.3) e o supremo é o ponto $(7, 5.1)$ (em azul).

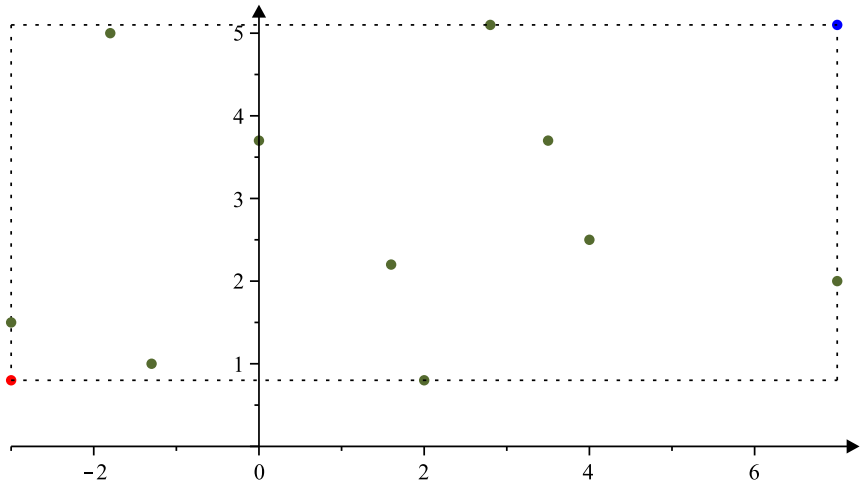


Figura 4.3: Ordenamento produto: ínfimo em vermelho e supremo em azul.

4.1.2 Ordenamento 2: Ordenamento simétrico

Consideremos uma divisão em quadrantes $\left\{\mathcal{H}^2_{-+}, \mathcal{H}^2_{--}, \mathcal{H}^2_{+-}, \mathcal{H}^2_{++}\right\}$ do espaço $\mathcal{H}^2$ a partir do eixo y e da reta de equação cartesiana $y = 1$ conforme Figura 4.4.

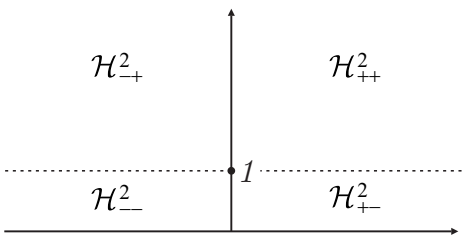


Figura 4.4: Divisão de $\mathcal{H}^2$ em quadrantes.

O ordenamento simétrico é um ordenamento dado por:

$$z_1 \preceq_{\mathcal{H}^2} z_2 \iff \begin{cases} 0 \leqslant x_1 \leqslant x_2 & \text{e} \quad 0 \leqslant \ln(y_1) \leqslant \ln(y_2) & \text{ou} \\ x_2 \leqslant x_1 \leqslant 0 & \text{e} \quad 0 \leqslant \ln(y_1) \leqslant \ln(y_2) & \text{ou} \\ x_2 \leqslant x_1 \leqslant 0 & \text{e} \quad \ln(y_2) \leqslant \ln(y_1) \leqslant 0 & \text{ou} \\ 0 \leqslant x_1 \leqslant x_2 & \text{e} \quad \ln(y_2) \leqslant \ln(y_1) \leqslant 0 & \end{cases}.$$

Observamos que ordenamos apenas pontos que estão no mesmo quadrante de forma que o ponto que estiver mais próximo (na métrica euclidana) do ponto $(0,1)$ é o menor.

O semiplano $\mathcal{H}^2$ com este ordenamento é um poset com menor elemento dado por $(0,1)$, entretanto, não existe um maior elemento.

Definimos o ínfimo $\curlywedge_{\mathcal{H}^2}$ e o supremo $\curlyvee_{\mathcal{H}^2}$ por:

$$z_1 \curlywedge_{\mathcal{H}^2} z_2 \iff \begin{cases} \min\{x_1,x_2\} + i\min\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{++} \\ \max\{x_1,x_2\} + i\min\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{-+} \\ \max\{x_1,x_2\} + i\max\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{--} \\ \min\{x_1,x_2\} + i\max\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{+-} \\ (0,1) & \text{caso contrário} \end{cases}$$

e

$$z_1 \curlyvee_{\mathcal{H}^2} z_2 \iff \begin{cases} \max\{x_1,x_2\} + i\max\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{++} \\ \min\{x_1,x_2\} + i\max\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{-+} \\ \min\{x_1,x_2\} + i\min\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{--} \\ \max\{x_1,x_2\} + i\min\{y_1,y_2\} & \text{se } z_1, z_2 \in \mathcal{H}^2_{+-} \\ \text{não existe} & \text{caso contrário} \end{cases}$$

De modo análogo ao ordenamento anterior, podemos definir o ínfimo e o supremo para um conjunto finito $Z \in \mathcal{H}^2$.

Temos que $\mathcal{H}^2$ com as definições $\curlywedge_{\mathcal{H}^2}$ e $\curlyvee_{\mathcal{H}^2}$ é um inf-semireticulado completo, pois o supremo não está definido para pontos em diferentes quadrantes de $\mathcal{H}^2$. Podemos definir as operações de erosão e dilatação utilizando $\curlywedge_{\mathcal{H}^2}$ e $\curlyvee_{\mathcal{H}^2}$. Esses operadores não irão satisfazer a lei da adjunção, pois o supremo entre dois pontos nem sempre está definido. Entretanto, conforme [7], mesmo com a dilatação não estando bem definida para todos os conjuntos de pontos de $\mathcal{H}^2$, ela estará bem definida no conjunto de pontos resultante da erosão desses pontos. Dessa forma, ainda poderemos calcular a abertura utilizando $\curlywedge_{\mathcal{H}^2}$ e $\curlyvee_{\mathcal{H}^2}$. É claro que, nesse caso, o fechamento está apenas parcialmente definido.

Exemplo: Neste caso o ínfimo do conjunto Z dado em (4.1) é o ponto $(0,1)$ (em vermelho na Figura 4.5), enquanto o supremo não está definido.

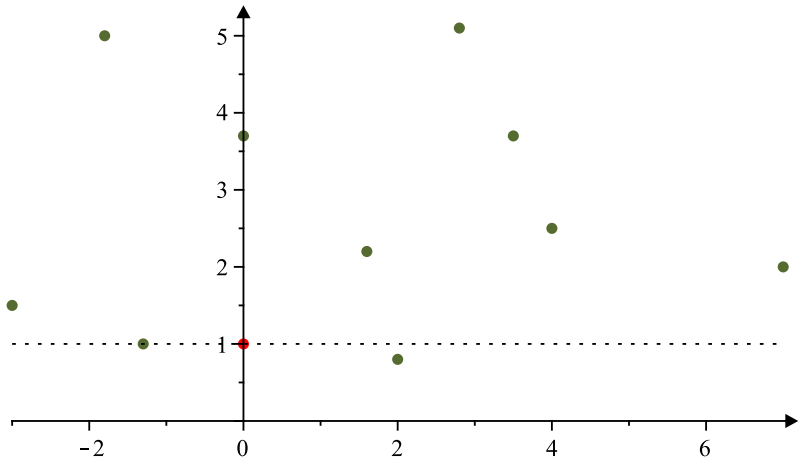


Figura 4.5: Ordenamento simétrico: ínfimo em vermelho.

4.1.3 Ordenamento 3: Ordenamento polar

Um ordenamento parcial, proposto em [7], utilizando a métrica hiperbólica de $\mathcal{H}^2$ é dado por:

$$z_1 \leqslant^1_{\mathcal{H}^2} z_2 \iff \begin{cases} \eta_1 < \eta_2 & \text{ou} \\ \eta_1 = \eta_2 & \text{e} \quad \tan\phi_1 \leqslant \tan\phi_2 \end{cases},$$

onde $\eta = dist_{\mathcal{H}^2}\left((x,y),(0,1)\right)$ e $\phi = \arctan\dfrac{x^2+y^2-1}{2x}$ (Figura 4.6) com $z = (x,y)$.

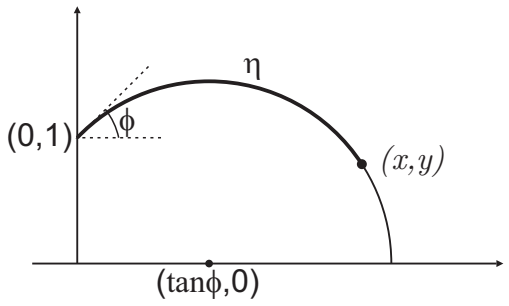


Figura 4.6: Ordenamento polar.

Observamos que $(\tan\phi, 0)$ é o centro da circunferência que determina a geodésica que passa por $(0,1)$ e (x,y). Dessa forma, em caso de $\eta_1 = \eta_2$, calculamos $\tan\phi_1$ e $\tan\phi_2$ para decidir o ordenamento entre os pontos. Entretanto, como $\tan(\phi) = \tan(\phi+\pi)$ temos infinitos pares de pontos tais que $\eta_1 = \eta_2$ e $\tan\phi_1 = \tan\phi_2$. Além disso, a tangente não está definida para os valores de $\phi = \dfrac{\pi}{2}$ ou $\phi = \dfrac{3\pi}{2}$. Assim , por exemplo, não conseguimos ordenar dois pontos no eixo y que estiverem a mesma distância hiperbólica do ponto $(0,1)$. Logo, o ordenamento é parcial.

Inspirados neste ordenamento, desenvolvido em [7], propomos neste trabalho o que chamaremos de ordenamento polar. Análogo ao que foi feito acima, caso $\eta_1 < \eta_2$ então $z_1 \leqslant^{pol}_{\mathcal{H}^2} z_2$. Porém, trataremos o caso em que $\eta_1 = \eta_2$ de maneira diferente. Nesse

caso, calculamos os ângulos ϕ_1 e ϕ_2 considerando que $\phi_1, \phi_2 \in [0, 2\pi[$. Em seguida, como $\mathcal{H}^2$ é conforme, definiremos $b(\phi) = (\cos(\phi), \text{sen}(\phi))$. Dessa forma, $b(\phi)$ é um ponto sobre uma circunferência euclidiana de centro no ponto $(0,1)$ e raio 1 de tal forma que ϕ é o ângulo que a semirreta de origem $(0,1)$ passando por $b(\phi)$ faz com a reta $y = 1$ (Figura 4.7). Utilizaremos então a projeção estereográfica τ a partir do ponto $(0,2)$ que levará os pontos da circunferência até o eixo x segundo a seguinte expressão: $\tau\left(b(\phi)\right) = \begin{cases} \dfrac{2\cos\phi}{1-\text{sen}\,\phi}, & \text{se } \phi \neq \dfrac{3\pi}{2} \\ 0 & \text{se } \phi = \dfrac{3\pi}{2} \end{cases}$.

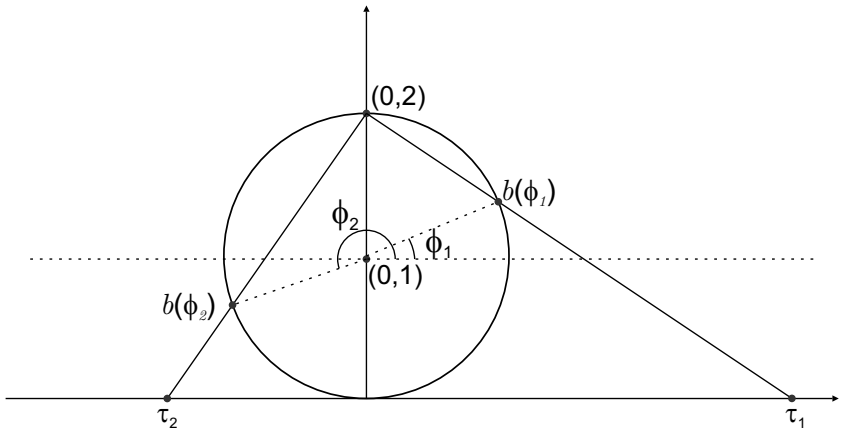


Figura 4.7: Circunferência auxiliar para o cálculo de $\tau(b)$.

Assim, se $z_1 = (x_1, y_1)$ e $z_2 = (x_2, y_2)$ o ordenamento polar será dado por

$$z_1 \leqslant^{pol}_{\mathcal{H}^2} z_2 \iff \begin{cases} \eta_1 < \eta_2 & \text{ou} \\ \eta_1 = \eta_2 & \text{e} \quad \begin{cases} x_2 = 0 \text{ e } y_2 > 1 & \text{ou} \\ \tau(b(\phi_1)) < \tau(b(\phi_2)) \end{cases} \end{cases}.$$

Como este ordenamento é total, podemos definir o ínfimo $\wedge^{pol}_{\mathcal{H}^2}$ e o supremo $\vee^{pol}_{\mathcal{H}^2}$ entre dois pontos de acordo com esse ordenamento, assim como o ínfimo e o supremo para um conjunto finito $Z \in \mathcal{H}^2$. Temos que $\overline{\mathcal{H}^2}$ com ínfimo $\wedge^{pol}_{\mathcal{H}^2}$ e supremo $\vee^{pol}_{\mathcal{H}^2}$ é um reticulado completo de forma que podemos definir as operações de erosão e dilatação utilizando $\wedge^{pol}_{\mathcal{H}^2}$ e $\vee^{pol}_{\mathcal{H}^2}$. O ordenamento no qual nos inspiramos satisfaz a lei da adjunção segundo [7]. Conjecturamos que este ordenamento também satisfará. Estão bem definidas, portanto, as operações de abertura e fechamento.

Exemplo: Neste caso, em nosso exemplo dado em (4.1), o ínfimo é dado pelo ponto $(-1.3, 1)$ (em vermelho na Figura 4.8), enquanto o supremo é dado pelo ponto $(2.8, 5.1)$ (em azul). Observamos que, como o ordenamento é total, tanto o ínfimo quanto o supremo são pontos do conjunto Z.

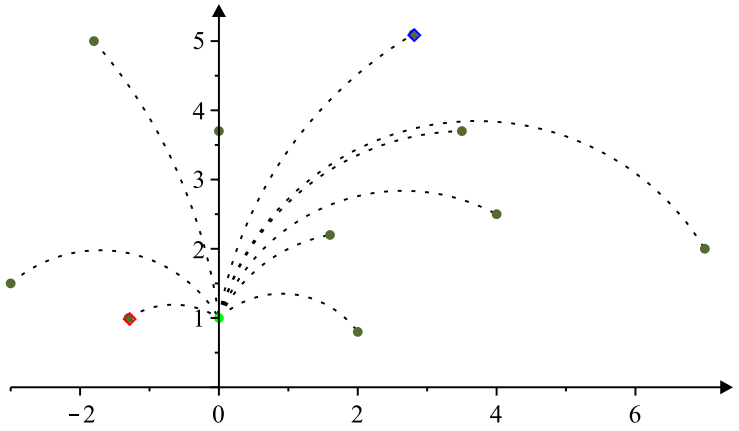


Figura 4.8: Ordenamento Polar: ínfimo em vermelho e supremo em azul.

4.1.4 Ordenamento 4: Ordenamento utilizando a distância de Hellinger

Este ordenamento é análogo ao ordenamento polar, diferenciando apenas a maneira de calcular a distância do ponto $z = (x, y)$ ao ponto $(1, 0)$. O cálculo dessa distância será feita utilizando a distância de Hellinger apresentada no Capítulo 1. Assim, temos:

$$z_1 \leqslant^{H}_{\mathcal{H}^2} z_2 \iff \begin{cases} \eta_1 < \eta_2 & \text{ou} \\ \eta_1 = \eta_2 & \text{e} \quad \begin{cases} x_2 = 0 \text{ e } y_2 > 1 & \text{ou} \\ \tau(b(\phi_1)) < \tau(b(\phi_2)) \end{cases} \end{cases}$$

onde $\eta = dist_{He}\left((x, y), (0, 1)\right)$ e τ é dado conforme ordenamento polar.

Este ordenamento também é total, de forma que $\overline{\mathcal{H}^2}$ com ínfimo $\wedge^{H}_{\mathcal{H}^2}$ e supremo $\vee^{H}_{\mathcal{H}^2}$ é um reticulado completo. Aqui também conjecturamos que seja válida a lei da adjunção e, também, que as operações de abertura e fechamento estão bem definidas.

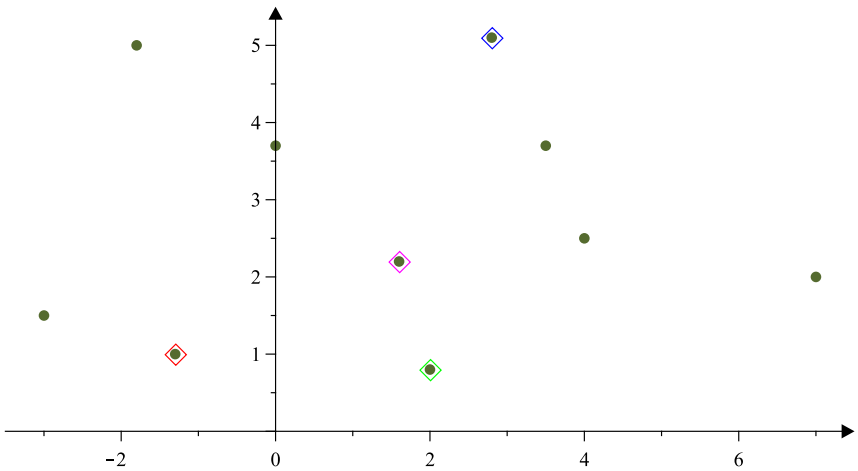


Figura 4.9: Ordenamento de Hellinger: para $\alpha = 0.01$ ínfimo em vermelho e supremo em azul. Para $\alpha = 20$ ínfimo em violeta e supremo em verde.

Exemplo: Utilizando a distância de Hellinger no conjunto Z dado em (4.1), temos que para $\alpha = 0.01$ o ínfimo é dado pelo ponto $(-1.3, 1)$ (em vermelho na Figura

4.9,) enquanto o supremo é dado pelo ponto $(2.8, 5.1)$ (em azul). Observamos que esses são os mesmos ínfimos e supremo dado pelo ordenamento polar. Para $\alpha = 20$ o ínfimo é dado pelo ponto $(1.6, 2.2)$ (em violeta), enquanto o supremo é dado por $(2, 0.8)$ (em verde). Observamos que a localização desses pontos destoa muito do caso em que $\alpha = 0.01$.

4.1.5 Ordenamento 5: Ordenamento geodésico

Para definirmos o ordenamento geodésico entre dois pontos z_1 e z_2 consideremos a geodésica que passa pelos dois pontos. Digamos que tal geodésica é a circunferência de centro $(a_{1\frown 2}, 0)$ e raio euclidiano $r_{1\frown 2}$. Assim, o ordenamento é dado por:

$$z_1 \preceq^{geo}_{\mathcal{H}^2} z_2 \Leftrightarrow \begin{cases} a_{1\frown 2} \leqslant x_1 < x_2 & \text{ou} \\ x_2 < x_1 \leqslant a_{1\frown 2} \end{cases}$$

caso $x_1 \neq x_2$ e

$$z_1 \preceq^{geo}_{\mathcal{H}^2} z_2 \Leftrightarrow y_2 \leqslant y_1$$

caso $x_1 = x_2$.

Neste ordernamento temos que a transitividade é válida apenas para pontos pertencentes a mesma geodésica.

Baseados nesse ordenamento podemos definir o ínfimo:

$$\text{se } x_1 \neq x_2 \quad z_1 \curlywedge^{geo}_{H^2} z_2 \Longleftrightarrow \begin{cases} \max\{x_1, x_2\} + i \max\{y_1, y_2\} & \text{se } x_1, x_2 \leqslant a_{1\frown 2} \\ \min\{x_1, x_2\} + i \min\{y_1, y_2\} & \text{se } x_1, x_2 \geqslant a_{1\frown 2} \\ z_{1\frown 2} & \text{caso contrário} \end{cases}$$
$$\text{se } x_1 = x_2 \quad z_1 \curlywedge^{geo}_{H^2} z_2 = x_1 + i \max\{y_1, y_2\}$$

Observamos, conforme Figura 4.10, que o ínfimo é dado pelo ponto da geodésica com parte imaginária maximal.

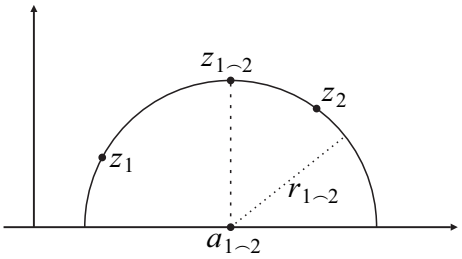


Figura 4.10: Ordenamento geodésico entre dois pontos.

Por outro lado, o supremo será definido por

$$\text{se } x_1 \neq x_2 \quad z_1 \curlyvee^{geo}_{H^2} z_2 \Longleftrightarrow \begin{cases} \min\{x_1, x_2\} + i \min\{y_1, y_2\} & \text{se } x_1, x_2 \leqslant a_{1\frown 2} \\ \max\{x_1, x_2\} + i \max\{y_1, y_2\} & \text{se } x_1, x_2 \geqslant a_{1\frown 2} \\ \complement z_{\complement 1\frown \complement 2} & \text{caso contrário} \end{cases},$$
$$\text{se } x_1 = x_2 \quad z_1 \curlyvee^{geo}_{H^2} z_2 = x_1 + i \min\{y_1, y_2\}$$

onde $\complement z_{\complement 1 \frown \complement 2}$ é o ponto dual associado à geodésica que conecta os pontos duais $\complement z_1$ e $\complement z_2$ ($\complement$ é a involução definida no ordenamento 1).

Entretanto, para os propósitos deste trabalho o ínfimo deve ser definido para qualquer quantidade finita de pontos. O cálculo de várias geodésicas é inviável computacionalmente. Para contornar esse problema, os autores sugeriram (em [7]) outra abordagem. Dado o conjunto de pontos $Z = \{(x_k, y_k)\}$, tomamos $Z^* = \{(x_k, -y_k)\}$ e calculamos o menor círculo que englobe todos os pontos Z e Z^* (com o algoritmo Menor Círculo Envolvente - MEC que propomos no Capítulo 1). Chamemos o centro desse círculo de $(a_{\text{inf}}, 0)$ e o raio de r_{inf} (Figura 4.11).

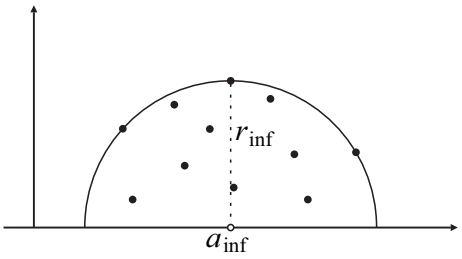


Figura 4.11: Ordenamento geodésico utilizando MEC.

Pela forma como foram escolhidos os pontos sabemos que a_{inf} está no eixo x, ou seja, essa semicirfunferência que engloba os pontos é uma geodésica. Definiremos o ínfimo como

$$\wedge_{\mathcal{H}^2}^{geo} Z = z_{\text{inf}} = a_{\text{inf}} + i r_{\text{inf}}.$$

Observemos que o z_{inf} divide o semicírculo geodésico determinado pelo MEC em dois arcos euclidianos congruentes, ou seja, podemos enxergar o z_{inf} como um ponto de simetria do semicírculo geodésico.

Novamente, nesse caso, o supremo não está definido para qualquer par de pontos. Para resolver isso consideraremos a involução $\complement$. Assim,

$$\vee_{\mathcal{H}^2}^{geo} Z = \complement\left(\wedge_{\mathcal{H}^2}^{geo} \complement Z\right) = z_{\text{sup}} = -x_c^{dual} + i\frac{1}{r^{dual}},$$

ou seja, aplicamos a involução em todos os pontos z_i, calculamos o ínfimo conforme explicado anteriormente, e calculamos a involução no ínfimo encontrado.

Definiremos operações de erosão e dilatação utilizando $\wedge_{\mathcal{H}^2}^{geo}$ e $\vee_{\mathcal{H}^2}^{geo}$, entretanto, essas operações não satisfazem a lei da adjunção [7]. Por isso, essas definições de supremo e ínfimo não geram erosão e dilatação conforme a definição formal utilizada na morfologia matemática. Porém, tais definições podem ser utilizadas para filtrar imagens.

Exemplo: Neste caso, o ínfimo do conjunto Z, dado em (4.1), é atingido no ponto $(1.406818, 5.940006)$ (em vermelho na Figura 4.12).

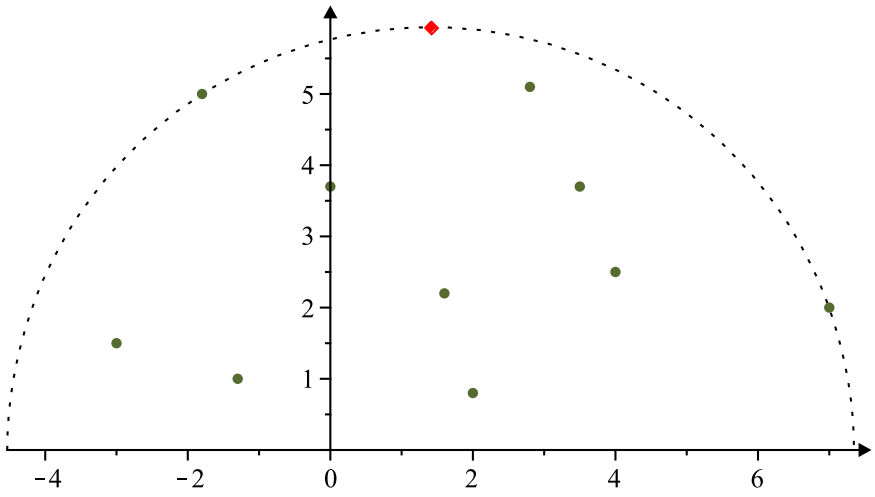


Figura 4.12: Ordenamento geodésico: ínfimo em vermelho.

Para o cálculo do supremo precisamos tomar a involução dos pontos de Z. Na Figura 4.13, temos rotulados de A a J os 10 pontos do conjunto Z. Temos que $\complement A = A', \complement B = B'$ e assim por diante. Temos, em roxo, o ponto $\wedge_{\mathcal{H}^2}^{geo}\complement Z$ e em azul o supremo $(1.990278, 0.198625)$.

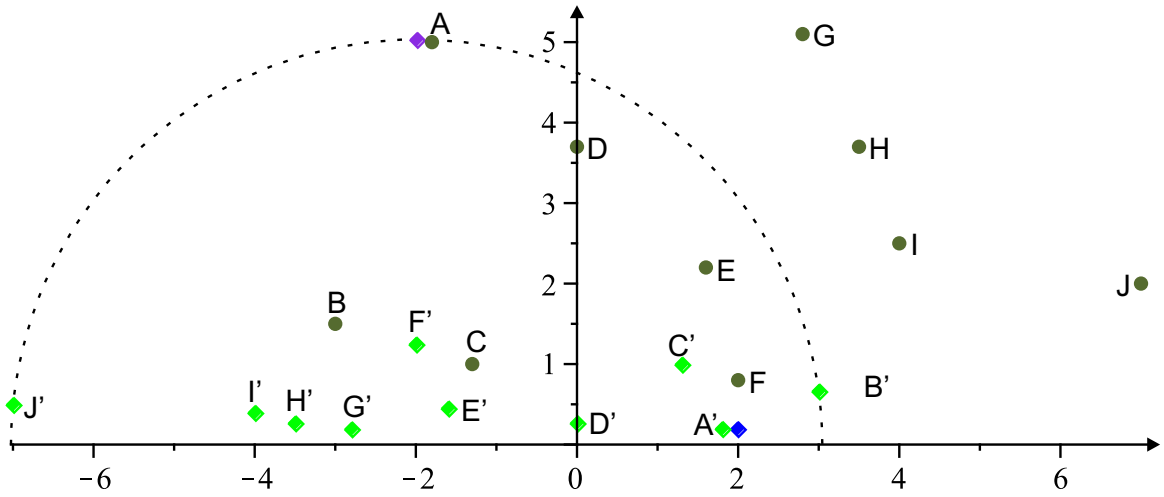


Figura 4.13: Pontos verde claro são imagens das involuções dos pontos verde escuros. Supremo em azul.

4.1.6 Ordenamento 6: Ordenamento geodésico assimétrico

As partes reais do z_{inf} e do z_{sup} do Ordenamento 5 podem não satisfazer a ordenação natural dos números reais. Entretanto, do ponto de vista da morfologia, pode ser potencialmente interessante impor um ordenamento entre as partes reais do ínfimo, do supremo e de todos os pontos de um conjunto finito $Z \in \mathcal{H}^2$. Para isso, vamos considerar o retângulo limitando o MEC (dado pelo ordenamento anterior) de dimensões $2r_{\text{inf}} \times r_{\text{inf}}$

(Figura 4.14). E dessa forma o ínfimo e o supremo são dados por

$$z_{\text{inf}}^{\overrightarrow{-}+} = \wedge_{\mathcal{H}^2}^{\overrightarrow{-}+} Z = (a_{\text{inf}} - r_{\text{inf}}) + i r_{\text{inf}}$$
$$z_{\text{sup}}^{\overrightarrow{-}+} = \vee_{\mathcal{H}^2}^{\overrightarrow{-}+} Z = -\left(x_c^{dual} - r^{dual}\right) + i\frac{1}{r^{dual}}$$

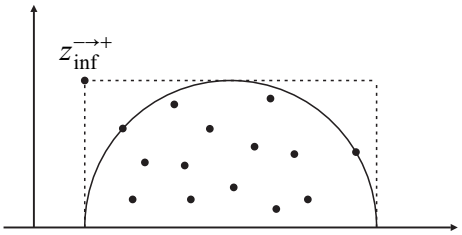


Figura 4.14: Ordenamento geodésico assimétrico.

Observemos que o ínfimo de $Z \in \mathcal{H}^2$ neste ordenamento deixou de ser um ponto que divide simetricamente o semicírculo determinado pelo MEC.

Definiremos operações de erosão e dilatação utilizando $\wedge_{\mathcal{H}^2}^{\overrightarrow{-}+}$ e $\vee_{\mathcal{H}^2}^{\overrightarrow{-}+}$ entretanto, essas operações não satisfazem a lei da adjunção [7]. Porém, análogo ao ordenamento 5, a erosão e a dilatação definidas com $\wedge_{\mathcal{H}^2}^{\overrightarrow{-}+}$ e $\vee_{\mathcal{H}^2}^{\overrightarrow{-}+}$ podem ser utilizadas para filtrar imagens.

Exemplo: Em Z, dado em (4.1), o ínfimo (em vermelho na Figura 4.15) é $(-4.533188, 5.940007)$ e o supremo (em azul) é $(7.024889, 0.198625)$.

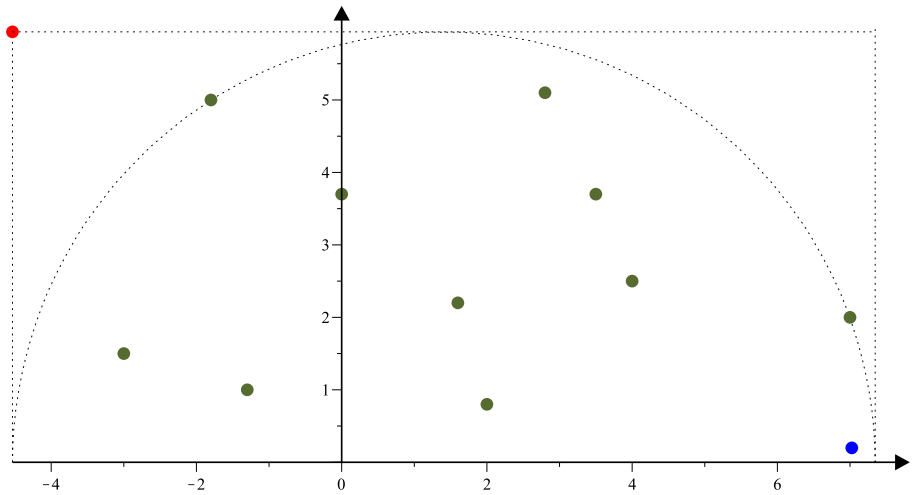


Figura 4.15: Ordenamento geodésico assimétrico: ínfimo representado em vermelho e supremo em azul.

4.1.7 Ordenamento 7: Ordenamento circular

Propomos esse ordenamento, de contribuição própria, inspirados no ordenamento produto, porém, utilizando a métrica hiperbólica que, como já vimos, é mais natural no espaço das distribuições gaussianas.

Dados dois pontos $z_1 = (x_1, y_1), z_2 = (x_2, y_2) \in \mathcal{H}^2$, consideraremos a geodésica que passa pelos dois pontos e diremos que $z_1 \leqslant^c_{\mathcal{H}^2} z_2$ quando z_1 estiver à esquerda de z_2 na geodésica, ou seja, quando $x_1 \leqslant x_2$.

Observemos que, assim como no ordenamento geodésico, esse ordenamento só satisfará a transitividade se os três pontos estiverem sobre a mesma geodésica.

Como já vimos devemos propor maneiras de ordenar, pelo menos, uma quantidade finita de pontos. A ideia que sugerimos é limitar os pontos por duas geodésicas de mesmo centro e por duas semirretas euclidianas passando pelo centro das geodésicas, conforme Figura 4.17.

Assim, seja $(m, 0)$ o centro e r o raio dado pelo MEC dos pontos dados e seja $d = \{d_1, ..., d_u\}$ onde d_i é a distância euclidiana entre $(m, 0)$ e z_i e seja $\theta = \{\theta_1, ..., \theta_u\}$ onde θ_i é de tal forma que

$$1^o \text{ caso (Figura 4.16 (a))}: \cos\theta_i = \frac{x_i - m}{d_i} \Longrightarrow \theta_i = \arccos\left(\frac{x_i - m}{d_i}\right)$$

$$2^o \text{ caso (Figura 4.16 (b))}: \cos\alpha = \frac{m - x_i}{d_i} \Longrightarrow \cos\theta_i = -\cos\alpha = \frac{x_i - m}{d_i},$$

ou seja, $\theta_i = \arccos\left(\frac{x_i - m}{d_i}\right) \; \forall\, (x_i, y_i) \in \mathcal{H}^2$.

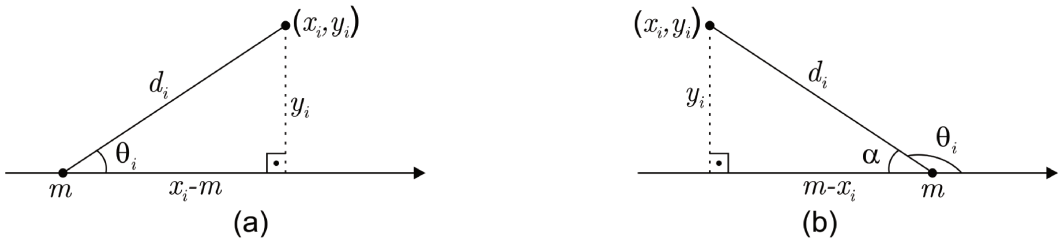


Figura 4.16: Cálculo de θ_i.

Denotemos $d_{\min} = \min_i \{d_i\}$, $d_{\max} = \max_i \{d_i\}$, $\theta_{\min} = \min_i \{\theta_i\}$ e $\theta_{\max} = \max_i \{\theta_i\}$.

Dessa forma, o ínfimo será tomado pela interseção da geodésica de centro $(m, 0)$ e raio $d_{\min}$ com a semirreta euclidiana passando por $(m, 0)$ e cujo ângulo com o eixo x é de $\theta_{\max}$. Analogamente, o supremo será dado pela interseção da geodésica de centro $(m, 0)$ e raio $d_{\max}$ com a semirreta euclidiana passando por $(m, 0)$ e cujo ângulo com o eixo x seja de $\theta_{\min}$. A Figura 4.17 ilustra como são tomados o ínfimo e o supremo neste ordenamento.

Resumidamente,

$$\wedge^c_{\mathcal{H}^2} z_i = (m + d_{\min}\cos(\theta_{\max}), d_{\min}\,\mathrm{sen}\,(\theta_{\max}))$$

e

$$\vee^c_{\mathcal{H}^2} z_i = (m + d_{\max}\cos(\theta_{\min}), d_{\max}\,\mathrm{sen}\,(\theta_{\min})).$$

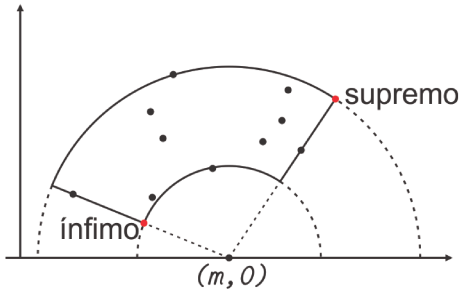


Figura 4.17: Ordenamento hiperbólico.

As definições de $\wedge^c_{\mathcal{H}^2}$ e $\vee^c_{\mathcal{H}^2}$ serão utilizadas para definir operações de erosão e dilatação.

Devido ao fato de o reticulado $(\mathcal{H}^2, \leqslant^c_{\mathcal{H}^2})$ não ser completo, as operações definidas utilizando $\wedge^c_{\mathcal{H}^2}$ e $\vee^c_{\mathcal{H}^2}$ não satisfazem a lei da adjunção, conforme podemos facilmente constatar pelo seguinte exemplo (Figura 4.18): suponhamos que a vizinhança de $q = (x, y)$ é dada por $\{q, q_1\}$ onde $q_1 = (x - \varepsilon, y)$ com $\varepsilon = 2$ (por exemplo). Analogamente tomemos o ponto $p = (x, y + 1)$ cuja vizinhança é dada por $\{p, p_1\}$ onde $p_1 = (x - \varepsilon, y + 1)$. Logo, $q = \vee^c_{\mathcal{H}^2} \{q, q_1\} \leqslant p$ e $p_1 = \wedge^c_{\mathcal{H}^2} \{p, p_1\} \leqslant q$.

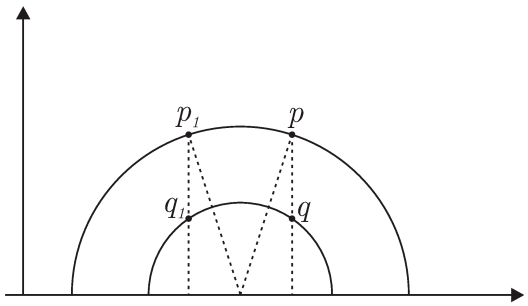


Figura 4.18: Lei da adjunção não é válida para $(\mathcal{H}^2, \leqslant^c_{\mathcal{H}^2})$.

Exemplo: Para o conjunto Z dado em (4.1), o ínfimo (em vermelho na Figura 4.19) é $(0.469044, 0.335327)$ e o supremo (em azul) é $(7.030000, 1.914028)$

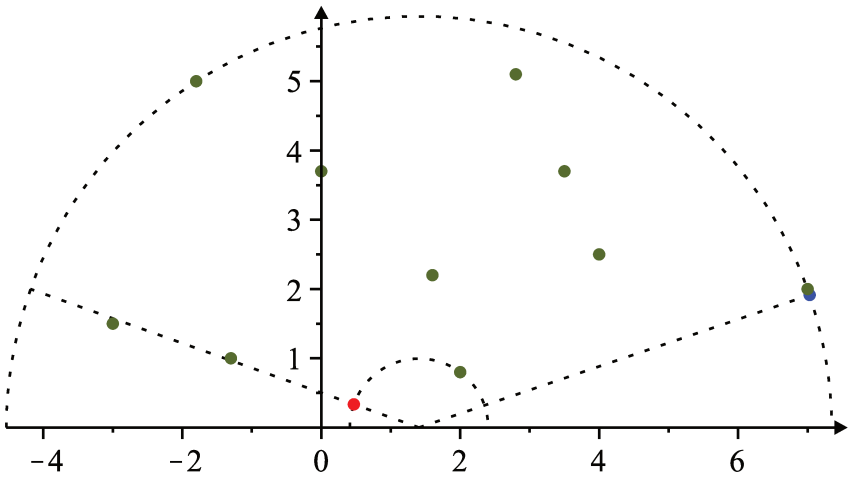


Figura 4.19: Ordenamento circular: ínfimo em vermelho e supremo em azul.

5 Aplicações dos Modelos Propostos

Vimos, no Capítulo 3, duas propostas de espaços suporte e, no Capítulo 4, um modelo matemático para tons de cinza em imagens em Ω_T (espaço suporte cartesiano tradicional). Neste capítulo iremos aplicar esse modelo matemático nos espaços suportes Ω_H e Ω_E, propostos nas subseções 3.1.2 e 3.1.3. Além disso, iremos apresentar algumas imagens geradas utilizando tais ferramentas. Observamos que, para a utilização dos nossos algoritmos, apesar das imagens serem coloridas, elas são primeiramente convertidas para tons de cinza (em escala $\mathbb{R}$) para que, em seguida sejam aplicados os processos propostos.

5.1 Conversão de imagens de Ω_T para os espaços Ω_H e Ω_E

Vejamos o resultado da conversão de uma imagem do espaço suporte Ω_T para os espaços suporte Ω_H e Ω_E. Consideremos a Figura 5.1 representada no espaço suporte Ω_T de 800×800 pixels.

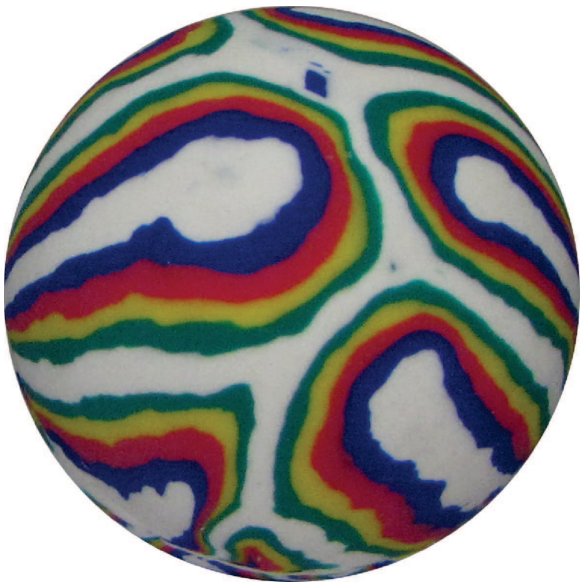


Figura 5.1: Imagem de 800×800 pixels em Ω_T.

Na Figura 5.2 (a) temos a Figura 5.1 representada no espaço suporte Ω_H e, na Figura 5.2 (b) a representação da mesma imagem no espaço suporte Ω_E.

Percebemos que, visualmente, as imagens são muito semelhantes à imagem original (considerando todas as imagens na escala de cinza de 0 a 255). Utilizaremos as medidas de PSNR (Peak signal-to-noise ratio - relação sinal-ruído de pico, medido em decibéis - dB) e MSE (Mean Squared Error - erro médio quadrático) [11] para comparar as imagens à imagem original dada pela Figura 5.1. Nessa comparação impomos que os

pixels que não pertencem ao disco dado pelo objeto de interesse tenham tons de cinza 255 (branco). Dessa forma, esses pixels não influenciarão no cálculo do PSNR. As equações que fornecem o PSNR e MSE entre duas imagens I_1 e I_2 ambas de $P \times P$ pixels em Ω_T são dadas por:

$$MSE = \frac{1}{P^2} \sum_{i=0}^{P-1} \sum_{j=0}^{P-1} \left[I_1(i,j) - I_2(i,j) \right]^2$$

$$PSNR = 10 \log_{10} \left(\frac{255^2}{MSE} \right) dB$$

O MSE varia de 0 a 255^2 sendo que quanto mais próximo de 0 menos erros de tons de cinza estamos cometendo na conversão. O $PSNR$ varia em $\mathbb{R}_+$ sendo que quanto maior menos erros.

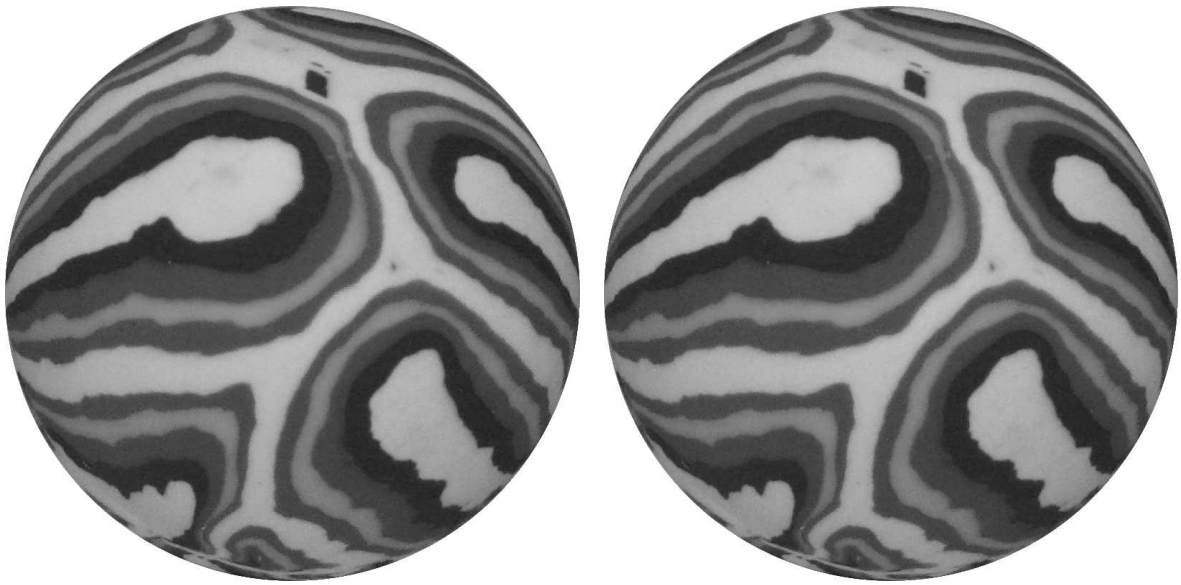


Figura 5.2: Conversão da Figura 5.1 para os espaço (a) Ω_H e (b) Ω_E.

Temos então os seguintes dados de comparação:

	Ω_H	Ω_E
MSE	85.5883546875	85.55414375
PSNR	28.806656830950715	28.80839311757859

Em termos de PSNR as duas imagens tem a mesma qualidade em relação à imagem original quando considerada em escala de cinza. Em trabalhos futuros abordaremos medidas alternativas de comparação das imagens que sejam mais adaptadas aos modelos que estamos trabalhando.

A Figura 5.2 nos leva a acreditar que não há muitas diferenças entre os espaços Ω_H e Ω_E quando há uma quantidade maior de pixels envolvida. Adaptamos os conceitos de MSE e PSNR para os nossos modelos para verificarmos tal conjectura, o que foi possível já que os dois modelos apresentam a mesma quantidade de pixels apesar de pixels correspondentes poderem estar em posições ligeiramente diferentes, já que os raios das circunferências dos modelos são diferentes. Assim,

$$MSE_\Omega = \frac{1}{1+\sum_{i=0}^{c-1} k_{aux}[i]} \left[\left[Q^H_{0,0} - Q^E_{0,0}\right]^2 + \sum_{i=1}^{c} \sum_{j=0}^{k_{aux}[i-1]} \left[Q^H_{i,j} - Q^E_{i,j}\right]^2 \right]$$

$$PSNR_\Omega = 10 \log_{10} \left(\frac{255^2}{MSE_\Omega} \right)$$

onde $Q^H_{i,j}$ e $Q^E_{i,j}$ são, respectivamente, os tons de cinza dos pixels da imagem em Ω_H e Ω_E.

Para as Figuras 5.2 (a) e 5.2 (b) temos $MSE_\Omega = 71.4$ e $PSNR_\Omega = 29.59$. Assim, as imagens, apesar de visualmente idênticas, apresentam diferenças ($MSE_\Omega \neq 0$).

Destacamos também que, apesar da quantidade de *pixels polares* dos dois modelos propostos ser exatamente a mesma, o espaço Ω_H concentra os pixels na borda da imagem, enquanto o espaço Ω_E distribui os pixels de maneira mais uniforme no modelo. O espaço Ω_E apresenta um custo de implementação maior devido à necessidade das informações preliminares sobre o objeto e a distância entre o objeto e o observador, embora retrate com mais fidelidade os elementos da esfera em termos de área. Por outro lado, o espaço Ω_H pode ser utilizado com qualquer objeto esférico, sem necessidade informações adicionais, porém não apresenta a relação entre as áreas.

5.2 Modelo matemático de imagem em Ω_H e Ω_E

O modelo matemático de imagens em Ω_H que estamos propondo é análogo ao apresentado no Capítulo 4 para imagens em Ω_T (ou seja, a conversão de $g : \Omega_H \to \mathbb{R}$ para $f = (f_x, f_y) : \Omega_H \to \mathcal{H}^2$), exceto por duas diferenças:

(i) as vizinhanças do ponto p necessárias no Item (2) do Capítulo 4 e nas definições de erosão e dilatação devem ser do tipo V_1 ou V_2 definidas no Capítulo 2;

(ii) como a rede de pontos de Ω_H não é uniformemente distribuída, devemos considerar as seguintes definições de erosão e dilatação:

$$\left[\varepsilon_\mathcal{G}(f)\right](v) = \min\left\{ f(v') \mid v' \in N^r_\mathcal{G}(v) \cup \{v\} \right\};$$
$$\left[\delta_\mathcal{G}(f)\right](v) = \max\left\{ f(v') \mid v' \in N^r_\mathcal{G}(v) \cup \{v\} \right\}.$$

Observamos que, como os conjuntos tomados aqui são vizinhanças (conjuntos finitos) podemos simplificar a notação e utilizar a nomenclatura mínimo e máximo ao invés de ínfimo e supremo.

Assim, para cada ponto $Q_{ij} \in \Omega_H$ consideraremos o conjunto

$$Z = \left\{ f\left(v'\right) \in \mathcal{H}^2 \mid v' \in N_{\mathcal{G}}^r\left(Q_{ij}\right) \cup \left\{Q_{ij}\right\} \right\}$$

e tomaremos o mínimo de Z segundo uma das definições de mínimo dadas no Capítulo 4 para o cálculo da erosão no ponto Q_{ij}. Para o cálculo da erosão da imagem o processo é repetido para todos os pixels $Q_{ij} \in \Omega_H$. O cálculo da dilatação é feito de modo análogo trocando-se o mínimo por máximo.

As definições de abertura e fechamento se mantêm as mesmas.

O mesmo raciocínio é válido para o espaço Ω_E.

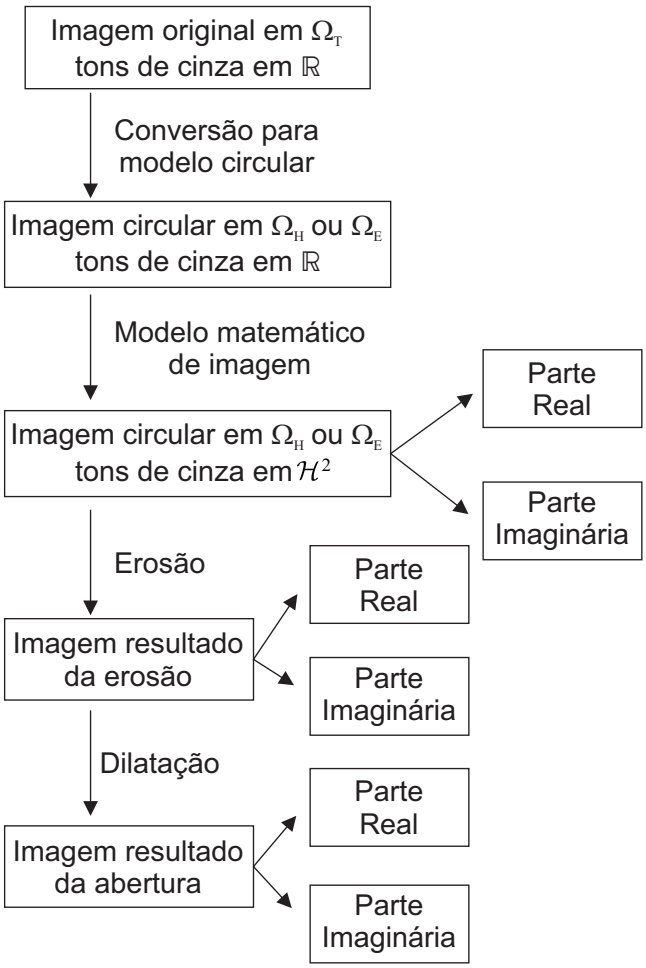


Figura 5.3: Esquema de processamento.

Para os exemplos a seguir o processamento será feito de acordo com o esquema dado pela Figura 5.3. Consideraremos uma imagem em Ω_T (com tons de cinza em $\mathbb{R}$), a converteremos para Ω_H ou Ω_E (com tons de cinza em $\mathbb{R}$) e, em seguida, aplicaremos o

modelo acima com as vizinhanças do tipo V_2 para obtermos uma imagem circular (Ω_H ou Ω_E) com tons de cinza em $\mathcal{H}^2$. No Item (2) do Capítulo 4 consideraremos vizinhanças de raio 2 e no cálculo de erosões ε e dilatações δ serão consideradas vizinhanças de raio 1. Os algoritmos que geraram tais imagens estão disponíveis no Apêndice B.

Conforme [7], é usual, em reticulados completos, trabalhar com as componentes real (eixo horizontal) e imaginária (eixo vertical) separadamente. Dessa forma, todas as imagens com tons de cinza em $\mathcal{H}^2$ apresentadas a seguir estão divididas em parte real (à esquerda) e parte imaginária (à direita).

Como vimos as imagens trabalhadas precisam ter escala de tons de cinza em $\mathbb{R}$ porém, não encontramos, na linguagem *Python*, nenhuma ferramenta que considere imagens em tal escala e, por isso, inserimos esse processo como parte do algoritmo através das funções abaixo, que convertem os tons de cinza do subintervalo $[-2, 2] \in \mathbb{R}$ (onde ocorre a maioria dos casos) de forma "aproximadamente linear" no intervalo $(0, 1)$.

$$X: \begin{array}{ccc} (0,1) & \longrightarrow & \mathbb{R} \\ t & \longmapsto & \dfrac{t-0.5}{t-t^2} \end{array} \qquad \text{e} \qquad X^{-1}: \begin{array}{ccc} \mathbb{R} & \longrightarrow & (0,1) \\ t & \longmapsto & \begin{cases} \dfrac{t-1+\sqrt{t^2+1}}{2t} & \text{se } t \neq 0 \\ 0.5 & \text{se } t = 0 \end{cases} \end{array}$$

Dessa forma, o *Python* reconhece as imagens com escala de cinza no intervalo $(0, 1)$, em seguida, o tom de cinza de cada pixel da imagem é convertido para escala $\mathbb{R}$ através da função X. A imagem é processada e a escala da imagem resultante é convertida para o intervalo $(0, 1)$ através da função X^{-1} para que o *Python* possa exibir a imagem no monitor.

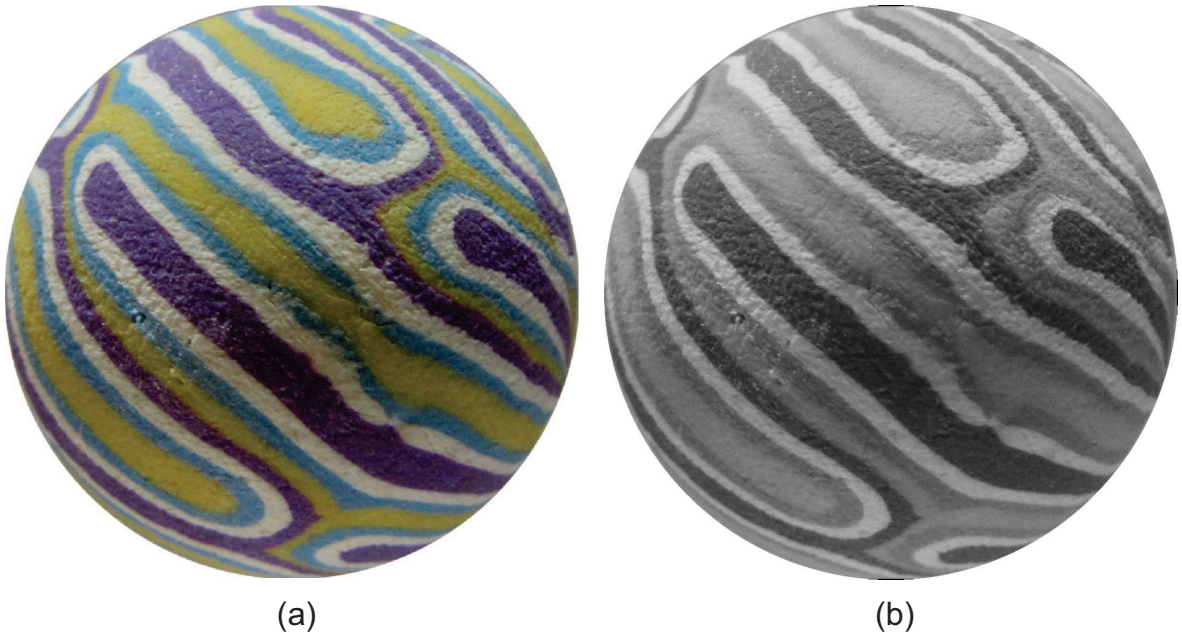


Figura 5.4: (a) Imagem em Ω_T e (b) em Ω_H.

Consideremos a Figura 5.4 (a) em Ω_T de 960×960 pixels e a Figura 5.4 (b) no espaço suporte Ω_H (chamaremos essa imagem de imagem teste). Para a construção desta imagem utilizamos $K = -5.760867571133922 \times 10^{-7}$ para a curvatura (subseção 3.1.2) e $R = 2719.575673736375$ para o raio que são obtidos resolvendo-se o sistema (3.2) com $P = 960$ e c dado pelo Algoritmo de Subdivisões descrito no Capítulo 3. Vejamos o resultado do modelo morfológico que estamos propondo aplicado à essa imagem.

Na geração de todas as imagens a seguir, como a parte imaginária é o desvio padrão, mudamos a escala dos tons de cinza de $\mathbb{R}$ para $[0, +\infty)$ para que 0 seja preto e $+\infty$ seja branco utilizando a seguinte função: $2X^{-1}(t) - 1$.

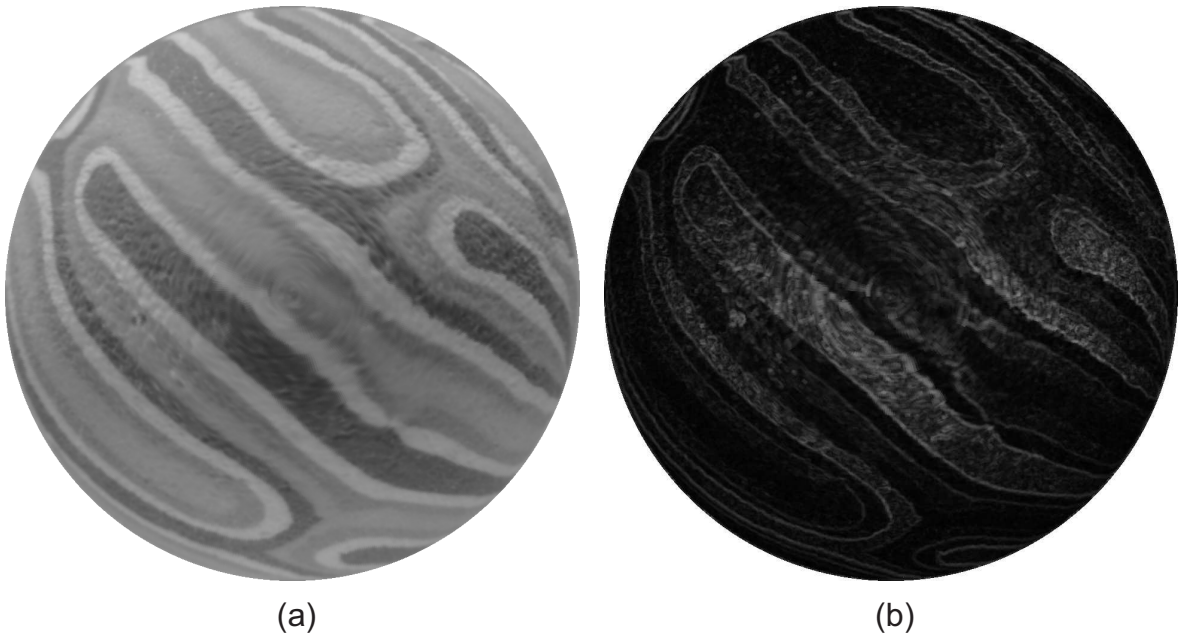


Figura 5.5: Partes real e imaginária da imagem teste.

Na Figura 5.5 podemos visualizar a imagem teste após o processo de troca de modelo de imagem de $g : \Omega_H \to \mathbb{R}$ para $f = (f_x, f_y) : \Omega_H \to \mathcal{H}^2$. Observamos que a parte real aparece meio borrada devido ao fato dessa imagem ser a média dos tons de cinza da vizinhança (que possui tamanho euclidiano variável de acordo com a posição na imagem). Além disso, a parte imaginária representa o quanto a parte real se distancia da imagem teste. Assim, percebemos que a parte imaginária sofre variações principalmente nos tons de cinza onde acontece as mudanças de cores da Figura 5.4 (a).

As figuras a seguir retratam as operações de erosão e abertura aplicados à imagem teste utilizando os 7 ordenamentos apresentados no Capítulo 4.

O ordenamento 1, que toma como ínfimo o pixel que tem a menor das coordenadas x e a menor das coordenadas y, tornou, na erosão, a parte real da imagem levemente mais escura e, na parte imaginária as regiões pretas se ampliaram (Figura 5.6).

Na abertura (Figura 5.7) percebemos que na parte real há um leve aumento de contraste em relação à imagem da Figura 5.5 (a), fazendo com que a imagem se assemelhe à imagem da Figura 5.4 (b). Já a abertura na parte imaginária não percebemos diferenças visuais significativas em relação à imagem da Figura 5.5 (b).

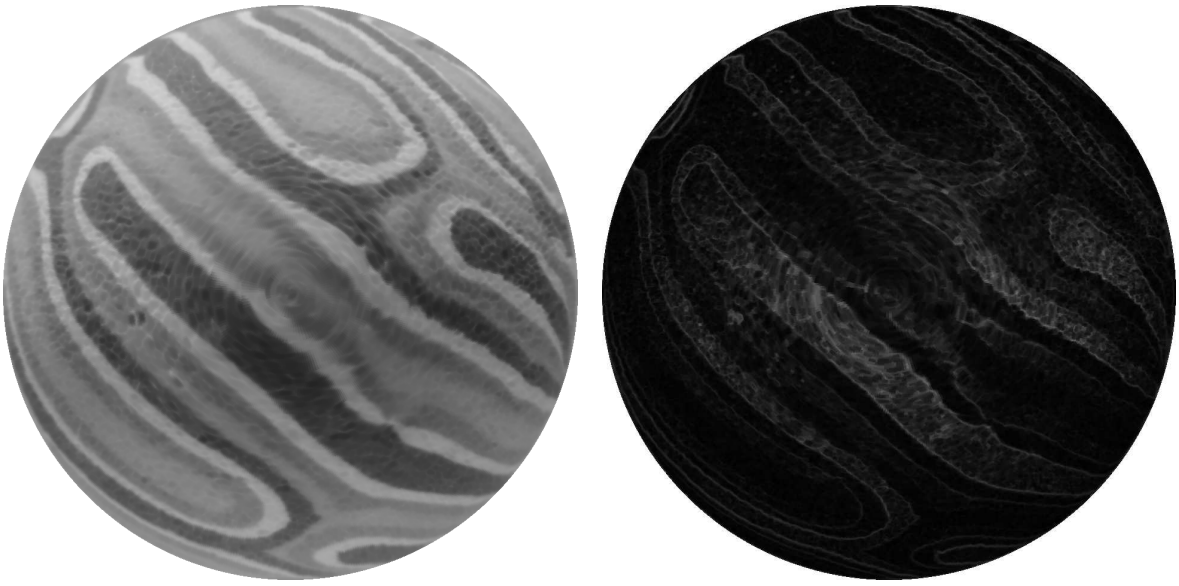


Figura 5.6: Partes real e imaginária da erosão utilizando o ordenamento 1.

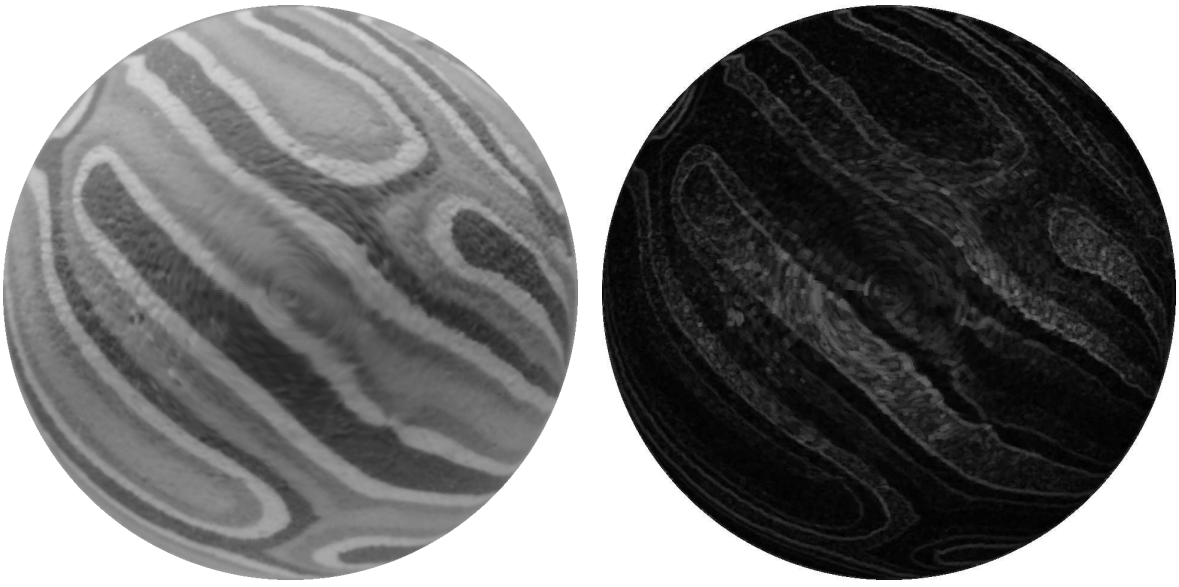
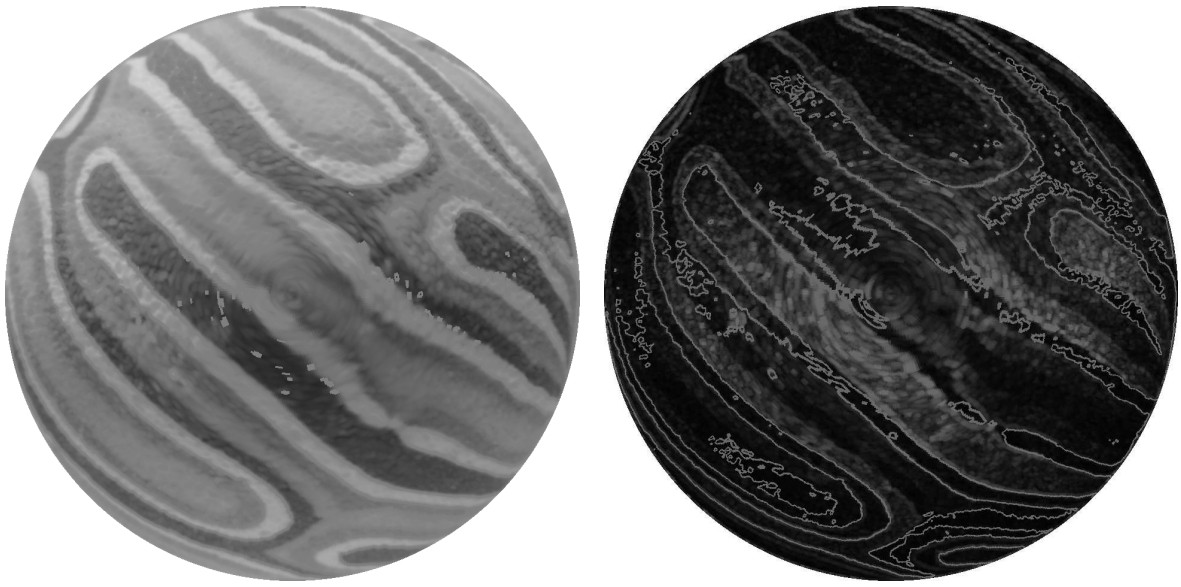


Figura 5.7: Partes real e imaginária da abertura utilizando o ordenamento 1.

O ordenamento 2 ordena os pontos por quadrantes e gera apenas pseudo-erosões e pseudo-aberturas. Na parte real surgiram faixas finas nas regiões onde ocorre as mudanças de tonalidade. Além disso, surgiram algumas pequenas regiões cinza claro em sua maioria

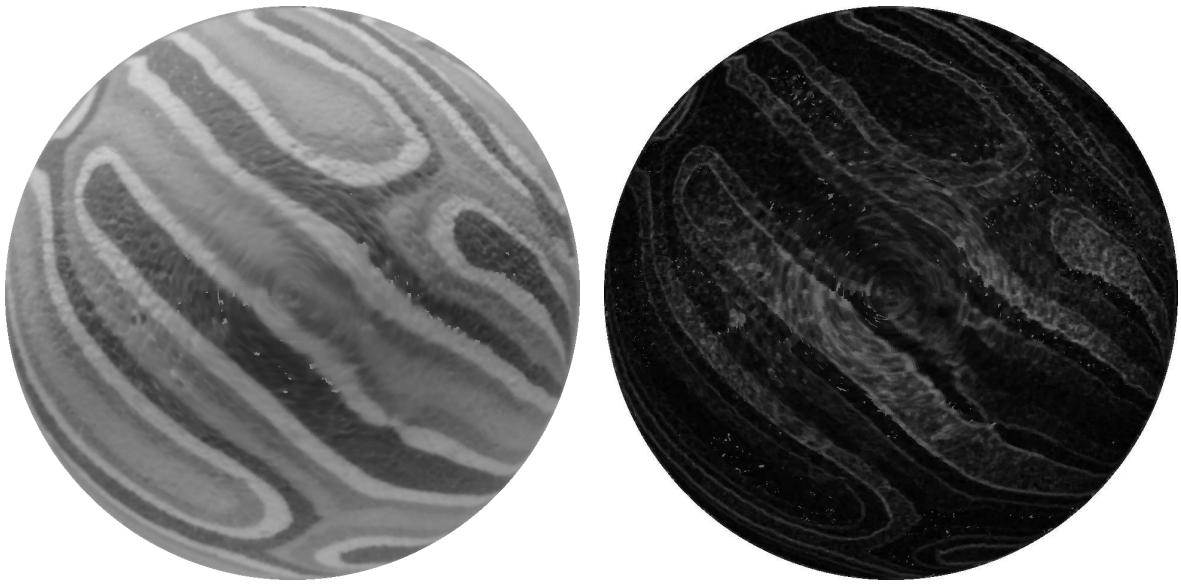
no centro da imagem. Na parte imaginária surgiram linhas cinza evidenciando as mudanças de tonalidades da imagem (Figura 5.8).

Na parte real da abertura do ordenamento 2 as faixas entre as regiões desapareceram e as regiões em cinza claro diminuiram, porém algumas ainda estão visíveis. Na parte imaginária restaram alguns pontos cinza claro parecidos com um ruído (Figura 5.9).



(a) (b)

Figura 5.8: Partes real e imaginária da erosão utilizando o ordenamento 2.



(a) (b)

Figura 5.9: Partes real e imaginária da abertura utilizando o ordenamento 2.

O ordenamento 3 é o ordenamento total que utiliza a métrica hiperbólica para ordenar os pontos determinando como ínfimo o ponto que estiver mais perto do ponto $(0, 1)$. Na parte real da erosão surgiram faixas cinza entre as mudanças de tonalidade

entretanto, essa faixa é diferente da faixa que surgiu no ordenamento anterior. Neste ordenamento a faixa se assemelha a uma faixa gerada por um spray. Na parte imaginária percebemos que as partes mais claras da imagem estão mais "borradas" (Figura 5.10).

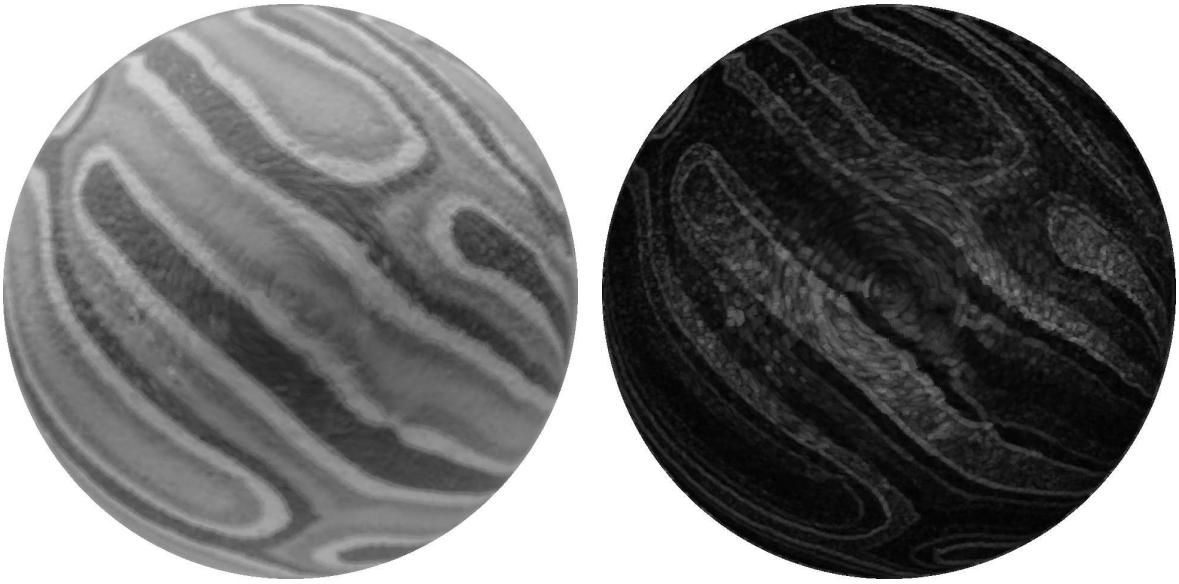


Figura 5.10: Partes real e imaginária da erosão utilizando o ordenamento 3.

A parte real da abertura dada pelo ordenamento 3 deixou alguns resquícios da faixa apresentada na erosão, restando alguns pontos (como que geradas por um leve spray) que ficam evidentes no meio da imagem. Na parte imaginária esse padrão de "leve spray" também fica evidenciado no meio da imagem (Figura 5.11).

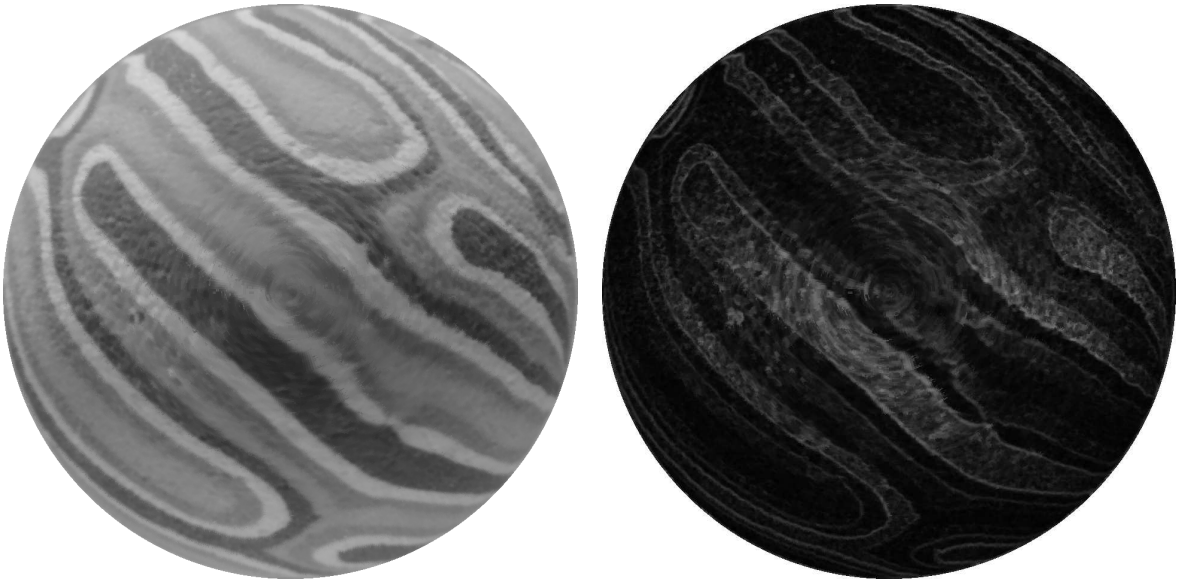
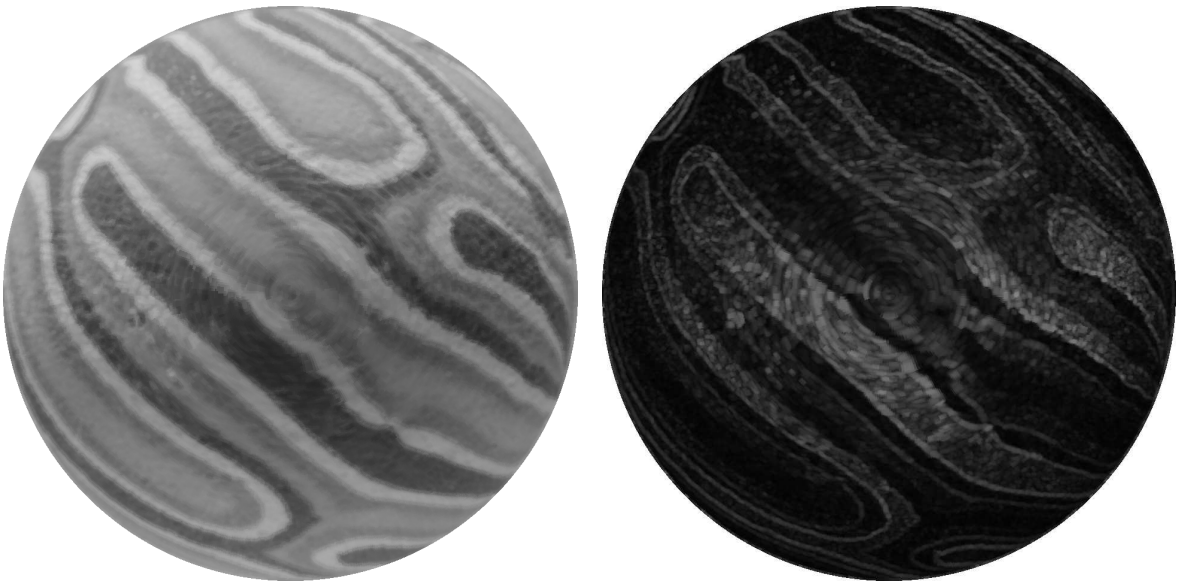


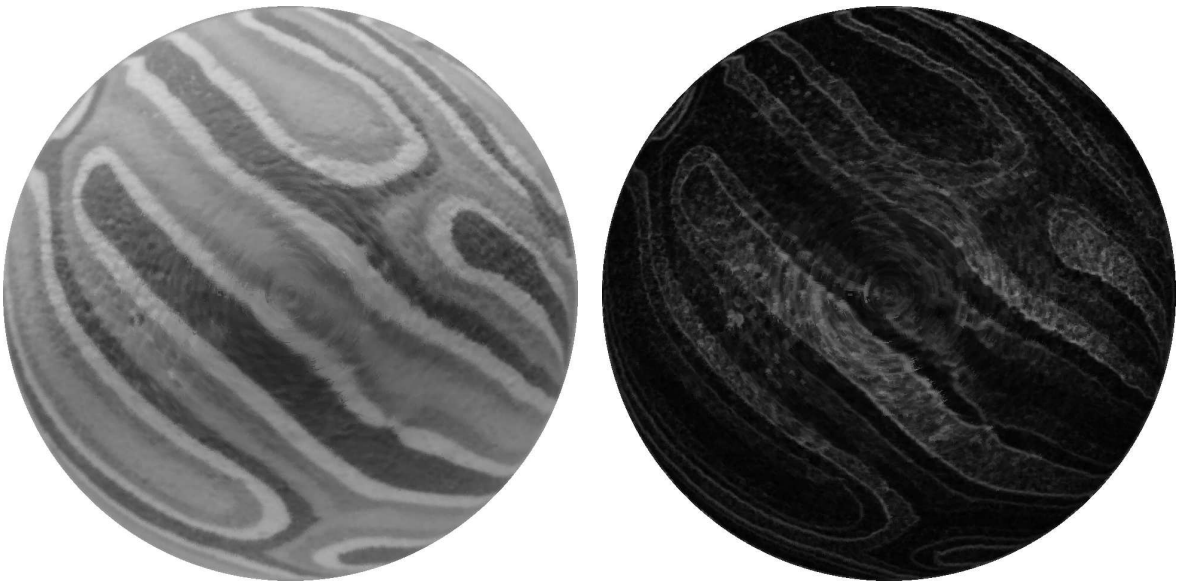
Figura 5.11: Partes real e imaginária da abertura utilizando o ordenamento 3.

O ordenamento 4 é o ordenamento total análogo ao ordenamento 3 utilizando a distância de Hellinger. Para gerar essa imagem utilizamos $\alpha = 20$. Observamos que, na parte real, ocorre o mesmo tipo de padrão das faixas que ocorreu no ordenamento 3, porém, nesse ordenamento as faixas são mais largas. Na parte imaginária percebemos que as faixas mais claras que delimitam as regiões da imagem são mais largas (Figura 5.12).



(a) (b)

Figura 5.12: Partes real e imaginária da erosão utilizando o ordenamento 4.



(a) (b)

Figura 5.13: Partes real e imaginária da abertura utilizando o ordenamento 4.

Novamente a abertura reduziu o efeito das faixas, mas não completamente, restando pontos. Na parte imaginária, além dos pontos parecidos com ruído, podemos

perceber que as linhas que delimitam a imagem ainda sao mais largas que a da parte imaginária original (Figura 5.13).

O ordenamento 5 é aquele dado pelo algoritmo MEC. Percebemos, novamente, a faixa cinza delimitando as regiões, entretanto essas faixas são mais finas e lisas do que as anteriores. Na parte imaginária percebemos um maior "borramento" no meio da imagem. Além disso, as linhas claras que delimitam a imagem estão mais evidentes (Figura 5.14).

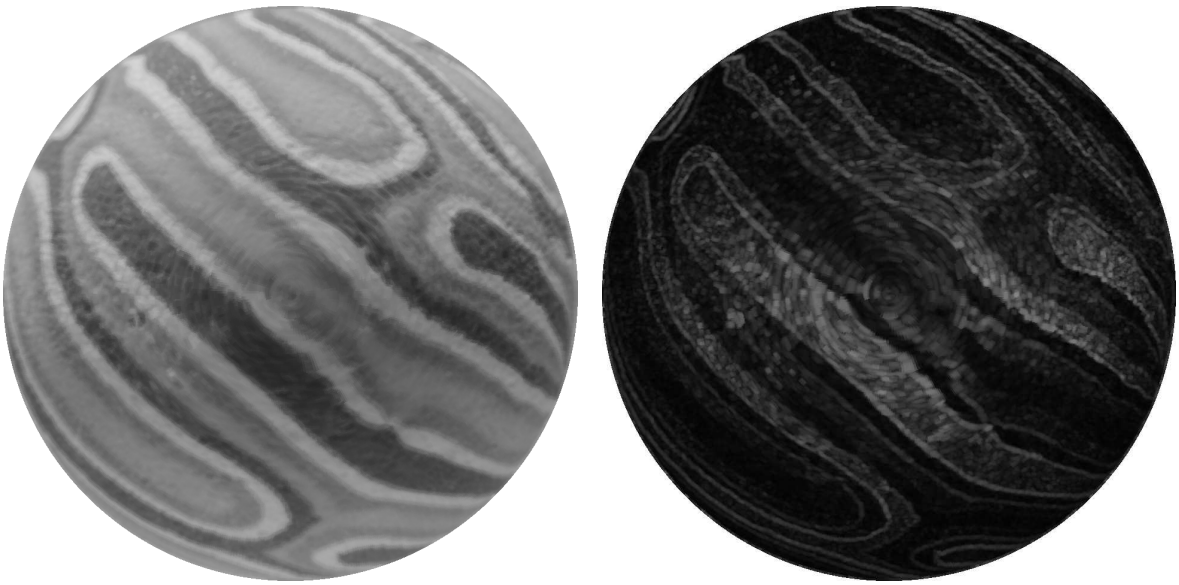


Figura 5.14: Partes real e imaginária da erosão utilizando o ordenamento 5.

A abertura desse ordenamento deixou a parte real da imagem mais escura. A parte imaginária também apresentou "borramento" no meio (Figura 5.15).

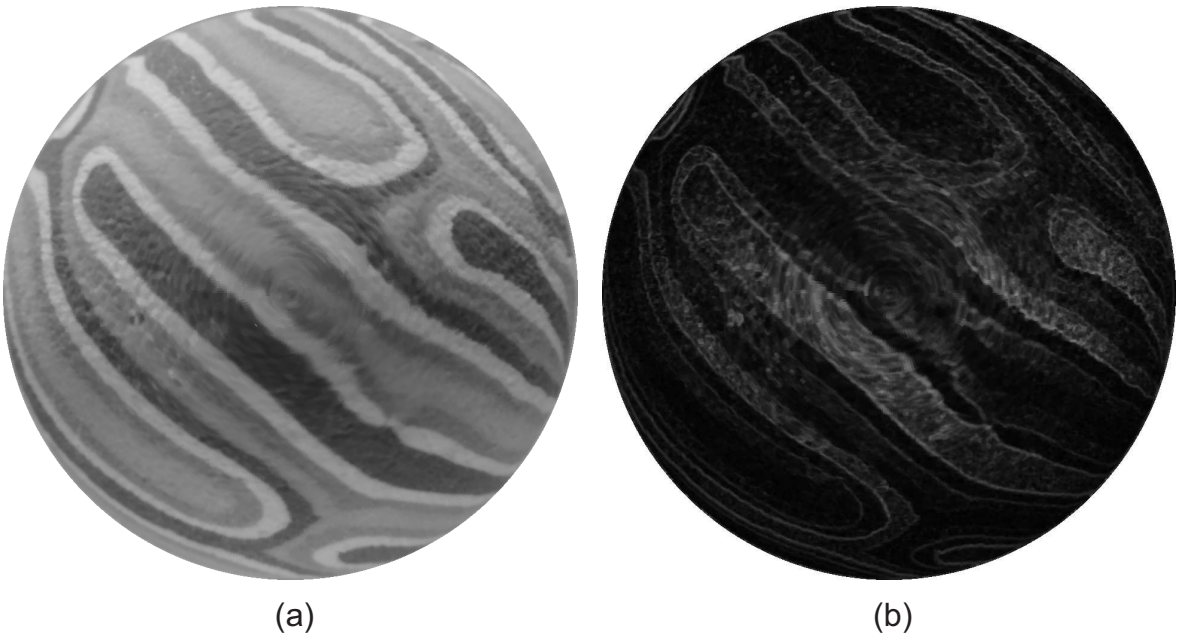
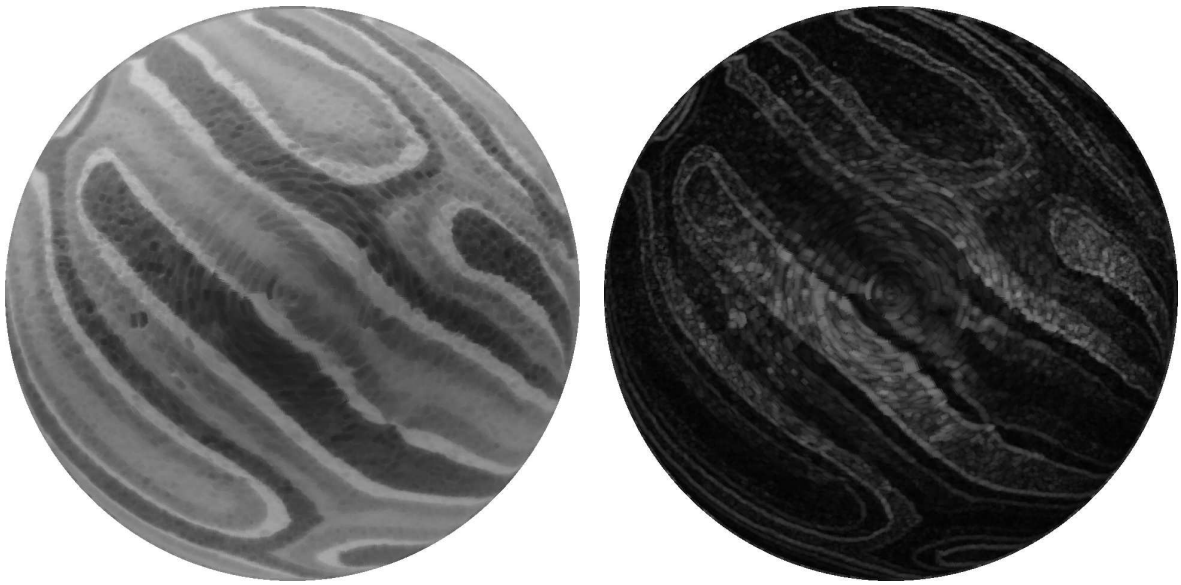


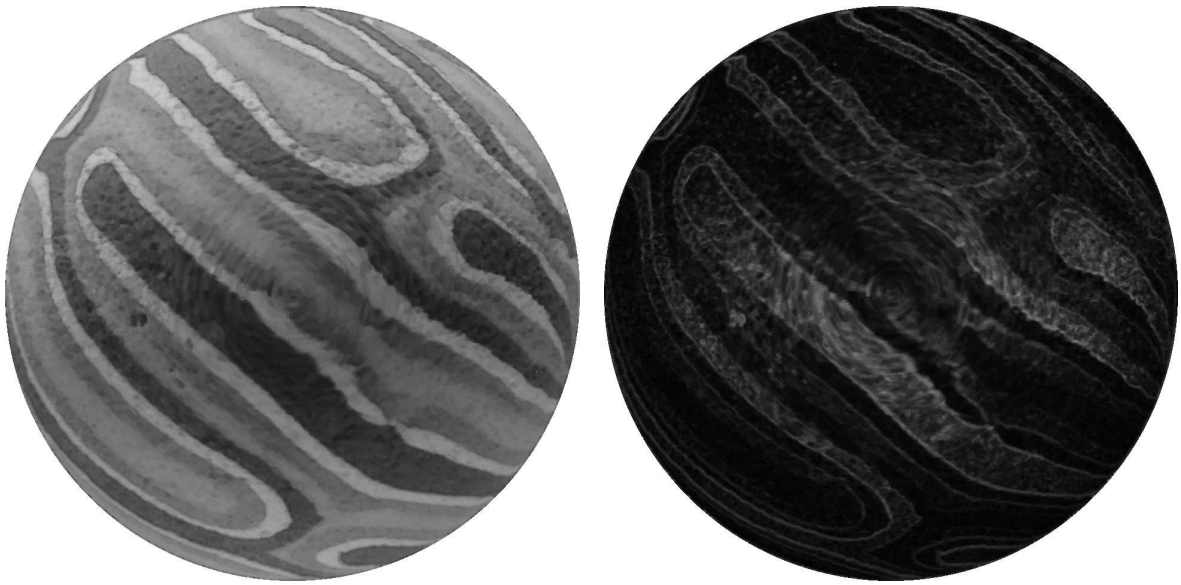
Figura 5.15: Partes real e imaginária da abertura utilizando o ordenamento 5.

O ordenamento 6 é uma modificação do ordenamento 5. Na parte real da erosão o contraste aumentou, as pequenas regiões mais escuras ficaram mais evidente e a imagem apresenta um efeito análogo à imagens submersas em água. Na parte real o meio da imagem está mais claro e as linhas que delimitam as regiões da imagem, mais evidentes (Figura 5.16).



(a) (b)

Figura 5.16: Partes real e imaginária da erosão utilizando o ordenamento 6.

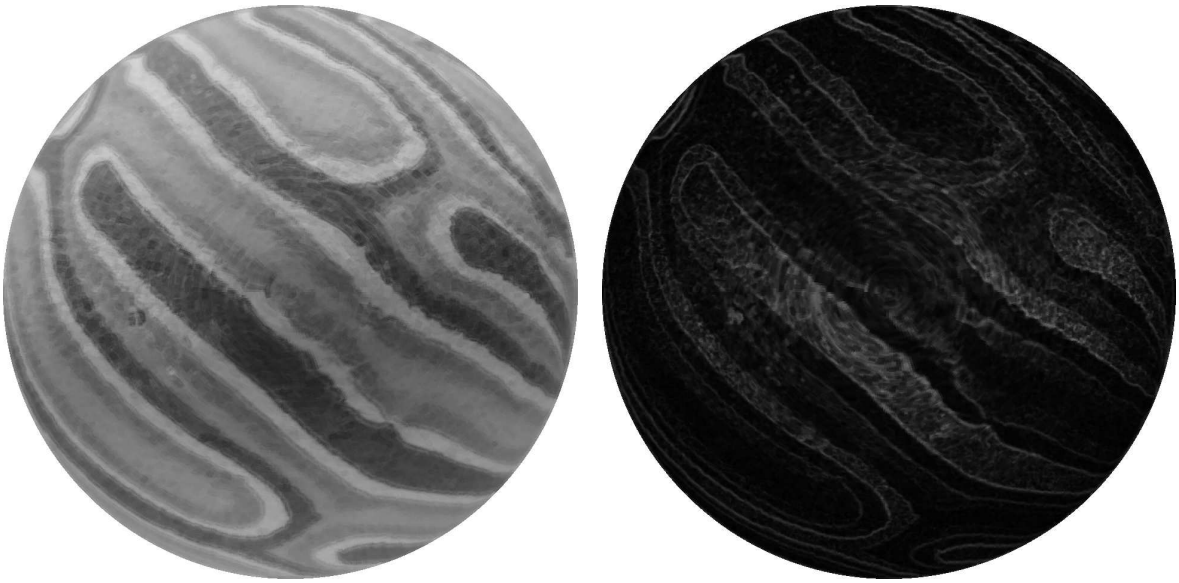


(a) (b)

Figura 5.17: Partes real e imaginária da abertura utilizando o ordenamento 6.

A parte real da imagem gerada pela abertura do ordenamento 6 apresenta um aspecto parecido com o de imagens que foram desenhadas a lápis. Além disso, surgiram

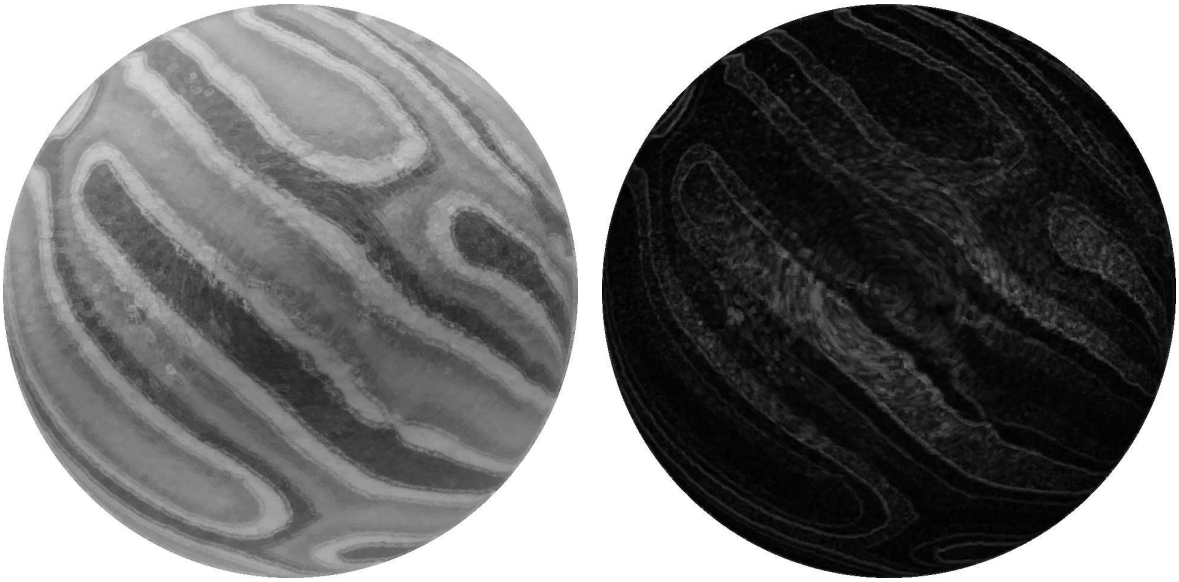
finas faixas escuras delimitando as regiões da imagem. Como um todo a imagem se tornou mais escura (Figura 5.17).



(a) (b)

Figura 5.18: Partes real e imaginária da erosão utilizando o ordenamento 7.

O ordemamento 7, que nós propomos, é o que apresenta maior distorção nas partes reais tanto da erosão quanto da abertura. Observamos, na parte real, que surgiram faixas nas regiões delimitando as imagens e, além disso, houve um maior contraste e um efeito parecido com o que ocorreu na erosão anterior, porém um pouco mais leve, semelhante a uma pintura à óleo. A parte imaginária ficou um pouco mais escura do que a original (Figura 5.18).

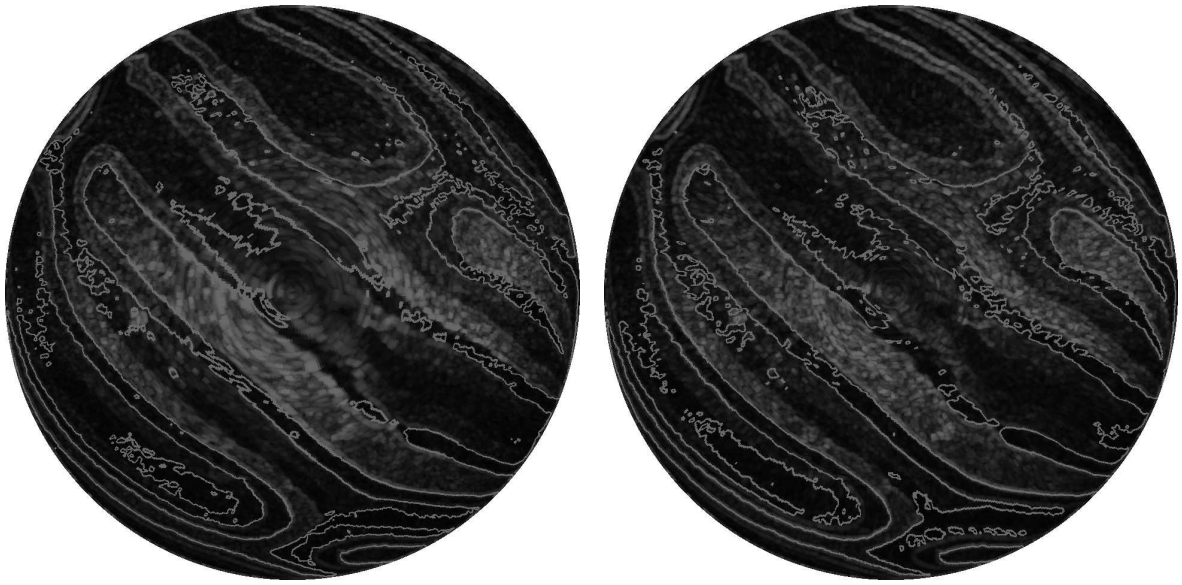


(a) (b)

Figura 5.19: Partes real e imaginária da abertura utilizando o ordenamento 7.

Na parte real da abertura as faixas se tornaram mais claras e o efeito "spray" se tornou mais evidente, tornando a imagem mais desfocada. Na parte imaginária as linhas se tornaram mais estreitas e a imagem ficou mais escura, especialmente no centro (Figura 5.19).

Repetimos a sequência de imagens dadas pelas Figuras 5.5 à 5.19 para o modelo Ω_E e observamos comportamentos semelhantes para cada ordenamento. Por exemplo, a Figura 5.20 ilustra as partes imaginárias da erosão obtida pelo ordenamento 2 utilizando os modelos Ω_H e Ω_E, respectivamente. Ambas as imagens apresentam linhas cinza contornando algumas estruturas da imagem. Observamos que, se compararmos tais imagens com as partes imaginárias originas em Ω_H e Ω_E, temos, respectivamente, PSNR de 32.716 e 32.789. Além disso, o PSNR entre as duas partes imaginárias citadas é 33.694. Ou seja, apesar de apresentarem diferenças entre elas, a alteração entre elas e as imagens originais parecem ser muito próximas. Esse comportamento pode ser percebido em todas as imagens geradas em Ω_E. Ressaltamos que não dispomos de muitas ferramentas de comparação entre os modelos, pois os métodos de comparação são baseados no modelo tradicional Ω_T. As figuras de mérito utilizadas para avaliar qualidade de imagens em Ω_T não são adequadas aos modelos propostos.



(a) (b)

Figura 5.20: Partes imaginárias da erosão utilizando o ordenamento 2 nos modelos (a) Ω_H e (b) Ω_E.

5.2.1 Sequência de vários frames

Como vimos no início do Capítulo 4, o modelo matemático de imagem pode ser aplicado considerando vários frames da mesma cena. Vejamos como é o resultado do processo neste caso.

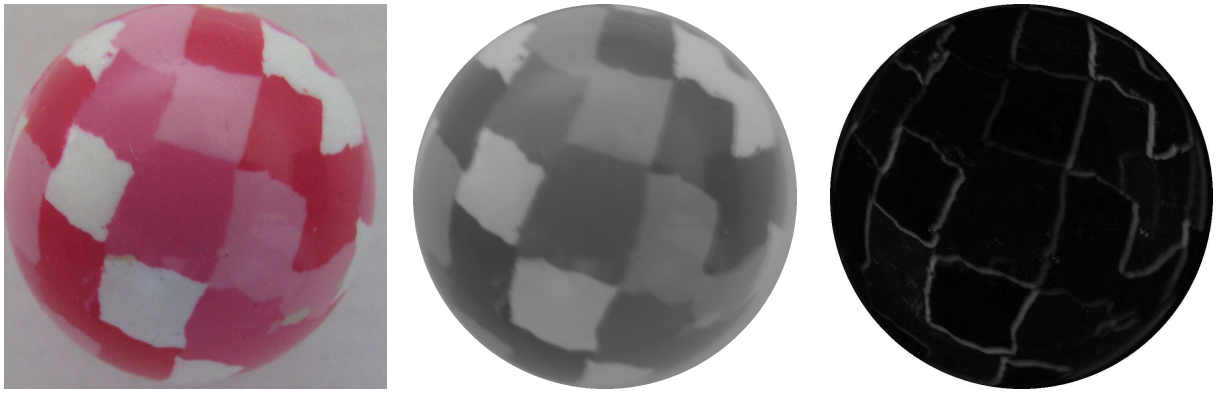


Figura 5.21: (a) Imagem I em Ω_T, (b) f_x (c) f_y.

Consideramos 10 frames I_w, $w = 1, ..., 10$, da cena dada pela imagem I em Ω_T de 960×960 pixels (Figura 5.21). Convertemos os 10 frames (denotados por Q^w) para o espaço Ω_H e, em seguida, para cada uma das posições (i, j) de um pixel de Q^w, tomamos a média e o desvio padrão do conjunto $\{Q^w[i][j], w = 1, ..., 10\}$. Denotamos por $f = (f_x, f_y) \in \Omega_H$, com tons de cinza em $\mathcal{H}^2$, a imagem resultada desse processo. As imagens I, f_x e f_y podem ser visualizadas na Figura 5.21 acima.

Em seguida, calculamos a abertura, que denotaremos por g^1, utilizando o ordenamento 3 (Figuras 5.22 (a) e (b)).

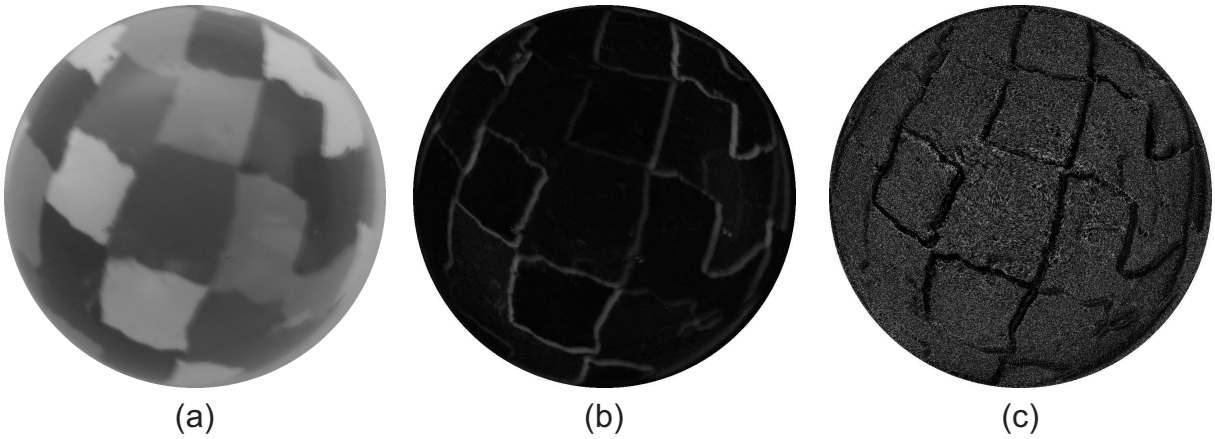


Figura 5.22: Imagens (a) g_x^1, (b) g_y^1 e (c) $dist_{\mathcal{H}^2}\left(f\left(p\right), g^1\left(p\right)\right)$.

Na Figura 5.22 (c) podemos visualizar uma imagem d_h^1, chamada imagem resíduo, que representa a distância hiperbólica ponto a ponto entre a imagem f e a abertura g^1, ou seja, $d_h^1(i, j) = dist_{\mathcal{H}^2}\left(f\left(i, j\right), g^1\left(i, j\right)\right)$, onde (i, j) representa as posições dos pixels. Notamos que, quanto mais nítida a imagem resíduo, mais próximas as imagens processadas estarão das imagens originais.

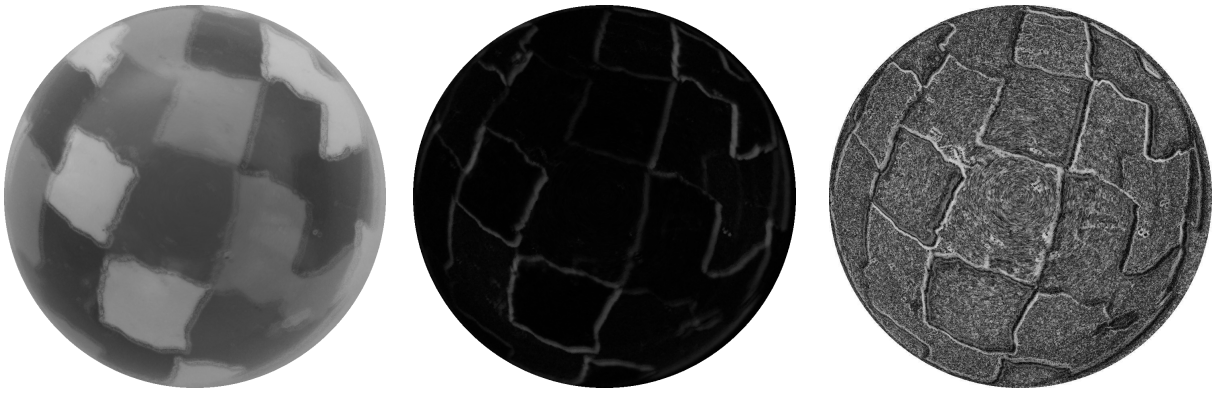


Figura 5.23: Imagens (a) g_x^2, (b) g_y^2 e (c) $dist_{\mathcal{H}^2}\left(f\left(p\right), g^2\left(p\right)\right)$.

De forma análoga, consideramos a abertura dada pelo ordenamento 7, denotada por g^2, e a distância hiperbólica ponto a ponto entre f e g^2 (Figura 5.23).

6 Conclusões e Perspectivas Futuras

6.1 Conclusões

Uma das vantagens do modelo Ω_E é a correspondência entre as áreas dos pixels esféricos de um objeto dessa natureza e os pixels polares. O desenvolvimento de modelos que atendessem esse quesito foi bastante trabalhoso, conforme relatado no Apêndice A. Sendo assim, acreditamos que o desenvolvimento que realizamos, no qual a correspondência entre áreas é exata, é uma contribuição para aplicações práticas, além da redução de cerca de 37.2% na quantidade de pixels quando comparado com o modelo cartesiano tradicional.

No modelo Ω_H temos a vantagem de, mesmo com uma estrutura métrica não euclidiana, podermos trabalhar com coordenadas inteiras para os pixels de forma correlacionada com a própria métrica do modelo, mesmo sem informação precisa do objeto retratado. Ressaltamos que desenvolvemos a expressão da métrica do modelo para qualquer curvatura negativa e raio, o que ajuda bastante nos processamentos. Em um estudo de transformadas de distâncias, conforme citado na próxima seção, a existência de uma expressão analítica para métrica do modelo é de extrema importância.

As imagens obtidas por meio de vários frames podem ser processadas utilizando-se o semiplano de Poincaré com métrica de curvatura gaussiana constante e negativa, que é um modelo para a Geometria Hiperbólica. Tais processamentos fazem uso de ordenamentos de pontos no semiplano. Sendo assim, a proposição de ordenamentos é importante nessa área e varia de acordo com as aplicações desejadas. Uma consequência do uso da Métrica de Fisher em tais processamentos nos conduz às chamadas imagens residuais, que nos modelos por nós propostos parecem ser de relevância para eventuais aplicações uma vez que esta é uma medida natural de distâncias entre distribuições gaussianas. Observamos que não é tarefa simples encontrar ordenamentos no semiplano adequados a aplicações práticas específicas.

Uma dificuldade inerente ao tipo de trabalho que desenvolvemos é, sem dúvida, os problemas envolvendo algoritmos e programação. A escolha pela linguagem Python é padrão na área de imagens devido aos inúmeros pacotes disponíveis e eficiência em tempo de execução de processamentos. Entretanto, nos espaços Ω_E e Ω_H os pacotes disponíveis não foram suficientes para nossos propósitos, o que nos obrigou ao desenvolvimento de uma rotina completa, sendo esta uma das contribuições originais de nosso trabalho. Como exemplo, citamos uma tentativa de utilização de pacotes públicos na construção de setores circulares para obtenção de pixels polares, cujo tempo de execução era demasiadamente grande, tornando inviável sua utilização. Esse problema foi amenizado em nossa proposta

de rotina, o que nos leva a propor um software completo para geração e processamento de imagens nos modelos propostos. Acreditamos que a aceitação dos métodos propostos nesse trabalho será maior após disponibilização deste software. Dentre os possíveis parâmetros a serem alterados pelo usuário do software destacamos o percentual de redução da quantidade de pixels dos modelos propostos, que é um dos parâmetros de qualidade de imagem.

Outro tópico que costuma gerar dificuldades em processamento de imagens é a mensuração de qualidade. Utilizamos o PSNR como parâmetro, mas sabemos que trata-se de uma medida não universalmente aceita. Apesar disso, encontramos valores razoáveis de PSNR para nossas imagens processadas. O estudo de novas medidas de qualidade de imagens que se adequem aos nossos modelos é um desafio, conforme citado nas seção seguinte. Ressaltamos que tanto o PSNR quanto a mudança dos parâmetros impostos nos algoritmos propostos, que influenciam na qualidade de uma imagem, podem ser mudados pelo usuário em um software como o proposto no parágrafo acima.

6.2 Perspectivas Futuras

Nesta seção indicaremos alguns assuntos que possuem conexão com o tema desta tese e que pretendemos abordar em trabalhos futuros.

Transformada de Distância

A transformada de distâncias (DT - Distance Transform) é um operador que fornece a menor distância de cada pixel a uma determinada região de interesse.

Consideremos uma imagem binária F em Ω_T. Denotemos por S o conjunto de pixels com valor 1, ou seja, o objeto da imagem. Denotemos por $\overline{S}$ o conjunto de pixels de valor 0, ou seja, o fundo da imagem. A transformada de distâncias ([18], [51] e [54]) é uma imagem M onde em cada pixel p de coordenadas (i, j) está armazenado o valor da menor distância entre o ponto (i, j) e $\overline{S}$:

$$M(p) = \min\left\{d(p, q) \mid q \in \overline{S}\right\}.$$

Observamos que M pode ser vista como uma imagem em tons de cinza.

Podemos ainda considerar os pixels do fundo como objetos da imagem e considerar o fundo da imagem como objeto de tal forma a gerar a tranformada de distâncias do fundo da imagem.

A definição de M utiliza uma distância d que pode ser uma distância definida em termos de uma métrica arbitrária. A métrica euclidiana é frequentemente a mais desejável em termos de aplicações práticas. Entretanto, as métricas discretas são, normalmente,

mais fáceis de manipular e computar. Por esse motivo, algoritmos utilizando métricas discretas são propostos desde 1966 enquanto aqueles que utilizam a métrica euclidiana (EDT - Euclidian Distance Transform) começaram a ser propostos apenas na década de 90. Os algoritmos utilizando métrica euclidiana costumam ser complicados. Assim, ainda [25] é incerto qual é o melhor algoritmo de EDT e até mesmo se os que foram propostos mais recentemente estão corretos ou não. Além disso, a performance de um algoritmo depende do conteúdo da imagem original e não apenas do seu tamanho. Outro fator a se levar em consideração é que existem muitos critérios utilizados para comparar os algoritmos.

Exemplo: A Figura 6.1 mostra uma transformada de distâncias cujo único pixel de valor 0 é o pixel central (cinza). Em (a) utilizamos a distância d_4 e em (b) a distância d_8. Em [51] é apresentada uma imagem análoga e diversos mapas computados com diferentes distâncias.

6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
3	2	1		1	2	3
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6

(a)

3	3	3	3	3	3	3
3	2	2	2	2	2	3
3	2	1	1	1	2	3
3	2	1		1	2	3
3	2	1	1	1	2	3
3	2	2	2	2	2	3
3	3	3	3	3	3	3

(b)

Figura 6.1: DT de um um pixel central utilizando (a) d_4 e (b) d_8.

A DT é um operador geométrico com grande aplicabilidade para as áreas de computação gráfica e visual, análise de imagens, reconhecimento de padrões, geometria computacional e robótica [65]. Dentre tais aplicações citemos:

- Erosão e Dilatação: A erosão e dilatação de imagens binárias, que vimos no Capítulo 2, pode ser feita utilizando o conceito de transformada de distâncias. Para o cálculo da erosão, obtemos o mapa de distâncias de uma imagem e em seguida aplicamos uma limiarização [38] que consiste em, a partir de um valor pré-determinado t, tornar 0 os valores abaixo de t e tornar 1 os valores acima de t. Para o cálculo da dilatação considera-se a troca de valores binários entre o objeto da imagem e o fundo e repetimos o processo feito na erosão.

Vejamos um exemplo. A Figura 6.2 é uma imagem onde os quadrados brancos representam o objeto da imagem e os cinza (de valor 0) representam o fundo da imagem.

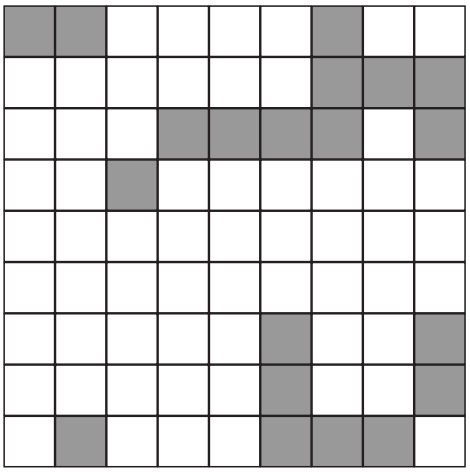


Figura 6.2: Exemplo de imagem binária para o cálculo de erosão utilizando DT.

Na Figura 6.3 (a) podemos ver a transformada de distâncias da Figura 6.2 utilizando-se a distância d_8. Fazendo a limiarização a partir do valor 1, ou seja, faremos todos os valores menores do que ou iguais a 1 se tornarem 0 e os acima serão 1. Na Figura 6.3 (b) temos a erosão resultada desse processo.

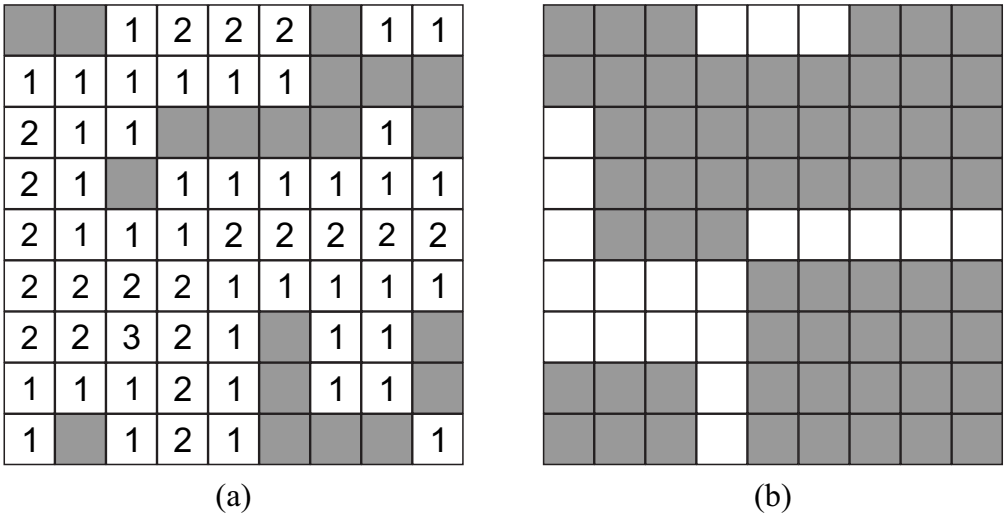


Figura 6.3: (a) DT da Figura 6.2 e (b) resultado da erosão por DT para d_8.

Analogamente ao caso anterior temos, na Figura 6.4, a transformada de distâncias utilizando a distância euclidiana d_ε e a erosão resultada deste processo. Observamos as diferenças entre as erosões causada pelas diferentes distâncias. Conforme [54], o uso da distância euclidiana fornece uma melhor distância entre os pixels já que trata de maneira diferenciada os pixels que estão em diagonal.

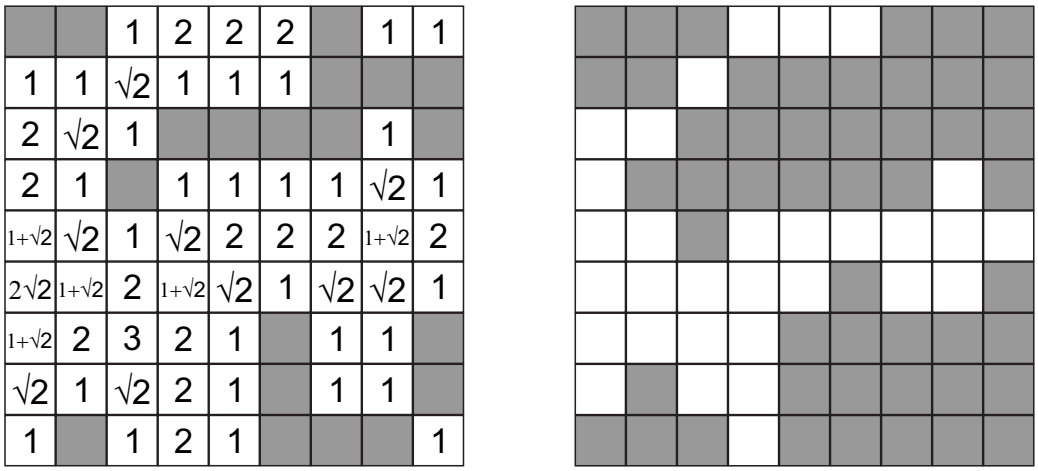


Figura 6.4: (a) DT da Figura 6.2 e (b) resultado da erosão por DT para d_ε.

- Navegação Robótica [43]: A DT pode ser utilizada por um robô para mapear a cena onde ele se encontra cujos pontos cinza (valor 0) representam os obstáculos e utilizar a DT para gerar uma rota qualquer ou encontrar o menor caminho (sem colisões) entre dois pontos.

De forma simplificada, o algoritmo consiste em gerar uma imagem binária da área a ser explorada de tal forma que os pixels de valor 1 representem a região livre e os pixels de valor 0 representem os obstáculos. A partir de tal imagem o método expande a distância em torno do pixel de destino associando valores aos pixels que representam a região livre. Observamos que o tipo de vizinhança considerada nessa aplicação é de extrema importância.

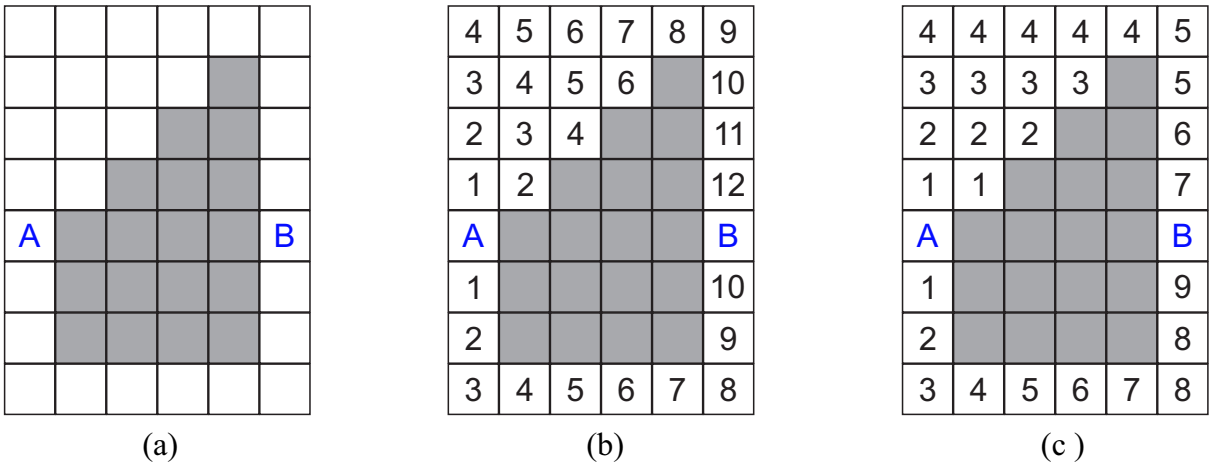


Figura 6.5: Exemplo de menor caminho entre A e B utilizando (b) d_4 e (c) d_8.

Exemplo: Na Figura 6.5 temos (a) uma imagem onde os pixels cinza representam os obstáculos e os brancos o caminho livre. Suponhamos que gostaríamos de encontrar

o menor caminho, a partir de B, até o ponto A. Assim, começando em A calculamos a distância entre A e os pontos de sua vizinhança (consideraremos que os pontos que pertençam a vizinhança de um dado ponto tem distância 1 de tal ponto). Em seguida, repetimos o processo para os pontos rotulados com 1, e assim por diante até que atinjamos o ponto B. Em (b), temos o algoritmo calculado para a vizinhança 4−conexo e, em (c), para o tipo 8−conexo. Observamos que o tipo de vizinhança influenciou no menor caminho. Em (b), o menor caminho é "por baixo" enquanto que, em (c), o menor caminho é "por cima".

- Outras aplicações da DT citadas são: esquelização, separação de objetos sobrepostos, diagramas de Voronoi, dimensão fractal e medidas de forma relacionadas à distância. A referência cita ainda que aplicações podem ser encontradas em botânica, medicina e geologia.

Em trabalhos futuros pretendemos desenvolver uma ferramenta para a transformada de distâncias para os espaços suporte Ω_H e Ω_E que propomos.

Morfologia Matemática em Imagens Coloridas

O processamento de imagens em cores é importante pelos seguintes motivos [38]:

(i) Quando se faz análise automática de imagens (reconhecimento de padrões), a cor descreve características importantes do objeto e isso pode simplificar a identificação e a segmentação. A cor ajuda, por exemplo, uma colheitadeira automática a colher os frutos maduros em uma plantação;

(ii) Quando a análise da imagem é feita com intervenção humana devemos levar em consideração que o olho humano é capaz de distinguir diversas nuances de cores e apenas poucas dezenas de tons de cinza.

As cores são diferenciadas entre si segundo três características:

- Brilho (B - Brightness): representa como a intensidade da luminosidade do objeto é percebida;

- Matiz (H - Hue): propriedade associada ao comprimento de onda predominante na combinação das várias ondas visíveis

- Saturação (S - Saturation): representa o grau de pureza da cor, ou seja, o grau de mistura do matiz original com a luz branca.

O matiz e a saturação são, normalmente, denominados de cromaticidade.

Existem diversos modelos de representação de cores e a escolha de um modelo conveniente ainda é um desafio. Os modelos permitem uma representação padronizada e geralmente são tridimensionais onde cada ponto do espaço representa uma cor. A maioria dos modelos são destinados para uso de hardware (monitores, impressoras, etc.) ou para aplicações que utilizam manipulação de cores. Dentre os modelos mais utilizados podemos citar o RGB (red, green, blue) e o HSI (hue, saturation, intensity).

O modelo RGB é um dos mais conhecidos. É conveniente considerar que os valores máximos de R, G e B estão normalizados na faixa $[0, 1]$. Sua representação é feita em coordenadas cartesianas. Mais especificamente, pode ser visualizado em um cubo onde as três cores primárias são os vértices principais (sobre os eixos cartesianos) do cubo. As demais cores são secundárias. O ponto $(0, 0, 0)$ representa o preto e o ponto $(1, 1, 1)$ representa o branco. A diagonal do cubo, que passa por esses dois pontos, representa a escala de cinza. A Figura 6.6, retirada de [38], representa o modelo RGB.

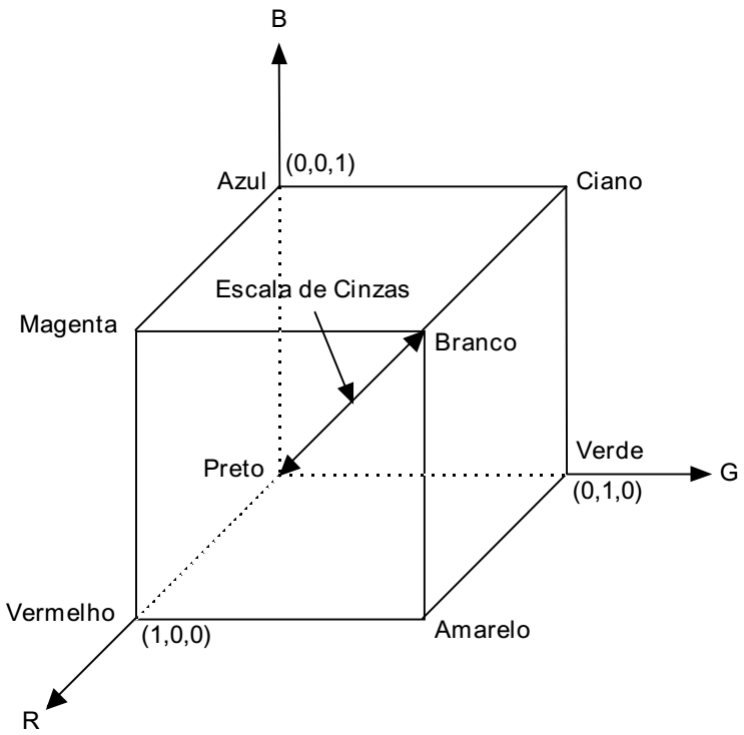


Figura 6.6: Espaço de cores RGB.

O modelo HSI permite separar os componentes de matiz, saturação e intensidade de cor em uma imagem, da maneira que o ser humano as percebe. Ele é baseado na aparência intuitiva de um artista com relação a cor, sombra e tom [13]. Esse sistema é muito utilizado em sistemas de visão artificial.

Em [38] podemos encontrar equações para mudar as cores entre os modelos.

O modelo RGB apresenta algumas desvantagens [3]: os componentes são fortemente correlacionados, interpretação humana precária, não uniformidade, etc. Além

disso, outros modelos que foram desenvolvidos para aplicações em computação gráfica, como o HSI, são inadequados para o processamento de imagens. Um modelo conveniente deve fornecer distância, ou norma, e proporcionar independência entre os componentes cromáticos e acromáticos de uma imagem. Diversos artigos trazem modelos que satisfazem esses pré-requisitos. Dentre eles: [10], [3], [34], [4], [1], [13]. O artigo [1] apresenta, inclusive, um modelo polar de representaçao de cores.

Conforme [10], os operadores morfológicos, em particular a dilatação δ (já que a erosão é definida em termos da adjunção), devem satisfazer algumas propriedades:

(1) Deve haver uma ideia de posicionamento induzido por um ordenamento;

(2) Para qualquer conjunto finito o posicionamento deve fornecer um supremo;

(3) Os operadores devem poder ser aplicados quantas vezes forem necessárias. Além disso, a definição de operadores morfológicos requer uma estrutura de reticulado completo.

Além dessas propriedades, é conveniente que:

(1) A dilatação em cores deve preservar as cores da imagem, ou seja, nenhuma cor que não estava na imagem original deve aparecer na imagem dilatada;

(2) A dilatação em cores deve ser um operador crescente, isto é, $\delta_B(I) \geqslant I$, onde I é uma imagem qualquer e B um SE. Isso implica que deve haver uma relação de ordem entre os pixels coloridos;

(3) A dilatação em cores deve ser compatível com a dilatação binária, ou seja, $\delta_B(I \oplus B') = \delta_{B' \oplus B}(I)$;

(4) A restrição para escala de cinza deve se tornar a definição padrão para a escala de cinza.

Assim como o modelo matemático de tons de cinza que apresentamos no Capítulo 4, a morfologia em cores esbarra na dificuldade de ordenamento de vetores do espaço, já que não existe um ordenamento natural para esse ambiente como ocorre em $\mathbb{R}$. Conforme [10], o ordenamento de vetores pode ser feito de diferente maneiras:

(1) considerando as cores como rótulos associados a cada pixel;

(2) utilizando o valor da cor para estabelecer um ordenamento total no espaço de cores.

A maioria das abordagens segue o tipo (2).

Existem quatro tipos de ordenamentos [10]:

(1) Ordenação marginal (M-ordenação): apenas um componente do vetor é utilizado para realizar o ordenamento. Este é o tipo mais simples de ordenação. Essa

abordagem pode apresentar alterações visuais nas cores o que pode ser inaceitável para processos que utilizem cor para reconhecimento de padrões.

(2) Ordenação condicional (C-ordenação): grupos de amostras são considerados e uma dada ordem é estabelecida entre os grupos. A ordenação *lexicográfica* constitui um exemplo muito conhecido de C-ordenação empregando potencialmente todos os componentes dos vetores disponíveis dados.

(3) Ordenação parcial (P-ordenação): a ordem é estabelecida utilizando um outro conjunto ordenado.

(4) Ordenação reduzida (R-ordenação): uma função de $\mathbb{R}^m$ em $\mathbb{R}$ é definida e os pontos são ordenados pelo valor dessa função utilizando o ordenamento usual de $\mathbb{R}$.

A R-ordenação é o tipo mais utilizado. Entretanto, tem a desvantagem de muitas vezes não ser intuitiva.

Em trabalhos futuros procuraremos estender, se possível, o modelo matemático apresentado no Capítulo 4 para imagens coloridas.

Outro modelo de espaço suporte

Em continuação a este trabalho gostaríamos de propor um espaço suporte em $\mathcal{H}^2$ para trabalhar com imagens de horizontes, tais que as informações perto do observador estão mais nítidas e, por isso, necessitam de menos pixels, enquanto as informações no horizonte precisam de mais atenção. Um exemplo desse tipo de imagem pode ser observado na Figura 6.7. Observamos que essa abordagem é parecida com a que fizemos no Capítulo 3. Além disso, a ideia inicial desse espaço suporte (Figura 6.8) é análogo ao plano polar $\rho\omega$ que vimos no Capítulo 3.

Figura 6.7: Exemplo de imagem de horizonte.

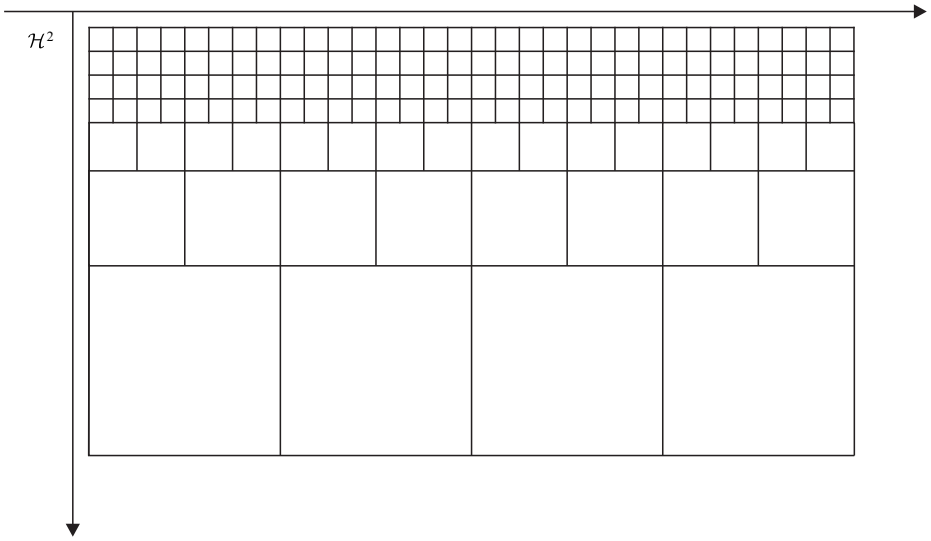


Figura 6.8: Ideia inicial de espaço suporte.

Outras aplicações

As aplicações envolvendo modelagem de imagens são bastante amplas na literatura científica e as possibilidades de abordagens específicas são extremamente diversificadas, o que faz com que o uso de novas ferramentas matemáticas como as geometrias não euclidianas e a métrica de Fisher sejam passíveis de consideração. Como exemplos de aplicações interessantes que pretendemos abordar, em continuação a este trabalho, temos a associação possível entre a geometria hiperbólica e o processamento de imagens de radares, presente no artigo [16], ou então a modelagem de retina com finalidade de aperfeiçoamento de visão robótica, como proposto em [66].

Além disso, devido a falta de consenso sobre o PSNR, gostaríamos de encontrar maneiras, adaptadas ao nosso modelo, de comparar os espaços que propomos e os modelos tradicionais. A noção de distância entre imagens é bem estabelecida na área, conforme podemos verificar em [20], entretanto, o estudo e introdução de novas métricas adequadas aos nossos modelos podem representar um caminho de pesquisas dentro da área de Geometria da Informação.

Referências

[1] J. Angulo. Morphological colour operators in totally ordered lattices based on distances: Application to image filtering, enhancement and analysis. *Computer Vision and Image Understanding*, 107:56–73, 2007.

[2] J. Angulo. Polar modelling and segmentation of genomic microarray spots using mathematical morphology. *Image Analysis and Stereology*, 27(2):107–124, 2008.

[3] J. Angulo and J. Serra. Mathematical morphology in color sspace applied to the analysis of cartographic images. In INP-Mexico, editor, *International Workshop Semantic Processing of Spartial Data*, pages 56–66, 2003.

[4] J. Angulo and J. Serra. Modelling and segmentation of color images in polar representation. *Image and Vision Computing*, 25:475–495, 2007.

[5] J. Angulo and S. Velasco-Forero. Complete lattice structure of poincaré upper-half plane and mathematical morphology for hyperbolic-valued images. In *Geometric Science of Information*, volume 8085, pages 535–542. Springer Berlin Heidelberg, August 2013.

[6] J. Angulo and S. Velasco-Forero. Mathematical morphology for real-valued images on riemannian manifolds. In *Mathematical Morphology and Its Applications to Signal and Image Processing*, volume 7883, pages 279–291. Springer Berlin Heidelberg, May 2013.

[7] J. Angulo and S. Velasco-Forero. Morphological processing of univariate gaussian distribution-valued images based on poincaré upper-half plane representation. In F. Nielsen, editor, *Geometric Theory of Information*, pages 331–366. Springer International Publishing, 2014.

[8] J. Angulo and S. Velasco-Forero. Riemannian mathematical morphology. *Pattern Recognition Letters*, 47:93–101, October 2014.

[9] A. Beardon. *The Geometry of Discret Groups*. Springer-Verlag, New York, 1982.

[10] X. Benavent, E. Dura, F. Vergara, and J. Domingo. Mathematical morphology for color image: An image-dependent approach. *Mathematical Problems in Engineering*, 2012, 2012.

[11] A. Bovik. *The Essential Guide to Image Processing*. Elsevier, 2009.

[12] A. Caliman, M. Ivanovici, and N. Richard. Probabilistic pseudo-morphology for grayscale and color images. *Pattern Recognition*, 47:721–735, 2004.

[13] A. L. B. Candeias. *Aplicação da morfologia matemática à análise de imagens de sensoriamento remoto*. PhD thesis, Instituto Nacional de Pesquisas Espaciais, São José dos Campos, Fevereiro 1997.

[14] E. J. Candès and M. B. Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 2008.

[15] M. P. Carmo. *Geometria Diferencial de Curvas e Superfícies*. IMPA, Rio de Janeiro, 2008.

[16] C. Chaire and F. Barbaresco. New generation doppler radar processing: Ultra-fast robust doppler spectrum barycentre computation scheme in poincaré's unit disk. In *European Radar Conference*, 2010.

[17] S. I. R. Costa, S. A. Santos, and J. E. Strapasson. Fisher information distance: a geometric reading. *Discrete Applied Mathematics*, 197:59–69, 2015.

[18] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14(3):227–248, 1980.

[19] I. Debusschere et al. A retinal ccd sensor for fast 2d shape recognition and tracking. *Sensors and Actuators A: Physical*, 22(1):456–460, 1989.

[20] M. M. Deza and E. Deza. *Encyclopedia of Distances*. Springer, 3 edition, 2014.

[21] E. R. Dougherty and R. A. Lotufo. *Hands-on Morphological Image Processing*, volume TT 59. Tutorial texts in optical engineering, 2003.

[22] A. B. Downey. *Think Python: How to Think Like a Computer Scientist*. O'Reilly, 2 edition, 2015.

[23] M. F. Duarte et al. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25:83 – 91, march 2008.

[24] K. Egiazarian, A. Foi, and V. Katkovnik. Compressed sensing image reconstruction via recursive spartially adaptive filtering. *IEEE International Conference on Image Processing*, 2007.

[25] R. Fabbri, L. F. Costa, J. C. Toreli, and O. M. Bruno. 2d euclidean distance transform algorithms: A comparative survey. *ACM Computing Surveys*, 40(1), 2008.

[26] G. M. Faustino. Efeitos de mosaico para imagens. Mestrado em matemática, IMPA, 2005.

[27] R. C. Fernandes Junior, A. Kanaan, and J. M. S. M. Gomes. *As ferramentas do Astrônomo: O que medimos, como medimos e como aprendemos.* Observatórios virtuais, Florianópolis, 2002.

[28] M. Fornasier and H. Rauhut. Compressive sensing. In O. Scherzer, editor, *Handbook of Mathematical Methods in Imaging*, pages 187–228. Springer New York, 2011.

[29] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Pearson, New Jersey, 3 edition, 2008.

[30] H. J. A. M. Heijmans. *Morphological Image Operators.* Academic Press, Boston, 1994.

[31] S. Katok. *Fuchsian Groups.* Chicago: The University of Chicago Press, 1992.

[32] H. Kruegle. *CCTV Surveillance: Analog and Digital Video Practices and Technology.* Elsevier, 2 edition, 2007.

[33] N. Kulkarni. Compressive sensing for computer vision and image processing. Master's thesis, Arizona State University, 2011.

[34] P. Lambert and J. Chanussot. Extending mathematical morphology to color image processing. In *1st International Conference on Color in Graphics and Image Processing*, pages 158–163, Saint Etienne, France, 2000.

[35] M. A. Luengo-Oroz and J. Angulo. Mathematical morphology in polar-logarithmic coordinates. application to erythrocyte shape analysis. *Pattern Recognition and Image Analysis*, 3523, 2005.

[36] M. A. Luengo-Oroz and J. Angulo. Cyclic mathematical morphology in polar-logarithmic representation. *IEEE Transactions on Image Processing*, 18(5):1090–1096, May 2009.

[37] M. Lutz and D. Ascher. *Aprendendo Python: Programação Orientada a Objetos.* Bookman, Porto Alegre, 2 edition, 2007.

[38] O. Marques Filho and H. Vieira Neto. *Processamento Digital de Imagens.* Brasport, Rio de Janeiro, 1999.

[39] N. N. C. Menezes. *Introdução à Programação com Python.* Novatec, 2 edition, 2014.

[40] P. L. Meyer. *Probabilidade e Aplicações à Estatística.* LTC, Rio de Janeiro, 2 edition, 1965.

[41] F. Nielsen and F. Barbaresco, editors. *Geometric Science of Information*, Lecture notes in computer science, 0302-9743 ; 8085 LNCS sublibrary. SL 6, Image processing, computer vision, pattern recognition, and graphics, Paris, France, August 2013. Heidelberg : Springer.

[42] F. Pardo, B. Dierickx, and D. Sheffer. Space-variant nonorthogonal structure cmos image sensor desing. *IEEE Journal of Solid-State Circuits*, 33(6):842–849, June 1998.

[43] M. J. Pinto, F. L. L. Medeiros, and M. M. de Marchi. Transformada de distância aplicada em cenários de vigilância aérea. *XLVII Simpósio Brasileiro de Pesquisa Operacional*, 2015.

[44] J. P. S. Porto. Geometria da informação: métrica de fisher. Mestrado em matemática aplicada, Universidade Estadual de Campinas, 2013.

[45] W. K. Pratt. *Digital Image Processing: PIKS Scientific Inside*. Wiley Interscience, New Jersey, 4 edition, 2007.

[46] S. Qaisar et al. Compressive sensing: From theory to applications, a survey. *Journal of Communications and Networks*, 15:443 – 456, 2013.

[47] P. B. Ramkumar. *A Study of Morphological Operators with Applications*. PhD thesis, Cochin University of Science and Technology, 2011.

[48] C. R. Rao. Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Math. Soc.*, 37:81–91, 1945.

[49] L. B. Rodrigues. Reticulados hiperbólicos em espaços quocientes mergulhados isometricamente em espaços euclidianos. Mestrado em matemática, Universidade Federal de Uberlândia, 2010.

[50] L. B. Rodrigues, S. I. R. Costa, and E. Agustini. Representação de imagens em espaço suporte hiperbólico. *XXXIII Simpósio Brasileiro de Telecomunicações*, 2015.

[51] A. Rosenfeld and J. L. Pfaltz. Digital functions on digital pictures. *Pattern Recognition*, 1:33–61, 1968.

[52] S. Ross. *Probabilidade: Um curso moderno com aplicações*. Bookman, Porto Alegre, 8 edition, 2010.

[53] G. G. Roussas. *A Course in Mathematical Statistics*. Press, California, 2 edition, 1997.

[54] J. C. Russ. *The Image Processing Handbook*. CRC Press, Boca Raton, 6 edition, 2011.

[55] G. Sandini et al. A retina-like cmos sensor and its applications. *Proceedings of: 1st IEEE SAM Workshop*, 2000.

[56] K. Schindler and H. Bischof. The epipolar geometry of the log-polar image plane. *Proceedings of the 17th International Conference on Pattern Recognition*, 4:40–43, 2004.

[57] J. Serra. *Lecture Notes on Morphological Operators*. First French-Nordic Summer Course in Mathematics, Uppsala University, Suécia, Junho 2001.

[58] L. G. Shapiro and G. C. Stockman. *Computer Vision*. Prentice Hall, New Jersey, 2001.

[59] P. Soille. *Morphological Image Analysis*. Springer, New York, 1 edition, 1999.

[60] C. Solomon and T. Breckon. *Fundamentals of Digital Image Processing: A Practical Approach with Examples in Matlab*. Wiley-Blackwell, 2011.

[61] I. Stanković. Recovery of images with missing pixels using a gradient compressive sensing algorithm. *arXiv:1407.3695, 7p.*, 2014.

[62] D. Stirzaker. *Elementary Probability*. Cambridge, New York, 2 edition, 2003.

[63] A. Thompson. Compressive single-pixel imaging. In *Proceedings of the 2nd IMA Conference on Mathematics in Defence*, 2011.

[64] M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *IEEE Transactions on PAMI*, 14(4):401–410, 1993.

[65] J. C. Torelli. Implementação paralela da transformada de distância euclidiana exata. Master's thesis, Universidade de São Paulo - ICMC, 2005.

[66] A. Trehub. Neuronal models for cognitive processes: Networks for learning, perceptions and imagination. *J. theor. Bio.*, 65:141–169, 1977.

[67] C. R. F. Weiman. Polar exponential sensor arrays unify iconic and hough space representation. *SPIE Int. Conf. on Intelligent Robots and Computer Vision VIII: Algorithms and Techniques*, 1990.

[68] E. Welzl. Smalles enclosing disks (balls and ellipsoids). *Lecture Notes in Computer Science*, 555:359–370, 1991.

[69] R. Wodnicki, G. W. Roberts, and M. D. Levine. A foveated image sensor in standard cmos technology. *Proceedings of custon integrate circuit conference*, pages 357–360, 1995.

Apêndices

APÊNDICE A – Histórico do desenvolvimento de Ω_H e Ω_E

Neste apêndice faremos um breve resumo de algumas das tentativas que fizemos para propormos os espaço Ω_H e Ω_E.

A primeira ideia que surgiu para distribuir os pixels no disco foi a de trabalhar com círculos concêntricos e divisões radiais. Além disso, gostaríamos que o espaço suporte proposto contivesse $0.628P^2$ pixels. Mantivemos estas ideias iniciais durante todo o trabalho. Como gostaríamos de ter mais pixels no bordo do disco, a primeira ideia que tivemos foi começarmos com uma determinada quantidade de divisões no disco central e a partir deste dobrar a quantidade de pixel a cada coroa subsequente. Entretanto, como podemos ver na Figura A.1, os pixels próximos ao bordo ficaram muito finos o que é um problema pois, como vimos no Capítulo 3 iremos sobrepor o espaço proposto com Ω_T.

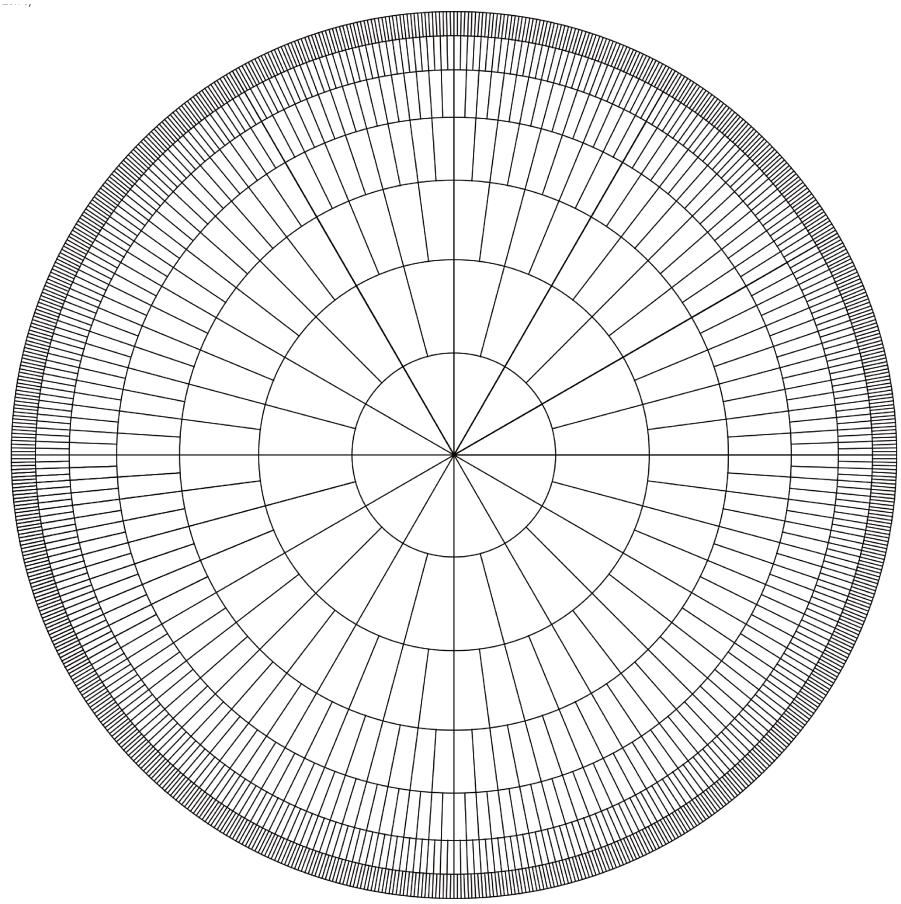


Figura A.1: Quantidade de pixels dobrando a cada coroa.

Para evitar os pixels finos tivemos a idéia (que ainda utilizamos) de forçar

os pixels da última coroa a se assemelhar, em "altura" aos pixels de Ω_T e dividimos todas as coroas igualmente. A quantidade de coroas foram contabilizadas levando-se em consideração a quantidade de pixels que gostaríamos que o espaço tivesse. Mas, como podemos ver na Figura A.2, agora foram os pixels do centro que ficaram finos demais.

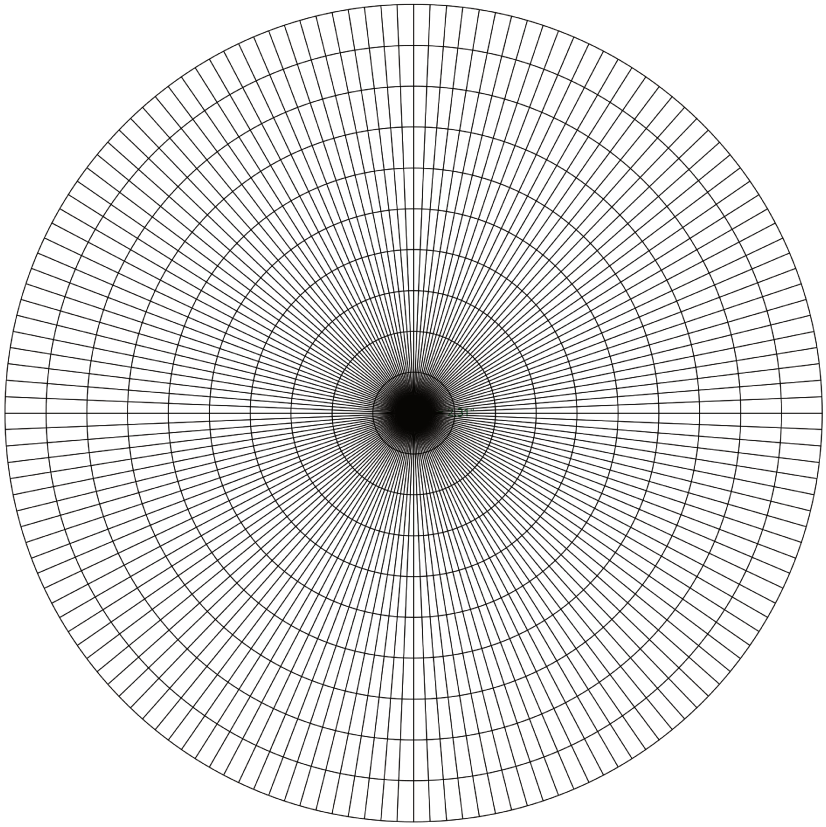


Figura A.2: Pixels da última coroa com altura similares aos de Ω_T.

Optamos então, por unir as duas abordagens. Construímos uma camada interna onde as divisões das coroas seriam como na primeira abordagem e uma camada externa onde as divisões das coroas seriam como na segunda abordagem. Daí, surgiu a necessidade de decidir qual a quantidade de pixels que ficaria na coroa externa e qual a quantidade ficaria na coroa interna. Lembrando que gostaríamos que uma maior quantidade de pixels ficasse na borda da imagem fizemos alguns testes e, aparentemente, a proporção de 75% dos pixels na coroa externa parece ser razoável. Nas figuras abaixo podemos visualizar as diferentes proporções de divisão para uma imagem de 50×50 pixels em Ω_T.

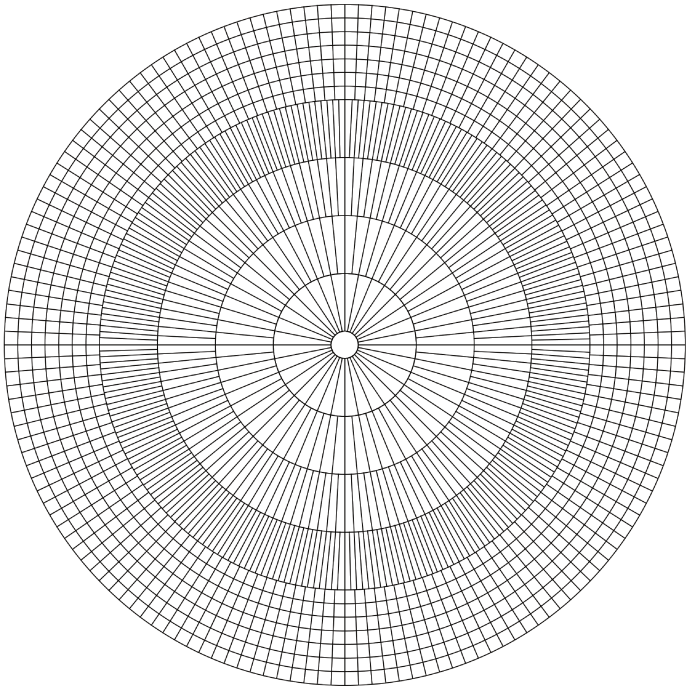


Figura A.3: Dois terços dos pixels na camada externa.

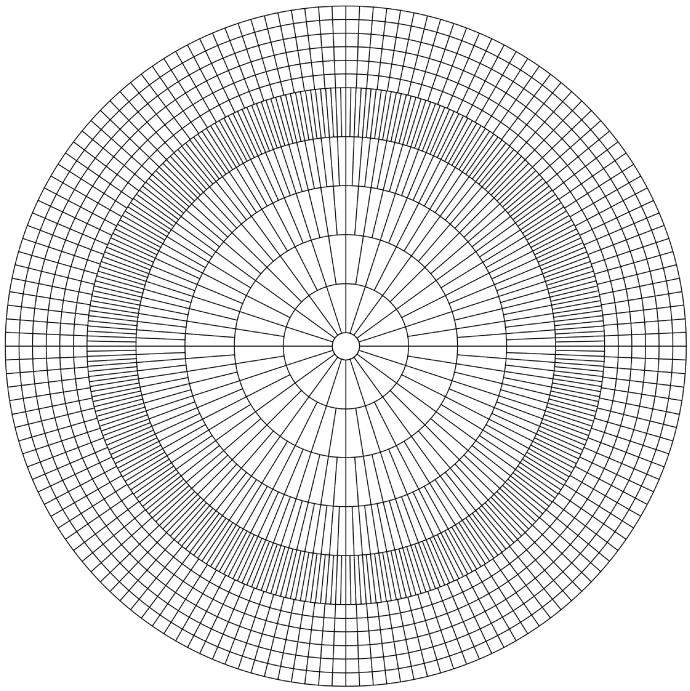


Figura A.4: Três quintos dos pixels na camada externa.

Na Figura A.3 temos 2/3 dos pixels na coroa externa, na Figura A.4 temos 3/5 e na Figura A.5 temos 3/4. Observamos que o espaço da Figura A.5 aparenta ser melhor distribuído do que as outras proporções propostas.

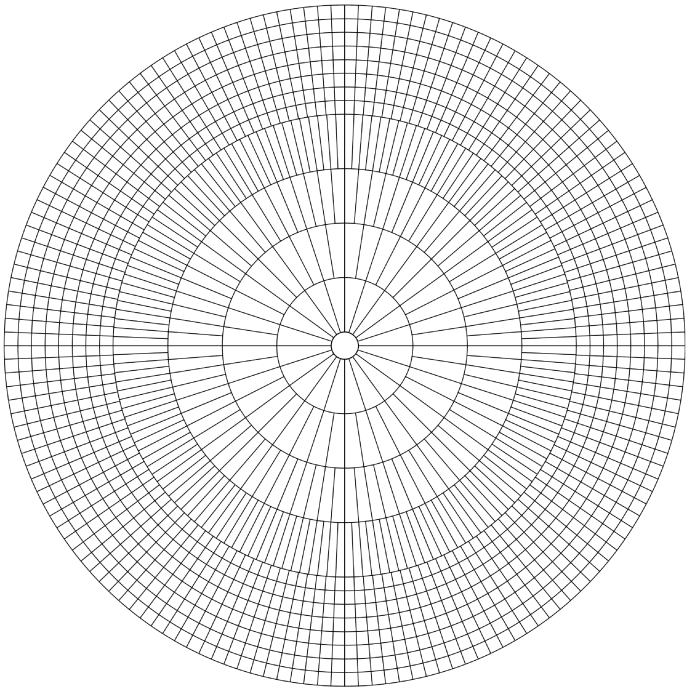


Figura A.5: Três quartos dos pixels na camada externa.

Porém, ao trabalhar com imagens com uma quantidade maior de pixels observamos que a última coroa da camada interna estava ficando com pixels muito finos, como pode ser observado na Figura A.6 o exemplo de uma imagem que tem 200×200 pixels em Ω_T.

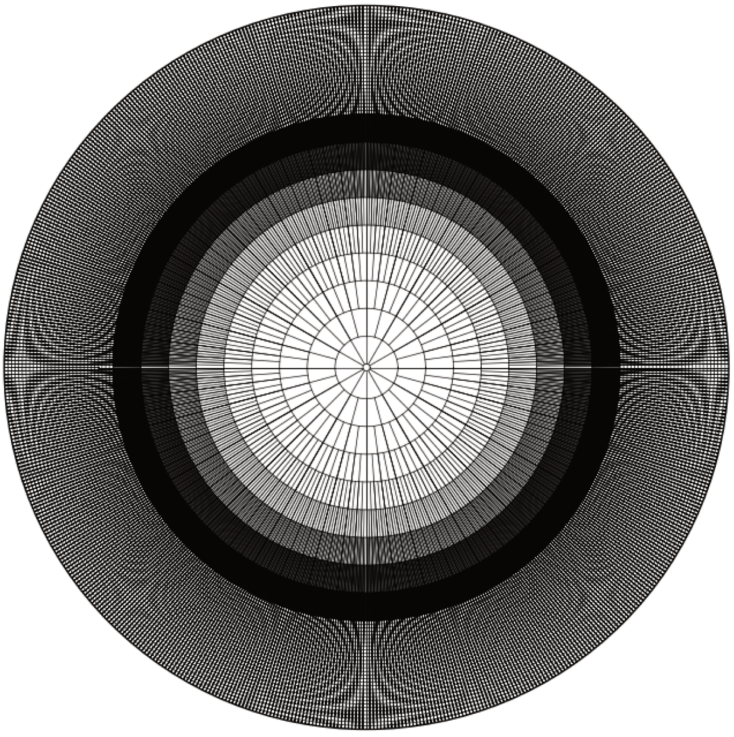


Figura A.6: Pixels da última coroa da camada interna muito finos.

Assim, incluímos um teste que excluía a última coroa da camada interna caso seus pixels fossem muito finos e aumentava uma coroa na camada externa. Podemos ver na Figura A.7 como fica o espaço suporte proposto para uma imagem que em Ω_T tem 50×50 pixels.

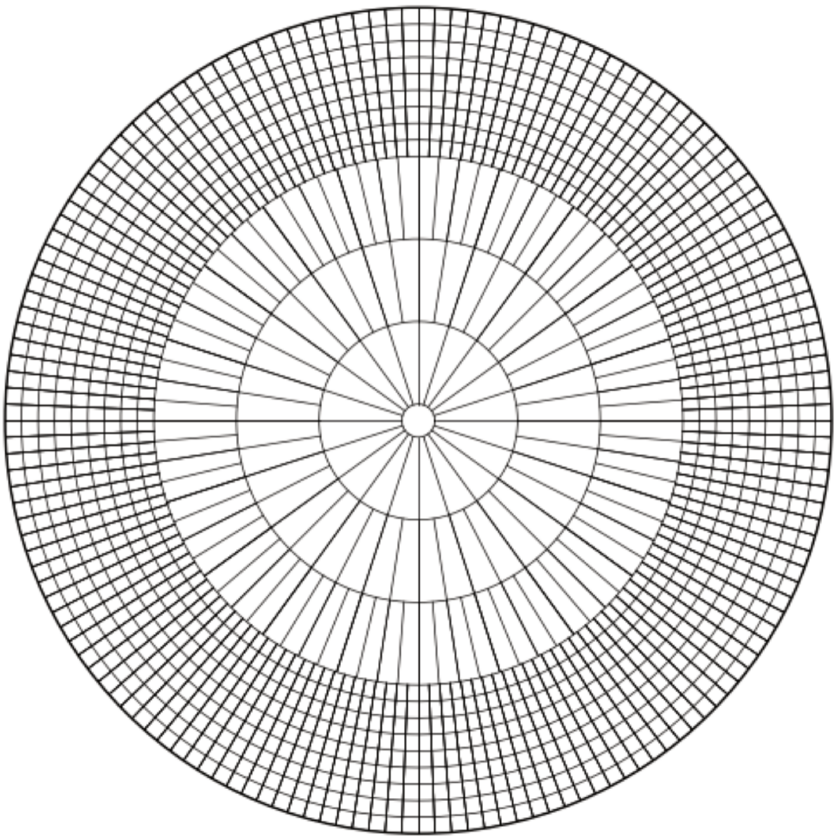


Figura A.7: Abordagem em camadas com teste para evitar pixels estreitos.

Até este momento havíamos trabalhando apenas com a métrica euclidiana no disco. Daí, surgiu a idéia de, com a mesma distribuição de pixels da Figura A.7, utilizar a métrica hiperbólica. Na Figura A.8 podemos perceber que a distribuição dos raios das circunferências é mais harmônica do que a dada pela Figura A.7.

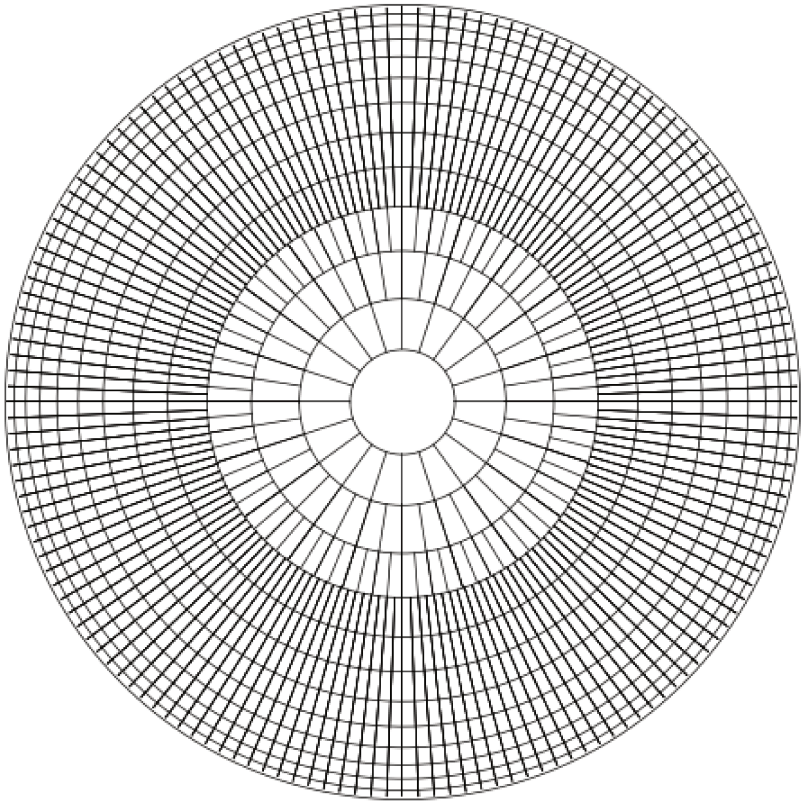


Figura A.8: Primeira abordagem utilizando métrica hiperbólica.

Veja na Figura A.9 a diferença de resolução entre uma imagem com uma proposta de divisão euclidiana dos raios (à esquerda) e uma imagem que utilizamos a métrica hiperbólica (à direita).

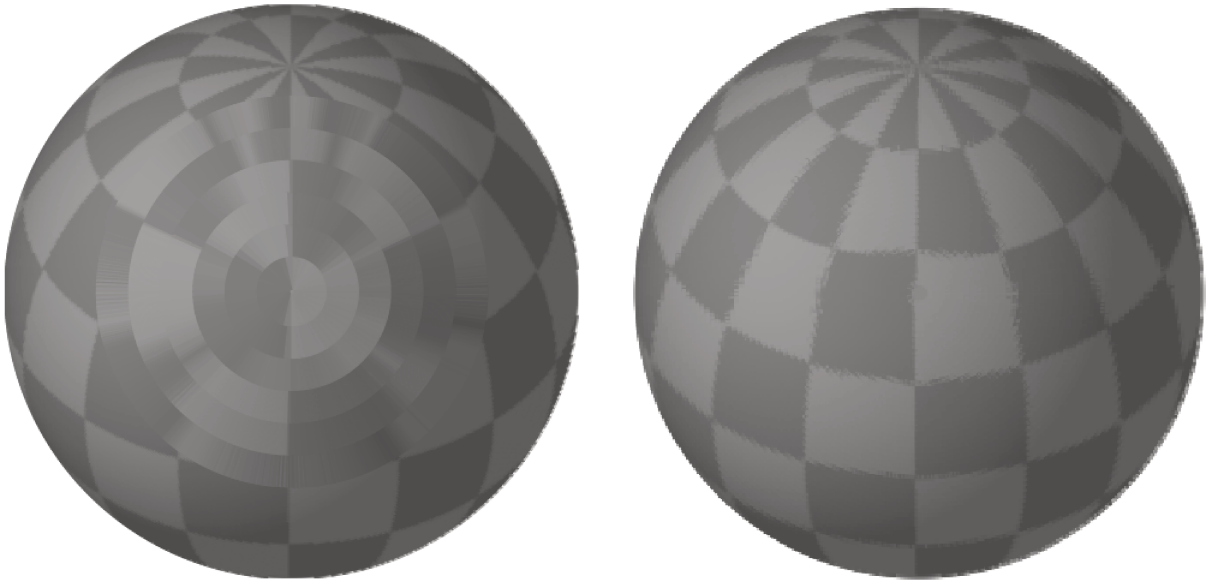


Figura A.9: Figuras geradas utilizando métrica euclidiana (esquerda) e métrica hiperbólica (direita).

Após utilizar este modelo de espaço suporte por bastante tempo percebemos que, na verdade, quando incluímos o teste para evitar pixels finos na primeira camada estávamos diminuindo muito a quantidade de pixels nesta camada, pois várias coroas estavam ficando muito finas, nos afastando da proposta inicial de manter 25% dos pixels iniciais na camada interna. Além disso, como as divisões angulares são calculadas de maneiras distintas para as duas camadas, acontecia frequentemente de as duas camadas não se "alinharem", formando assim "degraus". E isso implicava em uma complexidade computacional para determinar os vizinhos de um pixel nas coroas que dividiam as duas camadas. Além de não se mostrar agradável visualmente. A partir daí surgiu a idéia da distribuição de pixels que utilizamos nos espaços Ω_H e Ω_E apresentados no Capítulo 3.

Agora, devido a natureza esférica dos objetos em questão é natural pensarmos em uma abordagem que utilize a métrica sobre a esfera e projeções de um hemisfério da esfera no plano a fim de conseguirmos uma distribuição de pixels que satisfaça as condições propostas.

Vejamos, primeiramente, duas projeções de um disco $D_r = \left\{(x, y) \mid x^2 + y^2 \leqslant r^2\right\}$ em um hemisfério de uma esfera.

PROJEÇÃO T_1

A primeira projeção, que chamaremos de T_1, foi inspirada na forma como é construída a isometria entre o modelo do Disco de Poincaré e o modelo do Disco de Klein. [9]. A projeção $T_1 : D_r \to S_r$ leva os pontos do disco D_r em um dos hemisférios da esfera de mesmo raio $S_r = \left\{(x, y, z) \mid x^2 + y^2 + z^2 = r^2, z > 0\right\}$ da seguinte forma: a reta que passa pelo ponto $(0, 0, -r)$ e pelo ponto $(x, y) \in D_r$ intersecta S_r no ponto $T_1(x, y)$. Seja $P = (x, y)$ o ponto de D_r que é levado em $T_1(x, y) = (x', y', z')$.

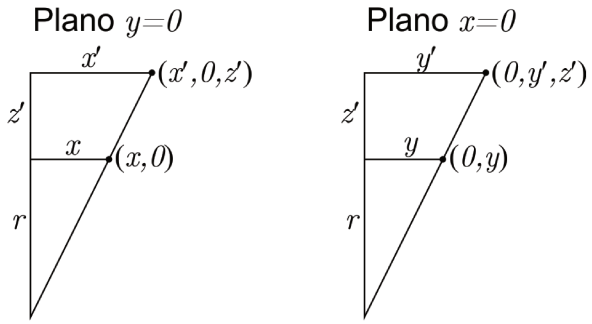


Figura A.10: Projeções nos planos $x = 0$ e $y = 0$.

Consideremos as projeções de $T_1(x, y)$ nos planos $x = 0$ e $y = 0$ (Figura A.10). Sabemos que x e x' tem o mesmo sinal, assim como y e y' nessas projeções. Assim, por semelhança de triângulos temos: $\frac{|x|}{|x'|} = \frac{r}{r + z'} \Longrightarrow x' = x\frac{r + z'}{r}$ e $\frac{|y|}{|y'|} =$

$\frac{r}{r+z'} \Longrightarrow y' = y\frac{r+z'}{r}$. Como $T_1(x,y) \in S_r$ então $(x')^2 + (y')^2 + (z')^2 = r^2$ donde $x^2\left(\frac{r+z'}{r}\right)^2 + y^2\left(\frac{r+z'}{r}\right)^2 + (z')^2 = r^2$ e daí, $(z')^2\left(x^2+y^2+r^2\right) + z'\left(2x^2r + 2y^2r\right) + x^2r^2 + y^2r^2 - r^4 = 0$. Resolvendo a equação do segundo grau em z' temos $\Delta = 4r^6$ e daí $z' = \frac{-2r\left(x^2+y^2\right) \pm 2r^3}{2\left(x^2+y^2+r^2\right)} = \frac{-r\left(x^2+y^2\right)+r^3}{x^2+y^2+r^2}$. O sinal deve ser positivo pois, caso contrário $z' = -r$ mas, por definição, $z' > 0$. Substituindo o valor de z' nas equações de x' e y' temos as coordenadas de T_1. Assim:

$$\begin{array}{rccc} T_1: & D_r & \longrightarrow & S_r \\ & (x,y) & \longmapsto & \left(\frac{2xr^2}{x^2+y^2+r^2}, \frac{2yr^2}{x^2+y^2+r^2}, \frac{r\left(r^2-x^2-y^2\right)}{x^2+y^2+r^2}\right). \end{array}$$

Observamos, na Figura A.11, a projeção T_1.

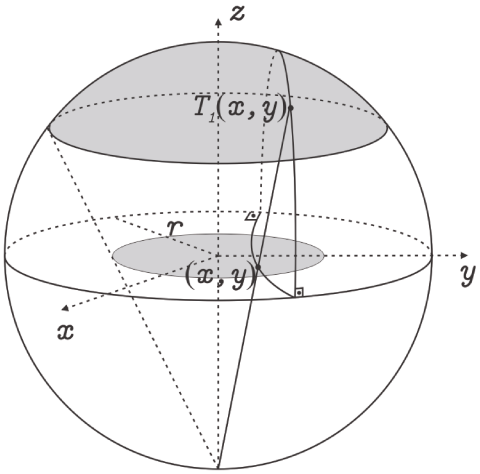


Figura A.11: Projeção T_1.

PROJEÇÃO T_2

A segunda projeção, que chamaremos de T_2, leva os pontos do disco D_r em pontos de S_r de maneira similar à projeção cilíndrica, ou seja, uma reta que intersecta perpendicularmente o disco D_r no ponto (x,y) intersecta S_r no ponto $T_2(x,y)$. Dessa forma, se $P = (x,y) \in D_r$ e $T_2(x,y) = (x',y',z')$ temos que $x' = x$ e $y' = y$. Daí, como $T_2(x,y) \in S_r$ então $(x')^2 + (y')^2 + (z')^2 = r^2$ donde $x^2 + y^2 + (z')^2 = r^2$. Assim, temos:

$$\begin{array}{rccc} T_2: & D_r & \longrightarrow & S_r \\ & (x,y) & \longmapsto & \left(x, y, \sqrt{r^2-x^2-y^2}\right) \end{array}$$

Observamos, na Figura A.12, que a projeção T_2 mantém o aspecto visual das formas geométricas em D_r se tomarmos como ponto de referência o eixo z.

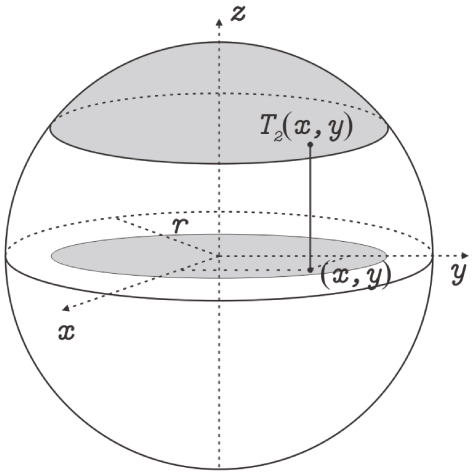


Figura A.12: Projeção T_2.

Considerando a divisão de pixels dada pela Figura A.2 gostaríamos de determinar, de forma eficiente, os raios dessas circunferências. Já vimos que essa divisão apresenta o problema no centro da imagem. Entretanto, para testar o comportamento dessa abordagem essa distribuição é a mais simples.

Seria interessante se, de alguma forma, a área dos pixels do espaço suporte tivesse relação com área da projeção dos pixels na esfera. Comecemos supondo que a área das projeções dos pixels na esfera sejam todas iguais. Nesse caso, como em cada coroa há a mesma quantidade de divisões, a esfera deve ser dividida em zonas esféricas de mesma altura h já que a área de uma zona esférica (Figura A.13) é dada por $2\pi rh$ onde r é o raio da esfera. Logo, supondo que temos c coroas então a altura h_i de cada uma das coroas deve ser $h_i = \dfrac{r}{c} = \dfrac{P}{2c}$. Como $h_1 + h_2 + ... + h_c = \dfrac{P}{2}$ então o raio R'_i das circunferências que são interseções da esfera com os planos $z = \dfrac{P}{2} - h_1 - h_2 - ... - h_i$ é $\left(R'_i\right)^2 = \left(\dfrac{P}{2}\right)^2 - \left(\dfrac{P}{2} - \dfrac{iP}{2c}\right)^2 \Longrightarrow R'_i = \dfrac{P}{2c}\sqrt{2ci - i^2}$. Em seguida, esses R'_is devem ser projetados no disco (plano $z = 0$). Utilizaremos, para isso, as duas projeções que trabalhamos. Podemos pensar que R'_i é a coordenada x' de um ponto $(x', 0, z')$ da esfera. Assim, R'_i projetado no disco por T_1^{-1} é $R_1(i) = \dfrac{P}{2}\left(\dfrac{i}{\sqrt{2ci - i^2}}\right)$ e projetado por T_2^{-1} é $R_2(i) = \dfrac{P}{2c}\sqrt{2ci - i^2}$.

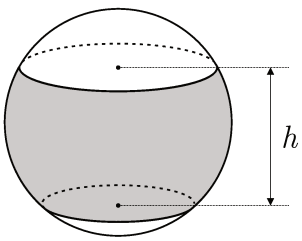


Figura A.13: Zona esférica.

Para $P = 50$, por exemplo, teremos $round\,(50\pi) = 157$ pixels (divisões) na última camada. Como queremos que o espaço suporte tenha $0.628\,(50)^2 = 1570$ pixels então devemos construir $round\left(\dfrac{1570}{157}\right) = 10$ coroas, ou seja, 9 circunferências dentro de C com 157 divisões em cada coroa. Observamos na tabela abaixo como ficariam os raios aproximados das 9 circunferências segundo as duas projeções que trabalhamos. Veja que, para a primeira projeção a diferença entre a 9^a circunferência e C (raio 25) é 2.39 e, com a segunda projeção (Figura A.14) a diferença é de 0.13. Ambos os valores estão distantes da unidade proposta para a última coroa

i	1	2	3	4	5	6	7	8	9	10
$R_1\,(i)$	5.73	8.33	10.5	12.5	14.43	16.36	18.34	20.41	22.61	25
$R_2\,(i)$	10.89	15	17.85	20	21.65	22.91	23.84	24.49	24.87	25

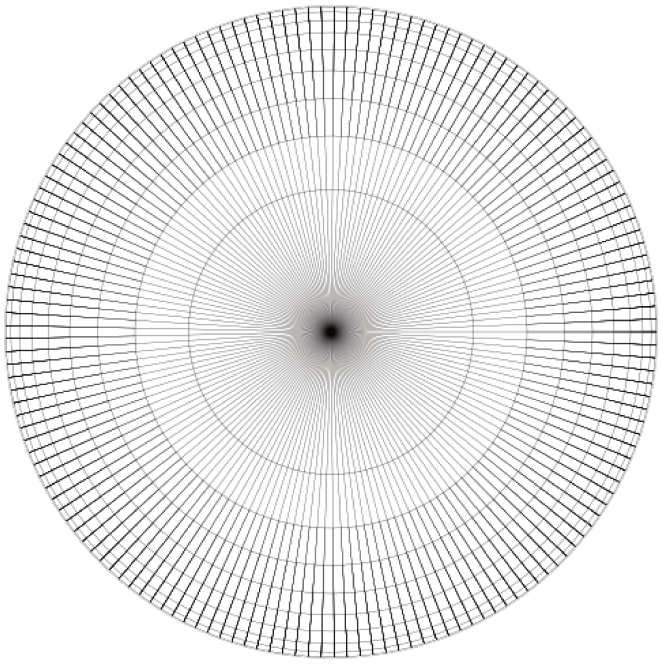


Figura A.14: Espaço suporte para $P = 50$ utilizando T_2.

Essas abordagens ficaram distantes da proposta inicial. Observamos, na Figura 14, que a última coroa fica praticamente imperceptível. Por isso, surgiu a ideia de tentar o caminho inverso: determinar, primeiramente, a altura da projeção da última coroa com base na hipótese da diferença entre os raios da última coroa ser 1. Devemos então projetar, por exemplo, o ponto $\left(\dfrac{P}{2} - 1, 0\right)$ na esfera segundo as duas projeções para descobrir a medida de h que será a coordenada z da projeção. Então $T_1\left(\dfrac{P}{2} - 1, 0\right) = \left(\dfrac{(P-2)\,P^2}{(P-2)^2 + P^2}, 0, \dfrac{2P\,(P-1)}{(P-2)^2 + P^2}\right)$ e $T_2\left(\dfrac{P}{2} - 1, 0\right) = \left(\dfrac{P}{2} - 1, 0, \sqrt{P-1}\right)$.

No caso de $P = 50$ temos que $h = 1,02$ para a projeção T_1 e $h = 7$ para T_2. Para $h = 1,02$ poderíamos construir 23 coroas de áreas iguais e deixar a coroa central com

área maior. Para $h = 7$ poderíamos construir 3 coroas de áreas iguais e uma com área um pouco maior. Em ambos os casos o número de coroas destoa muito das 9 camadas que propomos para haver $62,8\%$ dos pixels iniciais no espaço novo. Além disso, observamos que, por essa abordagem não é possível escolher a quantidade de camada e com isso, a quantidade de pixels.

PROJEÇÃO T_3

Como vimos, as duas projeções anteriores não fornecem bons raios para o nosso modelo. Consideraremos agora uma projeção, análoga à T_1, porém, o ponto que define a projeção não será mais o pólo sul da esfera $(0, 0, -R)$ e sim um ponto $(0, 0, N)$ escolhido de forma a satisfazer as hipóteses impostas pelo nosso problema.

Consideremos a circunferência dada pela interseção da esfera de raio R com o plano $y = 0$. Dado l queremos encontrar o ponto N onde a reta que passa pelo ponto $\left(\sqrt{R^2 - l^2}, l\right)$ e $(R - 1, 0)$ intersecte o eixo z (Figura 15). A reta que passa por esses pontos é dada por $z = \dfrac{l}{\sqrt{R^2 - l^2} - (R - 1)} (x - (R - 1))$. Dessa forma, $N = \dfrac{-l\,(R - 1)}{\sqrt{R^2 - l^2} - (R - 1)}$.

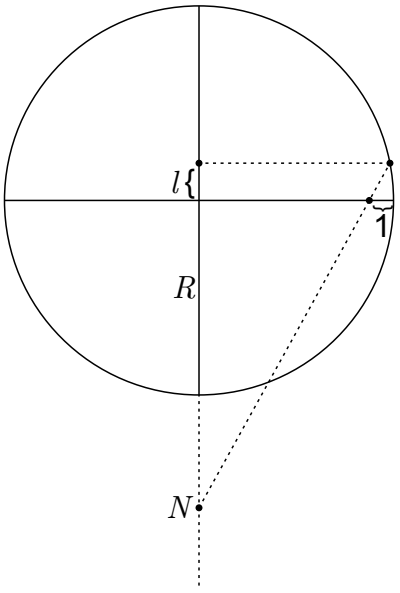


Figura A.15: Projeção T_3.

Consideremos agora a projeção $T_3 : D_R \longrightarrow, S_R$ análoga a T_1, que leva os pontos de D_R à S_R através da reta que passa por $(0, 0, N)$ e $(x, y, 0)$. Assim, a projeção é dada

por

$$T_3(x,y) = \begin{pmatrix} -x\left(\dfrac{(x^2+y^2)-\sqrt{(R^2-N^2)(x^2+y^2)+N^2R^2}}{x^2+y^2+N^2}\right), \\ -y\left(\dfrac{(x^2+y^2)-\sqrt{(R^2-N^2)(x^2+y^2)+N^2R^2}}{x^2+y^2+N^2}\right), \\ \dfrac{N(x^2+y^2)-N\sqrt{(R^2-N^2)(x^2+y^2)+N^2R^2}}{x^2+y^2+N^2} \end{pmatrix}.$$

Assim, utilizando T_3^{-1}, dado um ponto $0 \leqslant z' \leqslant R$, a coordenada x tal que $T_3^{-1}(0,0,z') = (x,0)$ é dada por $x = \dfrac{N\sqrt{R^2-(z')^2}}{N-z'}$.

Vejamos na Figura A.16 que para o caso de $P = 50$ a distribuição dos raios fica mais harmônica do que nas projeções T_1 e T_2. Além disso, observamos que os pixels do bordo se assemelham aos pixels de Ω_T embora os pixels centrais sejam finos demais.

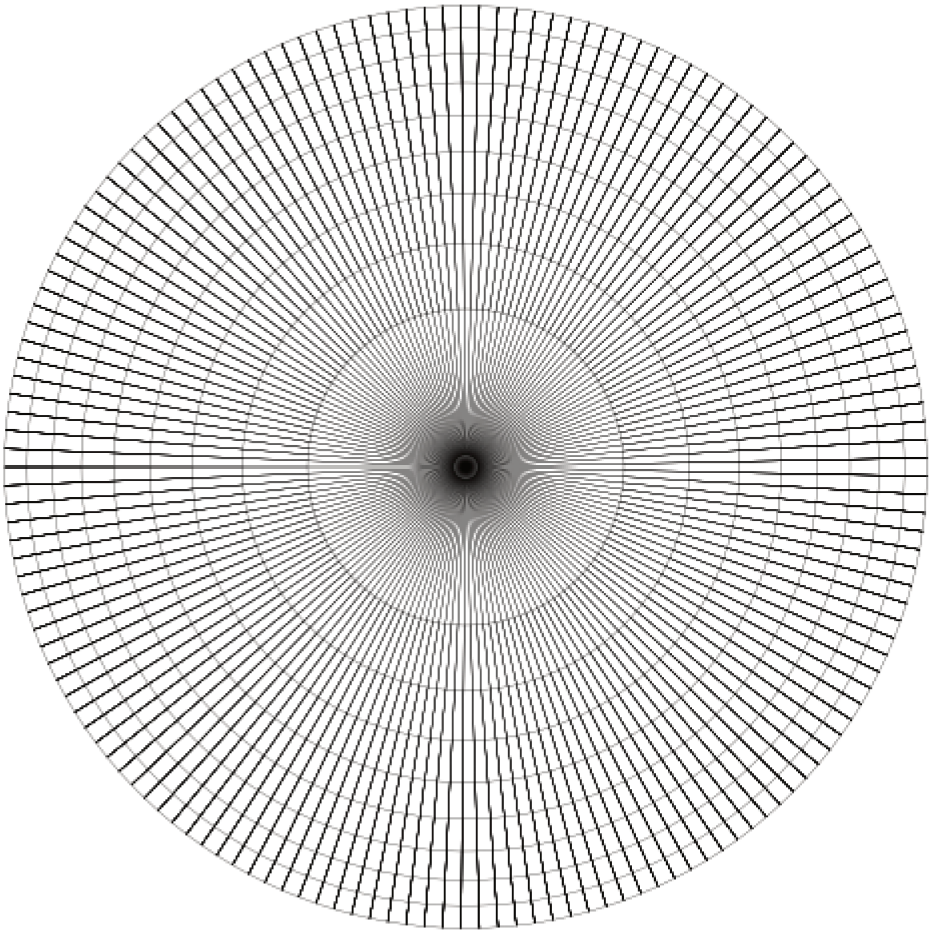


Figura A.16: Espaço Suporte utilizando T_3.

A partir desses testes chegamos às abordagens apresentadas no Capítulo 3.

Para fazer a quantização, a proposta inicial também foi sobrepor os espaços suporte. Entretanto, para determinar se um pixel de Ω_T contribuia para o tom de cinza de um pixel de Ω_H (ou Ω_E) testávamos quatro desigualdades para o pixel de Ω_T com base nas retas que delimitava o pixel em Ω_H (ver Figura A.17).

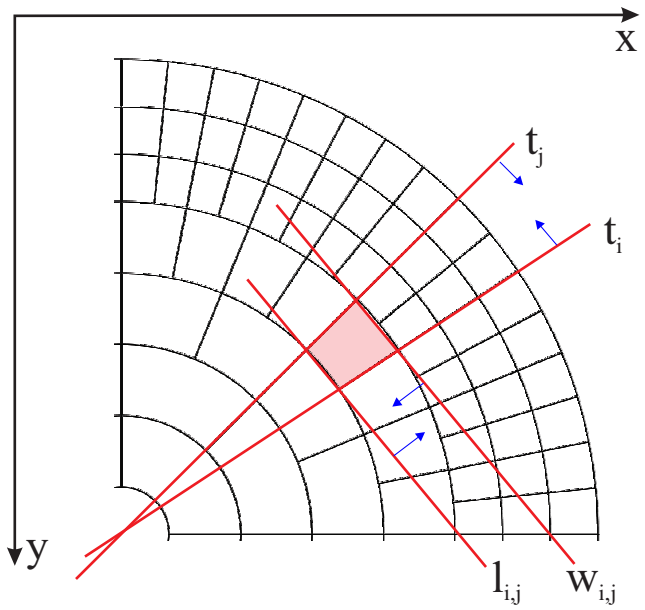


Figura A.17: Proposta inicial de quantização.

Para gerar a imagem após o processo proposto estávamos utilizando uma ferramenta do *Python* chamada Wedge. Esta ferramenta constrói setores circulares em um disco com o tom de cinza que desejarmos. Entretanto, as imagens geradas não eram do mesmo tamanho das imagens originais. E isso, juntamente com os vários testes necessários para a quantização, implicava em um tempo muito grande para gerar as imagens. Para as operações morfológicas houveram imagens que, com esta abordagem, demoravam cerca de 56 minutos para serem geradas enquanto que, no algoritmo atual (apresentado no Capítulo 3), é gerada em cerca de 4 minutos.

APÊNDICE B – Algoritmos em *Python*

Conversão de imagens de Ω_T para Ω_H

```
import matplotlib.pyplot as plt
import skimage.color
import math as m
import scipy.optimize
import numpy as np
import itertools as it
import skimage
from PIL import Image
from datetime import datetime
startTime = datetime.now()
img = plt.imread('Bola.png')
M = skimage.color.rgb2gray(img)
P = len(M)
"'Parâmetros
n é a quantidade de divisões da primeira coroa
k é a quantidade de camadas (locais onde ocorre a divisão por 2)(exceto Q0,0)
c1 é a quantidade de vezes que a segunda coroa é repetida (as seguintes dobra a quantidade de repetição)"'
"Cálculo dos parâmetros"
A2 = P*m.pi
A3 = []
while A2 > 10:
    A3.append(A2)
    A2 = A2/2
A4 = []
k = len(A3)
```

```
n = int(round(A3[-1]))
c1aux=0
for i in range(1,k):
    c1aux += 2**i*n*i
if c1aux == 0:
    c1 = 1
else:
    c1 = int(round((0.628*P*P-1-n)/c1aux))
pixelstotais = (1+n+c1*c1aux)/P**2
kaux = []
caux = [1]
naux = [n]
for i in range(1,k):
    caux.append(i*c1)
    naux.append(2**i*n)
for i in range(k):
    for j in range(caux[i]):
        kaux.append(naux[i])
c = len(kaux)+1
vetor = []
for i in kaux:
    vetor.append(int(i/4))
Q = np.zeros((c,kaux[-1]))
Mcont = np.zeros((c,kaux[-1]))
"raios hiperbolicos"
def f1(y):
    return y**(2*c-1)+(P-1)*y**(c-1)-(P-1)*y**c-1
y = scipy.optimize.brentq(f1,1.00001,1.3)
x = m.log(y)
curvatura = -x*x
```

```
raio = (P/2-1)*((m.exp((c-1)*x)+1)/(m.exp((c-1)*x)-1))
def rh(x):
    return raio*((m.exp(m.sqrt(-curvatura)*x)-1)/(m.exp(m.sqrt(-curvatura)*x)+1))
"Transformando coordenadas cartesianas em polares, guardando as informações apenas do disco e por quadrante"
Q0 = []
I = np.zeros((P,P,2),int)
O=(P-1)/2
def r(x,y):
    return m.sqrt((x-O)**2+(y-O)**2)
def theta(x,y):
    if x>O:
        a = m.atan((O-y)/(x-O))
        if y<O:
            a+=0
        if y>O:
            a+=2*m.pi
    if x < O:
        a = m.atan((O-y)/(x-O))
        if y<O:
            a+=m.pi
        if y>O:
            a+=m.pi
    if x==O:
        if y>O:
            a=m.pi/2
        if y<O:
            a=3*m.pi/2
        if y==O:
            if x>O:
```

```
                    a=0
                if x<O:
                    a=m.pi
    return a
def graus(x):
    return x*180/m.pi
r1 = rh(1)
rhaux=[]
for i in range(c):
    rhaux.append(rh(i+1))
for j in range(0,P):
    for i in range(0,P):
        tij = theta(i,j)
        rij = r(i,j)
        if rij<=r1:
            Q0.append(M[j][i])
            I[j][i][0] = 0
            I[j][i][1] = 0
        if rij>=P/2:
            I[j][i][0] = 0
            I[j][i][1] = 1
        for l, l1 in it.zip_ longest(rhaux [:c-1],rhaux [1:]):
            if l<rij<l1:
                i1 = rhaux.index(l)+1
                j1 = tij//(2*m.pi/kaux[i1-1])
                Q[i1][j1]+=M[j][i]
                Mcont[i1][j1]+=1
                I[j][i][0] = i1
                I[j][i][1] = j1
Q[0][0]=sum(Q0)/len(Q0)
```

```
Q[0][1] = 1
"' Se não receber nenhum pixel arredonda pro mais próximo"'
def raioaprox(i):
    return (rh(i+1)+rh(i+2))/2
def angaprox(i,j):
    return -(2*m.pi/kaux[i])*(j+1/2)
pixelvazio = 0
for i in range(c-1):
    for j in range(kaux[i]):
        if Mcont[i+1][j] == 0:
            xaprox = round((P-1)/2+raioaprox(i)*m.cos(angaprox(i,j)))
            yaprox = round((P-1)/2+raioaprox(i)*m.sin(angaprox(i,j)))
            Q[i+1][j] = M[yaprox][xaprox]
            pixelvazio +=1
        else:
            Q[i+1][j] /= Mcont[i+1][j]
"'Construindo Figura"'
F=np.zeros((P,P))
for i in range(P):
    for j in range(P):
        F[j][i] = Q[I[j][i][0]][I[j][i][1]]
result = Image.fromarray((F * 255).astype(np.uint8))
result.save('BolaPSNR-H.png')
print(datetime.now() - startTime)
```

Conversão de imagens de Ω_T para Ω_E

```
import matplotlib.pyplot as plt
import skimage.color
import math as m
import scipy.optimize
import numpy as np
```

```
import itertools as it
import skimage
from PIL import Image
from datetime import datetime
startTime = datetime.now()
img = plt.imread('Bola.png')
M = skimage.color.rgb2gray(img)
P = len(M)
"'Parâmetros
n é a quantidade de divisões da primeira coroa
k é a quantidade de camadas (locais onde ocorre a divisão por 2) (exceto Q0,0)
c1 é a quantidade de vezes que a segunda coroa é repetida (as seguintes dobra a quantidade de repetição)"'
"Cálculo dos parâmetros"
A2 = P*m.pi
A3 = []
while A2 > 10:
    A3.append(A2)
    A2 = A2/2
A4 = []
k = len(A3)
n = int(round(A3[-1]))
c1aux=0
for i in range(1,k):
    c1aux += 2**i*n*i
if c1aux == 0:
    c1 = 1
else:
    c1 = int(round((0.628*P*P-1-n)/c1aux))
pixelstotais = (1+n+c1*c1aux)/P**2
```

```
kaux = []
caux = [1]
naux = [n]
for i in range(1,k):
    caux.append(i*c1)
    naux.append(2**i*n)
for i in range(k):
    for j in range(caux[i]):
        kaux.append(naux[i])
c = len(kaux)+1
vetor = []
for i in kaux:
    vetor.append(int(i/4))
Q = np.zeros((c,kaux[-1]))
Mcont = np.zeros((c,kaux[-1]))
"raios elípticos"
R = 10/(2*m.pi)
d = 25- R
a = R*R/d
h0 = (R-a)/(1+sum(kaux))
l = (h0)*kaux[c-2]
ma = ((P-2)*m.sqrt(R**2-a**2)*(d-l-a)-P*(d-a)*m.sqrt(R**2-(a
+l)**2))/(P*m.sqrt(R*R-(a+l)**2)-(P-2)*m.sqrt(R**2-a**2))
lamb = (P/2)*(d+ma-a)/m.sqrt(R**2-a**2)-ma
haux = [h0]
for i in range(1,c):
    haux.append(h0*kaux[i-1])
zaux = []
h0aux = 0
for i in range(c):
```

```
    h0aux += haux[i]
    zaux.append(h0aux)
def z(x):
    return (lamb+ma)*m.sqrt(2*R*x-x**2)/(ma+d-R+x)
def rh(x):
    return z(zaux[x-1])
"Transformando coordenadas cartesianas em polares, guardando as informações apenas do disco e por quadrante"
Q0 = []
I = np.zeros((P,P,2),int)
O=(P-1)/2
def r(x,y):
    return m.sqrt((x-O)**2+(y-O)**2)
def theta(x,y):
    if x>O:
        a = m.atan((O-y)/(x-O))
        if y<O:
            a+=0
        if y>O:
            a+=2*m.pi
    if x < O:
        a = m.atan((O-y)/(x-O))
        if y<O:
            a+=m.pi
        if y>O:
            a+=m.pi
    if x==O:
        if y>O:
            a=m.pi/2
        if y<O:
```

```
            a=3*m.pi/2
        if y==O:
            if x>O:
                a=0
            if x<O:
                a=m.pi
    return a
def graus(x):
    return x*180/m.pi
r1 = rh(1)
rhaux=[]
for i in range(c):
    rhaux.append(rh(i+1))
for j in range(0,P):
    for i in range(0,P):
        tij = theta(i,j)
        rij = r(i,j)
        if rij<=r1:
            Q0.append(M[j][i])
            I[j][i][0] = 0
            I[j][i][1] = 0
        if rij>=P/2:
            I[j][i][0] = 0
            I[j][i][1] = 1
        for l, l1 in it.zip_ longest(rhaux [:c-1],rhaux [1:]):
            if l<rij<l1:
                i1 = rhaux.index(l)+1
                j1 = tij//(2*m.pi/kaux[i1-1])
                Q[i1][j1]+=M[j][i]
                Mcont[i1][j1]+=1
```

```
                    I[j][i][0] = i1
                    I[j][i][1] = j1
Q[0][0]=sum(Q0)/len(Q0)
Q[0][1] = 1
"' Se não receber nenhum pixel arredonda pro mais próximo"'
def raioaprox(i):
    return (rh(i+1)+rh(i+2))/2
def angaprox(i,j):
    return -(2*m.pi/kaux[i])*(j+1/2)
pixelvazio = 0
for i in range(c-1):
    for j in range(kaux[i]):
        if Mcont[i+1][j] == 0:
            xaprox = round((P-1)/2+raioaprox(i)*m.cos(angaprox(i,j)))
            yaprox = round((P-1)/2+raioaprox(i)*m.sin(angaprox(i,j)))
            Q[i+1][j] = M[yaprox][xaprox]
            pixelvazio +=1
        else:
            Q[i+1][j] /= Mcont[i+1][j]
"'Construindo Figura"'
F=np.zeros((P,P))
for i in range(P):
    for j in range(P):
        F[j][i] = Q[I[j][i][0]][I[j][i][1]]
result = Image.fromarray((F * 255).astype(np.uint8))
result.save('BolaPSNR-H.png')
print(datetime.now() - startTime)
```

Filtros Morfológicos

```
import matplotlib.pyplot as plt
import skimage.color
```

```
import math as m
import scipy.optimize
import numpy as np
import itertools as it
import skimage
from PIL import Image
from datetime import datetime
startTime = datetime.now()
img = plt.imread('Bola.png')
M = skimage.color.rgb2gray(img)
P = len(M)
ordenamento = int(input('Insira o ordenamento: '))
"'Trocando escala de cinza para escala real"'
for i in range(P):
    for j in range(P):
        if M[i][j] == 0:
            M[i][j] = -1000
        if M[i][j] ==1:
            M[i][j] = 1000
        M[i][j] = 1*(M[i][j]-1/2)/(M[i][j]-(M[i][j])**2)
"' Desfazendo Escala "'
def rees(x):
    if abs(x) < 10**(-10):
        return 1/2
    else:
        return (x-1+m.sqrt(1+x**2))/(2*x)
"Cálculo dos parâmetros"
A2 = P*m.pi
A3 = []
while A2 > 10:
```

```
        A3.append(A2)
        A2 = A2/2
A4 = []
k = len(A3)
n = int(round(A3[-1]))
c1aux=0
for i in range(1,k):
        c1aux += 2**i*n*i
if c1aux == 0:
        c1 = 1
else:
        c1 = int(round((0.628*P*P-1-n)/c1aux))
pixelstotais = (1+n+c1*c1aux)/P**2
kaux = []
caux = [1]
naux = [n]
for i in range(1,k):
        caux.append(i*c1)
        naux.append(2**i*n)
for i in range(k):
        for j in range(caux[i]):
                kaux.append(naux[i])
c = len(kaux)+1
vetor = []
for i in kaux:
        vetor.append(int(i/4))
Q = np.zeros((c,kaux[-1]))
Mcont = np.zeros((c,kaux[-1]))
"raios hiperbolicos"
def f1(y):
```

```
    return y**(2*c-1)+(P-1)*y**(c-1)-(P-1)*y**c-1
y = scipy.optimize.brentq(f1,1.00001,1.3)
x = m.log(y)
curvatura = -x*x
raio = (P/2-1)*((m.exp((c-1)*x)+1)/(m.exp((c-1)*x)-1))
def rh(x):
    return raio*((m.exp(m.sqrt(-curvatura)*x)-1)/(m.exp(m.sqrt(-curvatura)*x)+1))
"Transformando coordenadas cartesianas em polares, guardando as informações apenas do disco e por quadrante"
Q0 = []
I = np.zeros((P,P,2),int)
O=(P-1)/2
def r(x,y):
    return m.sqrt((x-O)**2+(y-O)**2)
def theta(x,y):
    if x>O:
        a = m.atan((O-y)/(x-O))
        if y<O:
            a+=0
        if y>O:
            a+=2*m.pi
    if x < O:
        a = m.atan((O-y)/(x-O))
        if y<O:
            a+=m.pi
        if y>O:
            a+=m.pi
    if x==O:
        if y>O:
            a=m.pi/2
```

```
        if y<O:
            a=3*m.pi/2
        if y==O:
            if x>O:
                a=0
            if x<O:
                a=m.pi
    return a
def graus(x):
    return x*180/m.pi
r1 = rh(1)
rhaux=[]
for i in range(c):
    rhaux.append(rh(i+1))
for j in range(0,P):
    for i in range(0,P):
        tij = theta(i,j)
        rij = r(i,j)
        if rij<=r1:
            Q0.append(M[j][i])
            I[j][i][0] = 0
            I[j][i][1] = 0
        if rij>=P/2:
            I[j][i][0] = 0
            I[j][i][1] = 1
        for l, l1 in it.zip_ longest(rhaux [:c-1],rhaux [1:]):
            if l<rij<l1:
                i1 = rhaux.index(l)+1
                j1 = tij//(2*m.pi/kaux[i1-1])
                Q[i1][j1]+=M[j][i]
```

```
                    Mcont[i1][j1]+=1
                    I[j][i][0] = i1
                    I[j][i][1] = j1
Q[0][0]=sum(Q0)/len(Q0)
Q[0][1] = 1
"' Se não receber nenhum pixel arredonda pro mais próximo"'
def raioaprox(i):
    return (rh(i+1)+rh(i+2))/2
def angaprox(i,j):
    return -(2*m.pi/kaux[i])*(j+1/2)
pixelvazio = 0
for i in range(c-1):
    for j in range(kaux[i]):
        if Mcont[i+1][j] == 0:
            xaprox = round((P-1)/2+raioaprox(i)*m.cos(angaprox(i,j)))
            yaprox = round((P-1)/2+raioaprox(i)*m.sin(angaprox(i,j)))
            Q[i+1][j] = M[yaprox][xaprox]
            pixelvazio +=1
        else:
            Q[i+1][j] /= Mcont[i+1][j]
Q1 = []
Q1.append([Q[0][0]])
for i in range(c-1):
    Q1linha = []
    for j in range(kaux[i]):
        Q1linha.append(Q[i+1][j])
    Q1.append(Q1linha)
"' SE e W "'
def SE(i,j):
    A1 = []
```

```
A2 = []
a1=2+c1
for l in range(2,k):
    A2.append(a1)
    A1.append(a1-1)
    a1+=l*c1
if i==0:
    Q00aux = [[0,0]]
    for l in range(kaux[0]):
        Q00aux.append([1,l])
    return Q00aux
elif i==1:
    return [[0,0],[1,(j-1)%kaux[i-1]],[1,j],[1,(j+1)%kaux[i-1]],
    [i+1,(2*j-1)%kaux[i]],[i+1,2*j],[i+1,2*j+1],[i+1,(2*j+2)%kaux[i]]]
elif i==2:
    if c1 ==1:
        return [[i-1,((j+1)//2-1)%kaux[i-2]],[i-1,((j+1)//2)%kaux[i-2]],
        [i,(j-1)%kaux[i-1]],[i,j],[i,(j+1)%kaux[i-1]],[i+1,(2*j-1)%kaux[i]],
        [i+1,2*j],[i+1,2*j+1],[i+1,(2*j+2)%kaux[i]]]
    else:
        return [[i-1,((j+1)//2-1)%kaux[i-2]],[i-1,((j+1)//2)%kaux[i-2]],
        [i,(j-1)%kaux[i-1]],[i,j],[i,(j+1)%kaux[i-1]],[i+1,(j-1)%kaux[i]],
        [i+1,j],[i+1,(j+1)%kaux[i]]]
elif i==c-1:
    return [[i-1,(j-1)%kaux[i-2]],[i-1,j],[i-1,(j+1)%kaux[i-2]],
    [i,(j-1)%kaux[i-1]],[i,j],[i,(j+1)%kaux[i-1]]]
elif i in A2:
    return [[i-1,((j+1)//2-1)%kaux[i-2]],[i-1,((j+1)//2)%kaux[i-2]],
    [i,(j-1)%kaux[i-1]],[i,j],[i,(j+1)%kaux[i-1]],[i+1,(j-1)%kaux[i]],
    [i+1,j],[i+1,(j+1)%kaux[i]]]
```

```
    elif i in A1:
        return [[i-1,(j-1)%kaux[i-2]],[i-1,j],[i-1,(j+1)%kaux[i-2]],
        [i,(j-1)%kaux[i-1]],[i,j],[i,(j+1)%kaux[i-1]],[i+1,(2*j-1)%kaux[i]],
        [i+1,2*j],[i+1,2*j+1],[i+1,(2*j+2)%kaux[i]]]
    else:
        return [[i-1,(j-1)%kaux[i-2]],[i-1,j],[i-1,(j+1)%kaux[i-2]],
        [i,(j-1)%kaux[i-1]],[i,j],[i,(j+1)%kaux[i-1]],[i+1,(j-1)%kaux[i]],
        [i+1,j],[i+1,(j+1)%kaux[i]]]
def W(Q,i,j):
    l5 = []
    for u in SE(i,j):
        l5.extend(SE(u[0],u[1]))
    l6=l5[:]
    for u in l6:
        while l5.count(u)>1:
            l5.remove(u)
    l1 = []
    for u in l5:
        l1.append(Q[u[0]][u[1]])
    return l1
        "'MEC"'
def Circ2pontos(p1,p2):
    a = (p2[0]**2+p2[1]**2-p1[0]**2-p1[1]**2)/(2*(p2[0]-p1[0]))
    r = m.sqrt(((p1[1]**2-p2[1]**2-(p1[0]-p2[0])**2)/(2*(p2[0]-p1[0])))**2+p1[1]**2)
    return [a,r]
def media(lista):
    return sum(lista)/len(lista)
def centromedio(lista):
    lista1coord = []
    for i in lista:
```

```
        lista1coord.append(i[0])
    centro = media(lista1coord)
    return centro
def d2p(p1,p2):
    return m.sqrt((p1[0]-p2[0])**2+(p1[1]-p2[1])**2)
def MEC(listateste):
    cm = centromedio(listateste)
    def MaxAoCentro(lista):
        lista1 =[]
        for i in lista:
            lista1.append(d2p([cm,0],i))
        x = lista1.index(max(lista1))
        y = lista1.count(max(lista1))
        return [max(lista1),x,y]
    i1 = MaxAoCentro(listateste)[1]
    x1 = listateste[i1]
    lix = listateste[:]
    lix.remove(x1)
    l1=[]
    raios=[]
    centros=[]
    for i in listateste:
        l1.append(d2p((x1[0],0),i))
    if max(l1)<=x1[1]:
        raios.append(x1[1])
        centros.append(x1[0])
    else:
        lesquerda=[]
        ldireita=[]
        lcentros=[]
```

```
            lix2=lix[:]
            for i in lix2:
                if i[0]==x1[0]:
                    lix.remove(i)
                else:
                    lcentros.append(Circ2pontos(i,x1)[0])
            for i in range(len(lix)):
                if lix[i][0]<x1[0]:
                    lesquerda.append(lcentros[i])
                if lix[i][0]>x1[0]:
                    ldireita.append(lcentros[i])
            if len(lesquerda)>0:
                iE=lcentros.index(min(lesquerda))
                p2=lix[iE]
                [c2,r2]=Circ2pontos(p2,x1)
                raios.append(r2)
                centros.append(c2)
            if len(ldireita)>0:
                iD=lcentros.index(max(ldireita))
                p3=lix[iD]
                [c3,r3]=Circ2pontos(p3,x1)
                raios.append(r3)
                centros.append(c3)
        r = min(raios)
        c = centros[raios.index(min(raios))]
        return [c,r]
def Comp(ponto):
        return [-ponto[0],1/ponto[1]]
                "' Ordenamentos"'
if ordenamento == 1:
```

```
    def ordmin(lista):
        l1 = []
        l2 = []
        for i in lista:
            l1.append(i[0])
            l2.append(i[1])
        return [min(l1),min(l2)]
    def ordmax(lista):
        l1=[]
        l2=[]
        for i in lista:
            l1.append(i[0])
            l2.append(i[1])
        return [max(l1),max(l2)]
if ordenamento == 2:
    def ordmin(lista):
        l1 = []
        l2 = []
        for i in lista:
            l1.append(i[0])
            l2.append(i[1])
        if all([i>0 for i in l1]) and all([j>1 for j in l2]):
            ptmin = [min(l1),min(l2)]
        elif all([i<0 for i in l1]) and all([j>1 for j in l2]):
            ptmin = [max(l1),min(l2)]
        elif all([i<0 for i in l1]) and all([j<1 for j in l2]):
            ptmin = [max(l1),max(l2)]
        elif all([i>0 for i in l1]) and all([j<1 for j in l2]):
            ptmin = [min(l1),max(l2)]
        else:
```

```
            ptmin = [0,1]
        return ptmin
    def ordmax(lista):
        l1 = []
        l2 = []
        for i in lista:
            l1.append(i[0])
            l2.append(i[1])
        if all([i>=0 for i in l1]) and all([j>=1 for j in l2]):
            ptm = [max(l1),max(l2)]
        elif all([i<=0 for i in l1]) and all([j>=1 for j in l2]):
            ptm = [min(l1),max(l2)]
        elif all([i<=0 for i in l1]) and all([j<=1 for j in l2]):
            ptm = [min(l1),min(l2)]
        elif all([i>=0 for i in l1]) and all([j<=1 for j in l2]):
            ptm = [max(l1),min(l2)]
        else:
            ptm =0
        return ptm
if ordenamento == 3:
    def raioord3(lista):
        ro = m.acosh((lista[0]**2+lista[1]**2+1)/(2*lista[1]))
        return ro
    def angord3(lista):
        return m.atan((lista[0]**2+lista[1]**2-1)/(2*lista[0]))
    def tal(lista):
        a = angord3(lista)
        if a == 3*m.pi/2:
            return 0
        else:
```

```
        return 2*m.cos(a)/(1-m.sin(a))
    def ordmin(lista):
        eta = []
        for j in lista:
            eta.append(raioord3(j))
        etamin = min(eta)
        ind = eta.index(etamin)
        ptminord3 = lista[ind]
        if eta.count(etamin)>1:
            lieta = []
            for i in range(len(eta)):
                if eta[i] == etamin:
                    lieta.append(i)
            etaarg=[]
            for j in lieta:
                lietaaux = tal(lista[j])
                etaarg.append(lietaaux)
            ind = etaarg.index(min(etaarg))
            ptminord3 = lista[ind]
        return ptminord3
    def ordmax(lista):
        eta = []
        for j in lista:
            eta.append(raioord3(j))
        etamax = max(eta)
        ind = eta.index(etamax)
        ptmaxord3 = lista[ind]
        if eta.count(etamax)>1:
            lieta = []
            for i in range(len(eta)):
```

```
                if eta[i] == etamax:
                    lieta.append(i)
            etaarg=[]
            for j in lieta:
                lietaaux = tal(lista[j])
                etaarg.append(lietaaux)
            ind = etaarg.index(max(etaarg))
            ptmaxord3 = lista[ind]
        return ptmaxord3
if ordenamento == 4:
    alphaH = 20
    def raioord3(lista):
        aH = (1-alphaH)/2
        ro=(2*(2*m.pi)**(aH/2))*m.sqrt((1-lista[1]**aH)**2+2*(lista[1]**aH)*
        (1-m.sqrt(2*lista[1]/(1+lista[1]**2))*m.exp(-alphaH*lista[0]**2/
        (4*(1+lista[1]**2)))))/(alphaH**(5/4))
        return ro
    def angord3(lista):
        return m.atan((lista[0]**2+lista[1]**2-1)/(2*lista[0]))
    def tal(lista):
        a = angord3(lista)
        if a == 3*m.pi/2:
            return 0
        else:
            return 2*m.cos(a)/(1-m.sin(a))
    def ordmin(lista):
        eta = []
        for j in lista:
            eta.append(raioord3(j))
        etamin = min(eta)
```

```
        ind = eta.index(etamin)
        ptminord3 = lista[ind]
        if eta.count(etamin)>1:
            lieta = []
            for i in range(len(eta)):
                if eta[i] == etamin:
                    lieta.append(i)
            etaarg=[]
            for j in lieta:
                lietaaux = tal(lista[j])
                etaarg.append(lietaaux)
            ind = etaarg.index(min(etaarg))
            ptminord3 = lista[ind]
        return ptminord3
    def ordmax(lista):
        eta = []
        for j in lista:
            eta.append(raioord3(j))
        etamax = max(eta)
        ind = eta.index(etamax)
        ptmaxord3 = lista[ind]
        if eta.count(etamax)>1:
            lieta = []
            for i in range(len(eta)):
                if eta[i] == etamax:
                    lieta.append(i)
            etaarg=[]
            for j in lieta:
                lietaaux = tal(lista[j])
                etaarg.append(lietaaux)
```

```
            ind = etaarg.index(max(etaarg))
            ptmaxord3 = lista[ind]
        return ptmaxord3
if ordenamento == 5:
    def ordmin(lista):
        return MEC(lista)
    def ordmax(pontos):
        laux = []
        for i in pontos:
            laux.append(Comp(i))
        ptaux = ordmin(laux)
        return Comp(ptaux)
if ordenamento == 6:
    def ordmin(pontos):
        ptaux = MEC(pontos)
        return [ptaux[0]-ptaux[1],ptaux[1]]
    def ordmax(pontos):
        laux = []
        for i in pontos:
            laux.append(Comp(i))
        ptaux = MEC(laux)
        [x,y] = [ptaux[0]-ptaux[1],ptaux[1]]
        return Comp([x,y])
if ordenamento == 7:
    def ord6(lista):
        centro = MEC(lista)[0]
        lista1 = []
        lista2 = []
        for i in lista:
            lista1.append(d2p((centro,0),i))
```

```
        for i in range(len(lista)):
            lista2.append(m.acos(-((lista[i][0]-centro)/lista1[i])))
        return [[centro+min(lista1)*m.cos(max(lista2)),min(lista1)*m.sin(max(lista2))],
        [centro+max(lista1)*m.cos(min(lista2)),max(lista1)*m.sin(min(lista2))]]
    def ordmin(lista):
        return ord6(lista)[0]
    def ordmax(lista):
        return ord6(lista)[1]
          "'Funções Auxiliares"'
def dp(lista):
    total = 0.0
    for i in lista:
        total += (i-media(lista))**2
    total = m.sqrt(total/(len(lista)-1))
    return total
def matrizemlista(Q):
    Qlista = []
    Qlista.append(Q1[0][0])
    for u in range(c-1):
        for v in range(kaux[u]):
            Qlista.append(Q1[u+1][v])
    return Qlista
def normal(Q):
    Qnorm = Q1[:]
    Qlista = matrizemlista(Qnorm)
    med = np.mean(Qlista)
    dp1 = np.std(Qlista)
    Qnorm[0][0]= (Qnorm[0][0]-med)/dp1
    for u in range(c-1):
        for v in range(kaux[u]):
```

```
                Qnorm[u+1][v]=(Qnorm[u+1][v]-med)/dp1
    return Qnorm
Qnorm = normal(Q)
def fp(Q,i,j):
    lw = W(Qnorm,i,j)
    lf = ([np.mean(lw)/m.sqrt(2),np.std(lw)+0.0001])
    return lf
        "' QReal e QIm "'
Qf = np.ones((c,kaux[-1],2))
Qf[0][0] = fp(Q1,0,0)
for i in range(c-1):
    for j in range(kaux[i]):
        Qf[i+1][j] = fp(Q1,i+1,j)
QfReal = np.ones((c,kaux[-1]))
QfIm = np.ones((c,kaux[-1]))
QfReal[0][0] = Qf[0][0][0]/m.sqrt(2)
QfIm[0][0] = Qf[0][0][1]
for i in range(c-1):
    for j in range(kaux[i]):
        QfReal[i+1][j] = Qf[i+1][j][0]/m.sqrt(2)
        QfIm[i+1][j] = Qf[i+1][j][1]
          "'Erosão, Dilatação e Abertura"'
def erosao(Q,i,j):
    lista = []
    for q in SE(i,j):
        lista.append(Q[q[0]][q[1]].tolist())
    lq = ordmin(lista)
    return lq
def dilatacao(Q,i,j):
    lista = []
```

```
    for q in SE(i,j):
        lista.append(Q[q[0]][q[1]].tolist())
    lq1 = ordmax(lista)
    return lq1
        ''' Erosao '''
QErosao = np.ones((c,kaux[-1],2))
QErosao[0][0] = erosao(Qf,0,0)
for i in range(c-1):
    for j in range(kaux[i]):
        QErosao[i+1][j] = erosao(Qf,i+1,j)
QErosaoReal = np.ones((c,kaux[-1]))
QErosaoIm = np.ones((c,kaux[-1]))
QErosaoReal[0][0] = QErosao[0][0][0]
QErosaoIm[0][0] = QErosao[0][0][1]
for i in range(c-1):
    for j in range(kaux[i]):
        QErosaoReal[i+1][j] = QErosao[i+1][j][0]
        QErosaoIm[i+1][j] = QErosao[i+1][j][1]
        ''' Abertura '''
QAbertura = np.ones((c,kaux[-1],2))
QAbertura[0][0] = dilatacao(QErosao,0,0)
for i in range(c-1):
    for j in range(kaux[i]):
        QAbertura[i+1][j] = dilatacao(QErosao,i+1,j)
QAberturaReal = np.ones((c,kaux[-1]))
QAberturaIm = np.ones((c,kaux[-1]))
QAberturaReal[0][0] = QAbertura[0][0][0]
QAberturaIm[0][0] = QAbertura[0][0][1]
for i in range(c-1):
    for j in range(kaux[i]):
```

```
            QAberturaReal[i+1][j] = QAbertura[i+1][j][0]
            QAberturaIm[i+1][j] = QAbertura[i+1][j][1]
              "'Construindo Figura"'
              "' Parte Real e Imaginária"'
QfReal[0][0] = rees(QfReal[0][0])
QfIm[0][0] = 2*(rees(QfIm[0][0]))-1
for i in range(c-1):
    for j in range(kaux[i]):
            QfReal[i+1][j] = rees(QfReal[i+1][j])
            QfIm[i+1][j] = 2*rees(QfIm[i+1][j])-1
QReal = np.ones((P,P))
QIm = np.ones((P,P))
for i in range(P):
    for j in range(P):
            QReal[j][i] = QfReal[I[j][i][0]][I[j][i][1]]
            QIm[j][i] = QfIm[I[j][i][0]][I[j][i][1]]
result = Image.fromarray((QReal * 255).astype(np.uint8))
result.save('Bola1HParteReal.png')
result = Image.fromarray((QIm * 255).astype(np.uint8))
result.save('Bola1HParteIm.png')
              "' Erosão e Abertura"'
QErosaoReal[0][0] = rees(QErosaoReal[0][0])
QErosaoIm[0][0] = 2*(rees(QErosaoIm[0][0]))-1
for i in range(c-1):
    for j in range(kaux[i]):
            QErosaoReal[i+1][j] = rees(QErosaoReal[i+1][j])
            QErosaoIm[i+1][j] = 2*rees(QErosaoIm[i+1][j])-1
QEReal = np.ones((P,P))
QEIm = np.ones((P,P))
for i in range(P):
```

```
    for j in range(P):
        QEReal[j][i] = QErosaoReal[I[j][i][0]][I[j][i][1]]
        QEIm[j][i] = QErosaoIm[I[j][i][0]][I[j][i][1]]
result = Image.fromarray((QEReal * 255).astype(np.uint8))
result.save('ErosaoReal.png')
result = Image.fromarray((QEIm * 255).astype(np.uint8))
result.save('ErosaoIm.png')
QAberturaReal[0][0] = rees(QAberturaReal[0][0])
QAberturaIm[0][0] = 2*(rees(QAberturaIm[0][0]))-1
for i in range(c-1):
    for j in range(kaux[i]):
        QAberturaReal[i+1][j] = rees(QAberturaReal[i+1][j])
        QAberturaIm[i+1][j] = 2*rees(QAberturaIm[i+1][j])-1
QAReal = np.ones((P,P))
QAIm = np.ones((P,P))
for i in range(P):
    for j in range(P):
        QAReal[j][i] = QAberturaReal[I[j][i][0]][I[j][i][1]]
        QAIm[j][i] = QAberturaIm[I[j][i][0]][I[j][i][1]]
result = Image.fromarray((QAReal * 255).astype(np.uint8))
result.save('AberturaReal.png')
result = Image.fromarray((QAIm * 255).astype(np.uint8))
result.save('AberturaIm.png')
print(datetime.now() - startTime)
```