2 RASTREAMENTO DE CURVAS

Neste capítulo são tratados os métodos utilizados para a detecção de elementos de borda, a construção de fragmentos de curvas a partir dos elementos de borda e o aprendizado de máquina supervisionado aplicado nos dois últimos estágios aos fragmentos de curvas construídos previamente. Alguns termos, definidos na seção 1.1, são utilizados para nomear estruturas comumente vistas ao longo das seções.

2.1 Detecção de elementos de borda e o método de terceira ordem

Elementos de borda (*edgels*) em imagens em tons de cinza são pontos de descontinuidade ou variação abrupta do contraste, junto à direção de propagação. Tal comportamento é proveniente da combinação de fatores como profundidade, geometria dos objetos, cor, iluminação e sombreamento.

Inicialmente, converte-se a imagem colorida, no formato RGB, para tons de cinza. Esta dá-se através de uma combinação linear, levando-se em conta fatores perceptuais. Sejam (r, g, b) respectivamente os valores das cores vermelho, verde e azul, com $r, g, b \in [0, 1]$. A luminância relativa, utilizada para tons de cinza é

$$l(r, g, b) = 0,2125r + 0,7154g + 0,0721b.$$

O método de terceira ordem, descrito por Tamrakar e Kimia (2007), tem como intenção melhorar principalmente o cálculo da direção dos elementos de borda. Segundo os autores a atribuição da direção em função da direção do gradiente da tonalidade é errônea; o processo de cálculo das direções deve envolver derivadas de uma ordem superior às utilizadas na localização destes.

Considera-se uma imagem em tons de cinza, (a luminância ou um canal de cor), como uma função I com $I : \mathbb{R}^2 \to \mathbb{R}$, que pode ser tomada como a tonalidade, e o contraste dessa imagem, $|\nabla I(x, y)|$. Os elementos de borda são pontos onde a direção de maior variação do contraste, $\nabla|\nabla I(x, y)|$, é ortogonal à direção de maior variação de tonalidade, dada pelo gradiente normalizado $\frac{\nabla I(x,y)}{|\nabla I(x,y)|}$. Isto nos dá a equação para a localização dos elementos de borda:

$$\nabla|\nabla I(x, y)|.\frac{\nabla I(x, y)}{|\nabla I(x, y)|} = 0. \tag{2}$$

Esta pode ser reescrita como

$$I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy} = 0, \tag{3}$$

sendo

$$I_x = \frac{\partial I}{\partial x}, \quad I_{xx} = \frac{\partial^2 I}{\partial x^2}, \quad I_{xy} = \frac{\partial^2 I}{\partial x \partial y}, \quad I_y = \frac{\partial I}{\partial y} \quad I_{yy} = \frac{\partial^2 I}{\partial y^2}. \tag{4}$$

Seja $\mathcal{I}(x, y) = I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy}$. A equação 3 pode ser revista como $\mathcal{I}(x, y) = 0$. Esta equação representa todos os contornos como a curva (conjunto) de nível de valor zero da função $\mathcal{I}$.

A orientação dos pontos pertencentes a esta curva é a tangente à curva nos mesmos. Do cálculo, sabemos que a tangente de uma curva de nível é ortogonal ao gradiente da curva de nível naquele ponto, Bouchara, Zanetic e Hellmeister (1996). De fato, seja $\boldsymbol{\lambda}(t) = (x(t), y(t))^\top$, onde a imagem de $\boldsymbol{\lambda}$ são os pontos pertencentes à curva de nível $\mathcal{I}(x, y) = c$, onde c é uma constante. Derivando ambos os lados

$$\frac{d}{dt}\mathcal{I}(\boldsymbol{\lambda}(t)) = \frac{dc}{dt}, \tag{5}$$

e aplicando a regra da cadeia tem-se,

$$\nabla \mathcal{I}(\boldsymbol{\lambda}(t)) \cdot \boldsymbol{\lambda}'(t) = 0, \tag{6}$$

demostrando que o gradiente da função nos pontos pertencentes a $\boldsymbol{\lambda}(t)$ é perpendicular a $\boldsymbol{\lambda}'(t)$, ou seja, ele é perpendicular à tangente da curva de nível c de $\mathcal{I}$. Então a normal da orientação do elemento de borda pode ser calculada:

$$\boldsymbol{n} = \nabla \mathcal{I}\Big|_{\mathcal{I}(x,y)=0} = (\mathcal{I}_x, \mathcal{I}_y)^\top \tag{7}$$

$$= \left((I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy})_x, (I_x^2 I_{xx} + 2I_x I_y I_{xy} + I_y^2 I_{yy})_y\right)^\top$$

$$= \begin{pmatrix} 2I_x^2 I_{xx} + 2I_x I_{xy}^2 + 2I_y I_{xx} I_{xy} + 2I_y I_{yy} I_{xy} + 2I_x I_y I_{xxy} + 2I_y^2 I_{xyy} + I_x^2 I_{xxx} \\ 2I_y^2 I_{yy} + 2I_y I_{xy}^2 + 2I_x I_{xx} I_{xy} + 2I_x I_{yy} I_{xy} + 2I_x I_y I_{xyy} + 2I_x^2 I_{xxy} + I_y^2 I_{yyy} \end{pmatrix},$$

e a tangente é dada por $\boldsymbol{n}^\top = (-\mathcal{I}_y, \mathcal{I}_x)$, logo o ângulo do *edgel* satisfaz $\tan\theta = -\frac{\mathcal{I}_x}{\mathcal{I}_y}$.

Com isso temos a base para o cálculo dos pontos de borda e suas direções utilizando derivadas até terceira ordem. Mais detalhes sobre este método podem ser encontrados em Tamrakar (2008), onde há também a formulação para imagens coloridas, e como calcular as derivadas terceiras numericamente de forma robusta quando usadas em escalas

pequenas, além de outras informações sobre a implementação deste método.

2.2 Vinculador simbólico de elementos de borda

Esta seção descreve funcionalmente a clusterização de *edgels* em curvas suaves proposta em Tamrakar e Kimia (2007) — *symbolic edge linker* — que pode ser dividida em duas etapas, a clusterização local e a global.

2.2.1 Clusterização local

Após a determinação dos elementos de borda, se constrói o mapa de bordas que está representado pictoricamente na Figura 17, esquerda. Pode-se notar que determinadas posições espaciais contém mais de um *edgel*, enquanto a maioria das posições não contém nenhum.

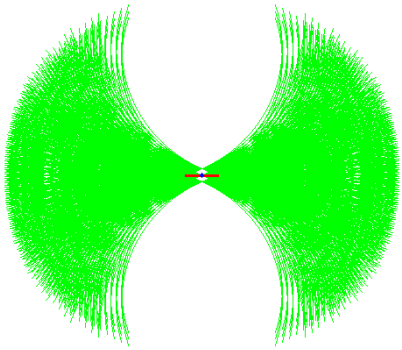
A primeira etapa inicia a partir do mapa de bordas e consiste em encontrar na vizinhança de cada *edgel* uma curva geometricamente coerente. Para isto é usado o modelo de curva de arco circular. Conforme mencionado brevemente no capítulo anterior, cada *edgel* tem o seu *curve bundle* ou feixe de curvas viáveis, Figura 16, descrito por um conjunto de parâmetros $(\delta n, \delta\theta, \delta k)$. Esses parâmetros representam todas as possíveis curvas, relativas ao modelo de curva de arco circular, passando por perturbações do *edgel* com posição $\boldsymbol{p} = (x, y)$, orientação θ da tangente ao ponto $\boldsymbol{p}$, e k a curvatura, que é proporcionalmente inversa ao raio desses arcos circulares.

Encontrar todas as *curvelets* dentro de uma vizinhança, formadas por *edgels* que têm curvas em comum em seus feixes, usando todas as possíveis permutações de *edgels* nesta vizinhança seria muito custoso. Para superar este fato, a seguinte solução foi proposta por Tamrakar (2008).

Constrói-se o grafo $G(V, E)$, de edgels vizinhos ou ***ENG*** (*edgel neighborhood graph*), onde cada vértice $v \in V$ é um *edgel* e as arestas são definidas como $e(v_i, v_j) \in E \leftrightarrow \text{dist}(v_i, v_j) < \epsilon$, com dist a distância euclidiana em píxeis e ϵ um limiar escolhido. A Figura 17 centro, ilustra um destes grafos.

Para cada *edgel* no mapa de bordas, faz-se este *edgel* âncora, centro de uma vizinhança, então computa-se a projeção ortogonal dos *edgels* vizinhos na linha paralela à orientação do *edgel* âncora. Obtém-se uma ordem viável para os *edgels* vizinhos ao âncora seguindo a distância das projeções ao *edgel* âncora. Ordenando a distância das projeções obtém-se uma ordem viável para os *edgels*. Com esta ordenação, constrói-se as *curvelets* sequencialmente usando busca em profundidade (***DFS***, *depth first search*) no grafo de *edgels* vizinhos e a cada novo *edgel* adicionado à *curvelet*, reduz-se o espaço de parâme-

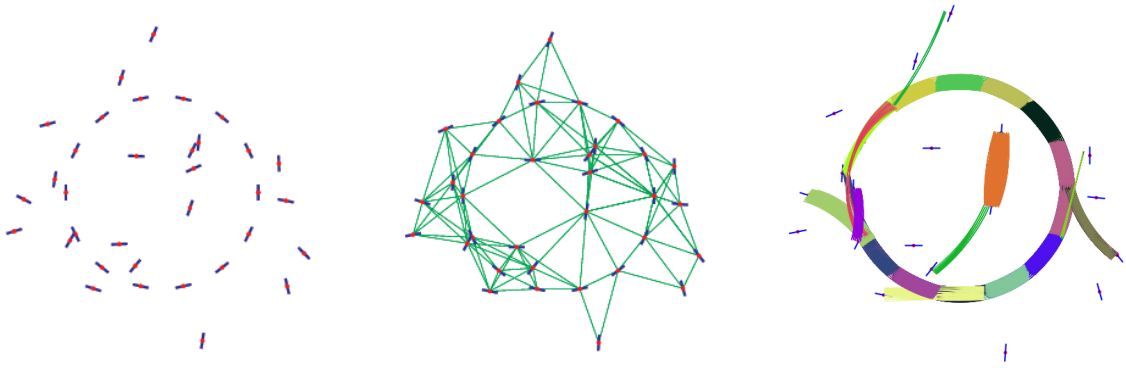
Figura 16 - Feixe de curvas de arco circulares.



Legenda: *Curve bundle* - feixe de curvas de arco circulares onde cada curva é etiquetada por $(\delta n, \delta\theta, \delta k)$, passando por perturbações de um *edgel* $\boldsymbol{e} = (\boldsymbol{p}, \theta, \Delta)$. Essas curvas passam pelo ponto $\boldsymbol{p} \pm (\delta n + \frac{1}{k \pm \delta k})(-\sin(\theta + \delta\theta), \cos(\theta + \delta\theta)) + \frac{1}{k \pm \delta k}(\cos\alpha, \sin\alpha)$, com δn a perturbação na posição, perpendicular à direção $(\cos\theta, \sin\theta)$, $\delta\theta$ a perturbação na direção θ, δk a perturbação na curvatura e $\alpha \in [\theta - \pi, \theta]$ ou $\alpha \in [\theta, \theta + \pi]$ respectivamente para o caso positivo e negativo. Este é o primeiro passo na obtenção de *curvelets*.

Fonte: Tamrakar (2008).

Figura 17 - Etapas na construção de *curvelets*



Legenda: [Esquerda] *Edgels* espacialmente organizados no mapa de bordas.
[Centro] ***ENG***, grafo de *edgels* vizinhos: os vértices são os *edgels* e as arestas as linhas em verde.
[Direita] *curvelets* com seus feixes de curvas em diferentes cores.

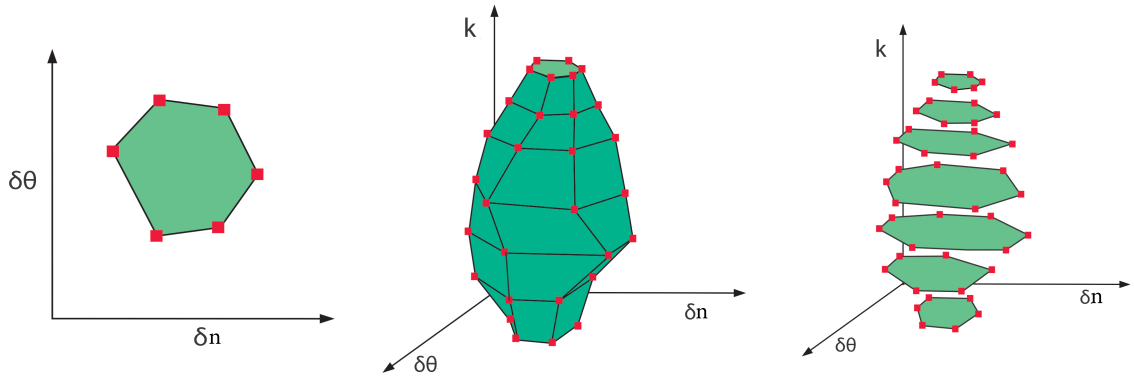
Fonte: (TAMRAKAR, 2008).

Figura 18 - Restrição do feixe de curvas conforme a adição de novos *edgels*.

Legenda: Restrição do feixe de curvas conforme a adição de novos *edgels*.
Inicia-se com o *edgel* A e seu feixe de curvas. Com a adição do *edgel* Z, o mais próximo de A, reduz-se o feixe de curvas, em seguida com a adição do *edgel* B descartam-se mais semi-círculos desse feixe de curvas, e assim sucessivamente para os outros *edgels*.

Fonte: (TAMRAKAR, 2008; GUO et al., 2014c).

Figura 19 - Representação de um feixe de curvas no espaço de parâmetros.



Legenda: Representação de um feixe de curvas no espaço de parâmetros $(\delta n, \delta\theta, k)$.
[Esquerda] A projeção em δn e $\delta\theta$.
[Centro] O poliedro do feixe de curvas de arco circular nas 3 dimensões.
[Direita] Discretização em uma das dimensões, no caso k, feita para reduzir a computação necessária para se encontrar as interseções.
Fonte: (TAMRAKAR, 2008).

tros do feixe de curvas. Desta forma, para cada *edgel* adicionado, se este não pertencer ao feixe de curvas atual, ou seja, se o feixe de curvas do novo *edgel* não tem interseção com o atual, qualquer *curvelet* subsequente nesta ordem de busca no grafo é descartada, diminuindo consideravelmente a quantidade de testes realizados. Na Figura 18 é possível ver a diminuição considerável dos semi-círculos do feixe de curvas conforme a adição de um novo *edgel*.

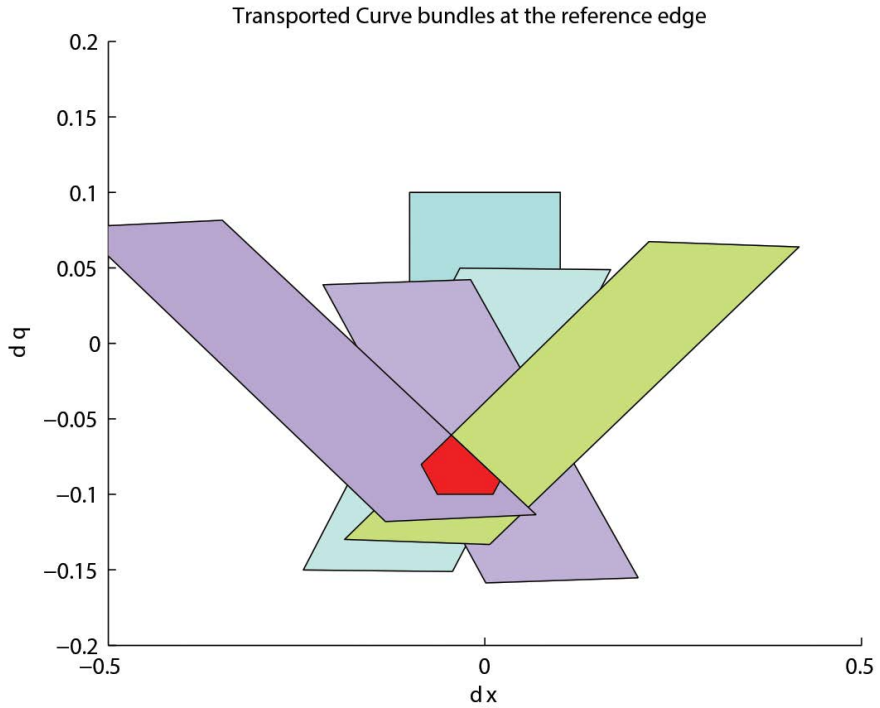
A busca destes feixes de curvas é feito no espaço de parâmetros (semelhante à transformada de Hough); pode ser visto na Figura 19 o poliedro das curvas possíveis das perturbações de um *edgel* e_1 no espaço de parâmetros. Este poliedro é transportado para o espaço de parâmetros do *edgel* e_2 e feita a intercessão neste espaço, Tamrakar (2008). A Figura 20 ilustra a redução do espaço, conforme planos do espaço de parâmetros são adicionados.

O resultado final deste processo é o mapa de *curvelets* discreto, contendo todas as possíveis *curvelets*, para todos os *edgels*, encontrados com o processo de transporte e redução do espaço de parâmetros.

2.2.2 Clusterização global

Nesta etapa procura-se o agrupamento global usando o modelo de curva poli-arco circular, construída por *curvelets* que compartilham um mesmo ponto e tangente numa extremidade. Porém, para construir estas curvas, é necessário remover as ambiguidades

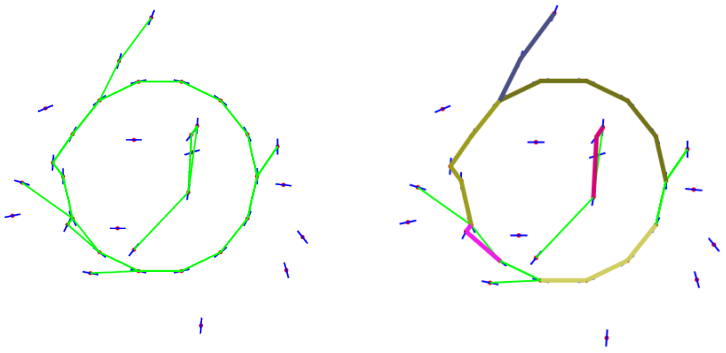
Figura 20 - Resultado das interseções dos espaços de curva de cada *edgel*.



Legenda: Resultado das interseções dos espaços de curva de cada *edgel*, reduzindo o espaço total de parâmetros (espaço indicado em vermelho).

Fonte: (TAMRAKAR, 2008).

Figura 21 - Etapas na extração de fragmentos de curva não-ambíguos



Legenda: [Esquerda] ***ETG*** construído a partir do ***ENG*** e o mapa de *curvelets* discreto.
[Direita] fragmentos de curvas não-ambíguas extraídos a partir do ***ETG***.
Fonte: (GUO et al., 2014c).

dos vários *curvelets* que possuem *edgels* em comum.

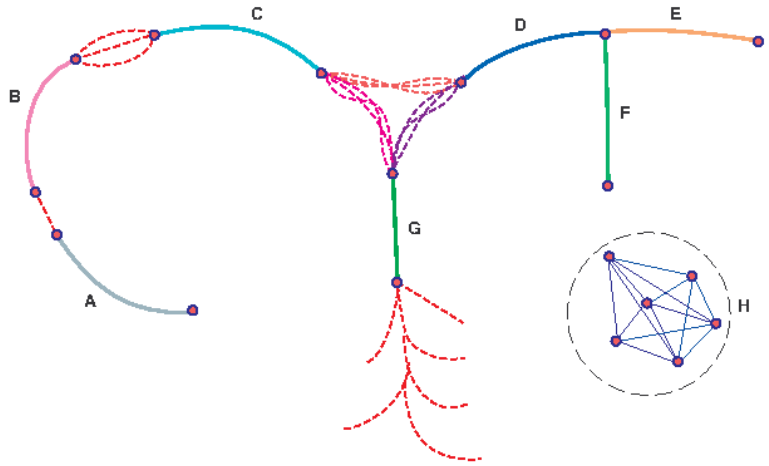
O primeiro passo para a construção das curvas é a extração de fragmentos de curvas não ambíguos. Para extrair tais fragmentos, é construído o grafo topológico de *edgels* ***ETG*** (*edgel topological graph*) a partir do ***ENG*** e das *curvelets* extraídas na clusterização local. O ***ETG*** pode ser definido como o grafo $G(V, E)$ onde os vértices v são *edgels* tais que $v \in V \leftrightarrow \exists C \ni v$, para C uma *curvelet* no mapa de *curvelets* discreto, as arestas $e(v_i, v_j) \in E \leftrightarrow \exists C$ tal que $v_i, v_j \in C$ e $j = i \pm 1$. Extrair os fragmentos de curvas, sequências de *edgels* conexos consistentes com as *curvelets*, não ambíguos desse grafo, torna-se simplesmente extrair em ordem os vértices conexos de grau um que obedecem o modelo de curva de arco circular, ilustrado na Figura 21.

Desses fragmentos de curvas não ambíguos constrói-se o grafo de fragmentos de curvas ***CFG*** (*curve fragment graph*), equivalente ao ***ENG***, porém os vértices são os fragmentos de curva e cada aresta representa uma possível ligação entre extremidades desses fragmentos.

Para explorar todas as possíveis ambiguidades são utilizadas árvores de hipótese, com raiz na extremidade não ambígua de um fragmento de curva. Esta árvore é construída a partir do grafo ***CFG*** utilizando a busca em profundidade. Cada possível caminho na árvore representa junções viáveis entre os fragmentos de curvas atendendo a uma das hipóteses de continuidade:

- C^2 (arco circular) — pertencer ao mesmo feixe de curvas. Os *edgels* $\{\boldsymbol{e}_0, ..., \boldsymbol{e}_s\}$ têm o mesmo feixe de curvas $(\delta n, \delta\theta, \delta k)$ em comum;
- C^1 (poli-arco circular) — compartilharem a mesma tangente nas extremidades de seus feixes de curvas. Os *edgels* $\{\boldsymbol{e}_0, ..., \boldsymbol{e}_{s_1}, ..., \boldsymbol{e}_{s_r}\}$ formam arcos circulares $\{\{\boldsymbol{e}_0, ..., \boldsymbol{e}_{s_1}\}, ..., \{\boldsymbol{e}_{s_1}, ..., \boldsymbol{e}_{s_2}\}, ..., \{\boldsymbol{e}_{s_{r-1}}, ..., \boldsymbol{e}_{s_r}\}\}$ com seus respectivos feixes de curvas

Figura 22 - ***CFG*** Fragmentos de curvas não-ambíguas



Legenda: ***CFG*** Fragmentos de curvas não-ambíguas coloridas e possíveis junções tracejadas.
Fonte: (GUO et al., 2014c).

$\{(\delta n_1, \delta\theta_1, \delta k_1),\ ...,\ (\delta n_r, \delta\theta_r, \delta k_r)\}$ onde em cada $\boldsymbol{e}_{s_i}$ existe a interseção entre os parâmetros $\delta\theta_i$ e $\delta\theta_{i+1}$ dos feixes de curvas que contêm o *edgel* $\boldsymbol{e}_{s_i}$.

Como podem existir várias arestas ligando dois vértices no ***CFG***, como visto na Figura 22, pois cada árvore de hipóteses construída para diferentes vértices pode gerar arestas diferentes, é necessário agrupar tais arestas em classes de equivalência, arestas ligando os mesmos dois pontos, dando origem a um novo grafo chamado de grafo topológico de fragmentos de curvas ***CFTG*** (*curve fragments topological graph*), Figura 23.

Finalmente resolvem-se as ambiguidades de forma gulosa (*greedy*) minimizando o custo total da curva, priorizando o comprimento e suavidade da curva, dado por

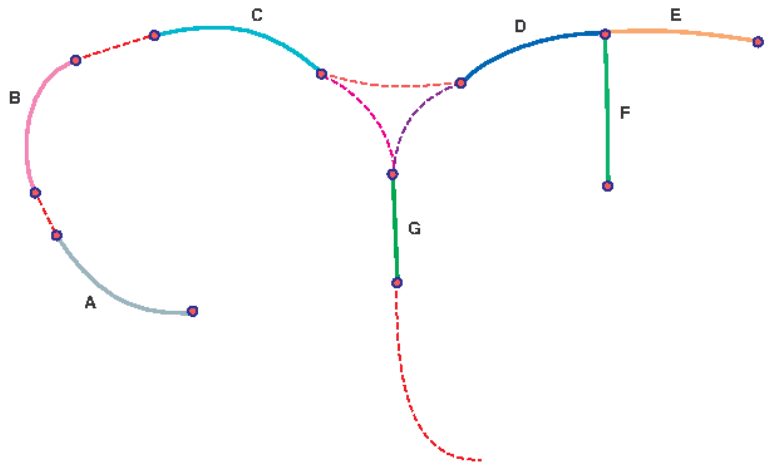
$$\text{custo}(\{C_1, ..., C_n\}) = \sum_{i=1}^{n-1} \frac{|k_i - k_{i+1}|}{l_i + l_{i+1}}$$

com C_i denotando as *curvelets* do possível poli-arco, com suas respectivas curvaturas k_i e comprimentos l_i. Na Figura 24 é ilustrado um possível caminho após a remoção das ambiguidades.

2.3 Aprendizado de máquina

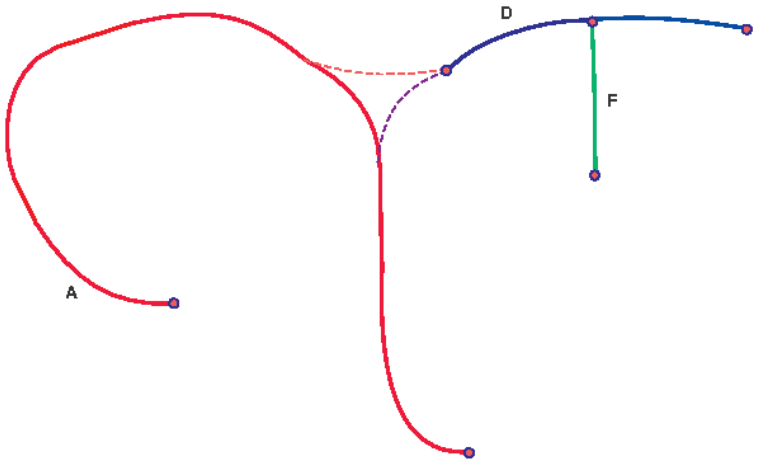
Nesta seção, são tratados os métodos de aprendizado de máquina aplicados aos fragmentos de curvas extraídas com o vinculador simbólico de *edgels*, a fim de aprimorar qualidades de totalidade, distinção e objetividade desse conjunto de fragmentos de curvas. Para isto três classificadores serão utilizados, um para cada qualidade.

Figura 23 - ***CFTG*** Fragmentos de curvas não-ambíguas em cores sólidas e classes de equivalência das possíveis junções tracejadas.



Legenda: ***CFTG*** Fragmentos de curvas não-ambíguas em cores sólidas e classes de equivalência das possíveis junções tracejadas.
Fonte: (GUO et al., 2014c).

Figura 24 - ***CFTG*** após remoção das ambiguidades.



Legenda: ***CFTG*** após remoção das ambiguidades, em vermelho a curva com menor custo.
Fonte: (TAMRAKAR, 2008).

Nas seções seguintes são apresentados o modelo geral utilizado e as características escolhidas como entrada do modelo com suas respectivas formulações matemáticas.

2.3.1 Modelo logístico

O modelo ou classificador logístico é utilizado para categorização binária de dados, 0 ou 1, representando, por exemplo, não pertence ou pertence. Neste modelo temos como entrada as variáveis independentes, no nosso contexto, as características extraídas dos fragmentos de curva e como saída a classificação deste fragmento, mais exatamente a probabilidade de que este pertença à classe em questão.

Esse modelo tem sua base na função logística ou sigmóide,

$$f(t) = \frac{1}{1 + e^{-t}} = \frac{e^t}{1 + e^t}, \tag{8}$$

que tem sua imagem definida em $(0, 1)$, o que a torna viável para a desejada interpretação de probabilidades. No modelo logístico, a variável t é substituída por uma função linear das características, $\boldsymbol{x} = (x_1, ..., x_n)$, da forma,

$$g(\boldsymbol{x}) = w_0 + \sum_{i=1}^{n} w_i x_i = w_0 + w_1 x_1 + w_2 x_2 + ... + w_n x_n, \tag{9}$$

onde w_0 é o viés do modelo. Desse modo temos:

$$f \circ g = \frac{1}{1 + e^{-\left(w_0 + \sum_{i=1}^{n} w_i x_i\right)}}.$$

Deseja-se encontrar os pesos, w_i, que melhor representem os dados, ou seja, tendo as classificações $y^j \in \{0, 1\}$ pré-determinadas para alguns casos conhecidos com entradas $\boldsymbol{x}^j$, $(\boldsymbol{x}^j \leftrightarrow y^j)$, o modelo deve generalizar para outros casos de tal forma que tenha uma probabilidade alta para os casos verdadeiros e baixa para os casos falsos.

Embora este seja um modelo relativamente simples, bons resultados podem ser obtidos quando observadas algumas premissas:

- as características x_i têm real influência sobre o processo e classificação;
- existem exemplos suficientes para o aprendizado;
- haja baixa multicolinearidade entre as variáveis independentes x_i.

Ajuste do modelo

O ajuste do modelo caracteriza-se por encontrar os pesos w_i tais que o erro de classificação seja mínimo. Uma forma de se alcançar isto é maximizando a verossimilhança entre a classificação obtida pelo modelo e as amostras $(\boldsymbol{x}^j, y^j)$. Uma função de verossimilhança comumente utilizada para o modelo logístico é a logarítmica.

Sejam $\boldsymbol{x}^j$ as variáveis independentes associadas a um amostra y^j, n o número de variáveis independentes, m o número de amostras e $\boldsymbol{w}$ os pesos do modelo; definimos a função de verossimilhança logarítmica para o modelo logístico,

$$\mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{m} y^i \log(p^i(\boldsymbol{w})) + (1 - y^i) \log(1 - p^i(\boldsymbol{w})), \tag{10}$$

com

$$p^i(\boldsymbol{w}) = \frac{1}{1 + e^{-\left(w_0 + \sum_{j=1}^{n} w_j x_j^i\right)}}.$$

Do cálculo, sabe-se que, muitas vezes, o ponto de máximo ou maximizador, de uma função pode ser encontrado no ponto crítico onde sua derivada é zero; neste caso buscamos $\boldsymbol{w}^*$ tal

$$\nabla \mathcal{L}|_{\boldsymbol{w}^*} = 0, \tag{11}$$

onde ∇ é o gradiente.

Como não é possível encontrar uma fórmula fechada explícita para $\boldsymbol{w}^*$ na equação 11, recorre-se aos métodos numéricos para solucioná-la. O método utilizado aqui para encontrar uma raiz de uma função é o de Newton-Raphson que no caso unidimensional consiste em aproximar uma função $f(x)$ em um ponto x_i por sua fórmula de Taylor e iterativamente encontrar a raiz x^* para esta aproximação:

$$f(x) = f(x_i) + (x - x_i) f'(x_i) + \mathcal{O}\left((x - x_i)^2\right),$$
$$f(x) \approx f(x_i) + (x - x_i) f'(x_i),$$

onde $\mathcal{O}\left((x - x_i)^2\right)$ representa os termos de mais alta ordem.

É fácil encontrar x^* tal que $f(x^*) \approx 0$, isto é, $f'(x_i)(x^* - x_i) + f(x_i) = 0$, donde, com alguma manipulação algébrica se obtém

$$x^* = x_i - \frac{f(x_i)}{f'(x_i)}.$$

Aplicando-se este método ao problema de encontrar o zero da derivada primeira, $f'(x) = 0$,

encontra-se:

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}. \tag{12}$$

Finalmente para o caso multivariável, considerando a equação 12, tem-se que

$$\boldsymbol{w}_{i+1} = \boldsymbol{w}_i - \mathcal{H}^{-1}(\mathcal{L})|_{\boldsymbol{w_i}} \nabla \mathcal{L}|_{\boldsymbol{w_i}}, \tag{13}$$

onde $\mathcal{H}(\mathcal{L})$ é a matriz hessiana, e o índice i indica o contador do método. Assim, $\mathcal{H}_{i,j}(\mathcal{L}) = \frac{\partial^2 \mathcal{L}}{\partial w_i \partial w_j}$ e $\nabla_i \mathcal{L} = \frac{\partial \mathcal{L}}{\partial w_i}$. Da equação 10;

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \frac{\partial}{\partial w_j}\left(\sum_{i=1}^{m} y^i \log(p^i) + (1-y^i)\log(1-p^i)\right) \\
&= \sum_{i=1}^{m} y^i \frac{\partial}{\partial w_j}\Big(\log(p^i)\Big) + \sum_{i=1}^{m}(1-y^i)\frac{\partial}{\partial w_j}\Big(\log(1-p^i)\Big).
\end{aligned}$$

Aplicando a regra da cadeia

$$\frac{\partial \mathcal{L}}{\partial w_j} = \sum_{i=1}^{m} y^i \frac{1}{p^i}\frac{\partial p^i}{\partial w_j} - \sum_{i=1}^{m}(1-y^i)\frac{1}{1-p^i}\frac{\partial p^i}{\partial w_j}. \tag{14}$$

Das equações 8 e 9, e sendo $g^i = g(\boldsymbol{x}^i)$,

$$\begin{aligned}
\frac{df}{dt} &= \frac{e^t}{(e^t+1)^2} = f(t)(1-f(t)), \\
\frac{\partial g^i}{\partial w_j} &= x_j^i, \\
\frac{\partial p^i}{\partial w_j} &= \frac{\partial}{\partial w_j}\Big(f \circ g^i\Big) = \frac{\partial f}{\partial g^i}\frac{\partial g^i}{\partial w_j} = (f \circ g^i)(1-(f \circ g^i))x_j^i = p^i(1-p^i)x_j^i.
\end{aligned}$$

Substituindo na equação 14:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial w_j} &= \sum_{i=1}^{m} y^i \frac{1}{p^i} p^i(1-p^i)x_j^i - \sum_{i=1}^{m}(1-y^i)\frac{1}{1-p^i}p^i(1-p^i)x_j^i, \\
&= \sum_{i=1}^{m} y^i(1-p^i)x_j^i - (1-y^i)p^i x_j^i, \\
&= \sum_{i=1}^{m}(y^i - p^i)x_j^i,
\end{aligned}$$

e para a hessiana:

$$\begin{aligned}\frac{\partial^2 \mathcal{L}}{\partial w_j \partial w_i} &= \frac{\partial}{\partial w_j}\left(\sum_{k=1}^{m}(y^k - p^k)x_i^k\right) \\ &= \sum_{k=1}^{m} -x_i^k \frac{\partial p^k}{\partial w_j} \\ &= \sum_{k=1}^{m} -x_i^k p^k (1-p^k) x_j^k.\end{aligned}$$

2.3.2 Classificadores e características

Nesta seção, são apresentados os classificadores utilizando o modelo logístico geral descrito anteriormente e as variáveis independentes que são chamadas características proveniente das curvas e da imagem de onde as curvas são obtidas. Essas características podem ser divididas em fotométricas, adquiridas diretamente da imagem, e geométricas adquiridas das curvas.

Três classificadores são utilizados, um para cada qualidade desejada. O primeiro, relativo à totalidade, é chamado de *merge*, e tem por objetivo classificar se um par de curvas C_1 e C_2 pertence ou não a um mesmo contorno. Este tem como entradas

$$\boldsymbol{x}_{merge} = \begin{pmatrix} \mathbf{\Delta bg\_grad} \\ \mathbf{\Delta sat\_grad} \\ \mathbf{\Delta hue\_grad} \\ \mathbf{\Delta abs\_k} \\ \mathbf{\Delta edge\_sparsity} \\ \mathbf{\Delta wigg} \\ \mathbf{geom\_diff} \\ \mathbf{tex\_diff} \end{pmatrix},$$

com $\mathbf{\Delta caracterßstica} = \mathbf{caracterßstica(C_1)} - \mathbf{caracterßstica(C_2)}$, a diferença entre uma característica calculada respectivamente para as curvas C_1 e C_2, onde as características são dadas na tabela 1.

O segundo classificador, chamado *break* geométrico, é relativo à distinção quando uma curva contém parte de dois contornos distintos. Este tem como entrada:

$$\boldsymbol{x}_{break} = \left(\mathbf{geom\_diff}\right).$$

Finalmente o terceiro classificador chamado *rank*, relativo à objetividade, tem en-

Tabela 1 - Características utilizadas nos classificadores.

bg_grad	*average background gradient*, diferença média de brilho, neste caso a componente "V" no espaço de cor HSV, entre os lados da curva na direção da normal
sat_grad	*average saturation gradient*, diferença média de saturação entre os lados da curva na direção da normal
hue_grad	*average hue gradient*, diferença média de matiz entre os lados da curva na direção da normal
abs_k	*average absolute curvature*, média do valor absoluto da curvatura ao longo da curva
edge_sparsity	*edge sparsity*, média de *edgels* no entorno da curva
wigg	*wiggle*, média dos índices relativos aos pontos onde há mudança de sinal na curvatura
len	*length*, comprimento da curva
avg_conf	*average edge strength*, intensidade média das bordas que compõem a curva
geom_diff	*geometry difference* cosseno do ângulo entre duas curvas que têm um ponto de extremidade em comum
tex_diff	*texture difference*, diferença de textura entre duas curvas que têm um ponto de extremidade em comum

Fonte: O autor, 2017.

tradas:

$$\boldsymbol{x}_{rank} = \begin{pmatrix} \textbf{bg\_grad} \\ \textbf{sat\_grad} \\ \textbf{hue\_grad} \\ \textbf{abs\_k} \\ \textbf{edge\_sparsity} \\ \textbf{wigg} \\ \textbf{len} \\ \textbf{mean\_conf} \end{pmatrix}.$$

Com as características descritas na tabela 1, tem-se informações sobre cor, geometria e textura, provenientes da imagem colorida, do mapa de bordas e dos fragmentos de curvas.

2.3.3 Características bg_grad, sat_grad e hue_grad

Para calcular as características **bg_grad**, **sat_grad** e **hue_grad** ao longo de uma curva discreta C_1, tem-se como pré-requisitos:

- converter a imagem RGB para HSV;
- calcular as normais em cada ponto da curva.

2.3.3.1 Conversão RGB para HSV

Imagens no formato RGB e HSV são, de forma simplificada, matrizes ou vetores 2D, I_{rgb} ou I_{hsv}, onde cada elemento indexado por duas coordenadas, $I(i, j)$, é uma tripla (x, y, z). Para imagens no formato RGB essa tripla representa as cores vermelho, verde e azul,(r, g, b) (com r — *red*, g — *green*, b — *blue*), onde cada componente pode assumir valores no intervalo $[0, 1]$; já para imagens no formato HSV essa tripla representa matiz, saturação e valor (h, s, v) (com h — *hue*, s — *saturation*, v — *value*), onde a componente h representa cores por um ângulo e pode assumir valores entre $[0, 360]$, e s, v assumem valores em $[0, 1]$, sendo v outra forma de representar a luminância relativa descrita na seção 2.1.

A conversão de RGB para HSV é feita com a seguinte função. Seja (r, g, b) a entrada, max = máximo$\{r, g, b\}$, min = mínimo$\{r, g, b\}$. Então,

$$h(r,g,b) = \begin{cases} 60\frac{g-b}{\max-\min}, & \text{se} \quad \max = r, \quad g \geq b \\ 60\frac{g-b}{\max-\min} + 360, & \text{se} \quad \max = r, \quad g < b \\ 60\frac{b-r}{\max-\min} + 120, & \text{se} \quad \max = g \\ 60\frac{r-g}{\max-\min} + 240, & \text{se} \quad \max = b \end{cases},$$

$$s(r,g,b) = \frac{\max}{\max-\min},$$

$$v(r,g,b) = \max.$$

2.3.3.2 Cálculo da normal a uma curva discreta

Seja $\boldsymbol{c}(t) = (f(t), g(t))$ uma curva parametrizada. Toma-se a direção da tangente à curva no ponto (x, y) como,

$$(t_x, t_y) = (f'(t), g'(t))$$

e a normal o vetor perpendicular normalizado,

$$(n_x, n_y) = \frac{1}{|(t_x, t_y)|}(t_x, t_y)^{\perp} = \frac{1}{\sqrt{t_y^2 + t_x^2}}(t_y, -t_x) = \frac{1}{\sqrt{f'^2 + g'^2}}(g', -f')$$

A normal ao longo de uma curva, neste caso, uma curva discreta é uma sequência de pares ordenados (x_i, y_i), é calculada aproximando o vetor tangente da curva através da fórmula de Taylor e calculando o vetor perpendicular a esta tangente 2D. Da fórmula de Taylor tem-se:

$$\phi(x + \Delta) = \phi(x) + \phi'(x)\frac{\Delta}{1!} + \mathcal{O}(\Delta^2),$$
$$\phi(x - \Delta) = \phi(x) - \phi'(x)\frac{\Delta}{1!} + \mathcal{O}(\Delta^2).$$

Pode-se aproximar a derivada primeira com qualquer uma das expressões a seguir:
Diferença avançada

$$\phi(x + \Delta) - \phi(x) = \phi'(x)\frac{\Delta}{1!} + \mathcal{O}(\Delta^2)$$
$$\phi'(x) \approx \frac{\phi(x + \Delta) - \phi(x)}{\Delta}.$$

Diferença recuada

$$\phi(x - \Delta) - \phi(x) = -\phi'(x)\frac{\Delta}{1!} + \mathcal{O}(\Delta^2)$$
$$\phi'(x) \approx \frac{\phi(x) - \phi(x - \Delta)}{\Delta}.$$

Diferença centrada

$$\phi(x + \Delta) - \phi(x - \Delta) = 2\phi'(x)\frac{\Delta}{1!} + \mathcal{O}(\Delta^2)$$
$$\phi'(x) \approx \frac{\phi(x + \Delta) - \phi(x - \Delta)}{2\Delta},$$

Aplica-se qualquer das formulações à curva desejada. Neste trabalho, para os casos das extremidades utiliza-se diferença avançada e recuada, já para os pontos no interior utiliza-se diferenças centradas, que apresentam uma melhor aproximação.

Seja $\boldsymbol{c}(t) = (f(t), g(t))$ uma curva paramétrica com $\boldsymbol{c}(t_i) = (x_i, y_i)$, então aproximam-se suas derivadas f' e f':

$$\frac{x_1 - x_0}{\Delta_0} = f'(t_0) \qquad \frac{x_{n-1} - x_n}{\Delta_n} = f'(t_n)$$
$$\frac{y_1 - y_0}{\Delta_0} = g'(t_0) \qquad \frac{y_{n-1} - y_n}{\Delta_n} = g'(t_n)$$

$$\frac{x_{i+1} - x_{i-1}}{\Delta_i} = f'(t_i)$$
$$\frac{y_{i+1} - y_{i-1}}{\Delta_i} = g'(t_i)$$

com $\Delta_0 = t_1 - t_0$, $\Delta_n = t_n - t_{n-1}$ e $\Delta_i = t_{i+1} - t_{i-1}$, $i = 1, ..., n-1$.

2.3.3.3 Cálculo das características **bg_grad**, **sat_grad** e **hue_grad**

Conhecendo a normal e a conversão de RGB para HSV calculam-se as características ao longo da curva como dado a seguir. Seja (n_x^i, n_y^i) a direção normal em cada ponto (x_i, y_i) da curva C, com $0 \leq i < n$, $I_{h|s|v}(x, y)$ os valores $(h|s|v)$ da imagem em (x, y) e r o raio da vizinhança da curva. Então,

$$\textbf{bg\_grad} = \frac{1}{n}\sum_{i=0}^{n-1} I_v(x_i + rn_x^i, y_i + rn_y^i) - I_v(x_i - rn_x^i, y_i - rn_y^i)$$

$$\textbf{sat\_grad} = \frac{1}{n}\sum_{i=0}^{n-1} I_s(x_i + rn_x^i, y_i + rn_y^i) - I_s(x_i - rn_x^i, y_i - rn_y^i)$$

$$\textbf{hue\_grad} = \frac{1}{n}\sum_{i=0}^{n-1} I_h(x_i + rn_x^i, y_i + rn_y^i) - I_h(x_i - rn_x^i, y_i - rn_y^i)$$

2.3.4 Características abs_k e wigg

As características **abs_k** e **wigg** são obtidas a partir da curvatura com sinal. A curvatura k num ponto (x, y) é definida como a variação do ângulo tangente Φ em relação ao comprimento de arco s,

$$k = \frac{d\Phi}{ds}, \quad \text{ou} \qquad\qquad k = \frac{d\Phi/dt}{ds/dt}$$

2.3.4.1 Curvatura

Proposição: Seja $\boldsymbol{\alpha}(t) = (x(t), y(t))$ uma curva 2D paramétrica então

$$k(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{\left(\sqrt{x'(t)^2 + y'(t)^2}\right)^3}.$$

Demonstração: A tangente à curva é dada por $\boldsymbol{\alpha}'(t) = (x'(t), y'(t))$ então o comprimento de arco de curva, s, é dado por:

$$s(t) = \int_0^t ||\boldsymbol{\alpha}'(t)||dt,$$

logo sua derivada é a norma da tangente

$$\frac{ds}{dt} = ||\boldsymbol{\alpha}'(t)|| = \sqrt{x'(t)^2 + y'(t)^2}.$$

Por outro lado,

$$\tan\Phi = \frac{y'(t)}{x'(t)},$$

e aplicando a regra da cadeia:

$$\frac{d}{dt}(\tan\Phi) = \frac{d}{d\Phi}(\tan\Phi)\frac{d\Phi}{dt},$$
$$\frac{d\Phi}{dt} = \frac{1}{\frac{d}{d\Phi}(\tan\Phi)}\frac{d}{dt}(\tan\Phi),$$

donde,

$$\frac{d\Phi}{dt} = \frac{1}{\sec^2\Phi}\frac{d}{dt}(\tan\Phi).$$

Mas, derivando,

$$\frac{d}{dt}(\tan\Phi) = \frac{d}{dt}\left(\frac{y'(t)}{x'(t)}\right) = \frac{x'(t)y''(t) - y'(t)x''(t)}{x'(t)^2},$$

e usando a relação trigonométrica,

$$\sec\Phi = \frac{1}{\cos\Phi},$$

obtém-se

$$\cos\Phi = \frac{x'(t)}{\sqrt{x'(t)^2 + y'(t)^2}},$$
$$\frac{1}{\sec^2\Phi} = \cos^2\Phi = \frac{x'(t)^2}{x'(t)^2 + y'(t)^2}.$$

Chega-se às igualdades:

$$\begin{aligned}\frac{d\Phi}{dt} &= \frac{1}{\sec^2\Phi}\frac{d}{dt}(\tan\Phi),\\ &= \frac{x'(t)^2}{x'(t)^2+y'(t)^2}\frac{x'(t)y''(t)-y'(t)x''(t)}{x'(t)^2},\\ &= \frac{x'(t)y''(t)-y'(t)x''(t)}{x'(t)^2+y'(t)^2}.\end{aligned}$$

Finalmente:

$$\begin{aligned}k(t) &= \frac{d\Phi/dt}{ds/dt},\\ &= \frac{x'(t)y''(t)-y'(t)x''(t)}{x'(t)^2+y'(t)^2}\frac{1}{\sqrt{x'(t)^2+y'(t)^2}},\\ k(t) &\doteq \frac{x'(t)y''(t)-y'(t)x''(t)}{\left(\sqrt{x'(t)^2+y'(t)^2}\right)^3}.\end{aligned}$$

Donde se conclui a demonstração da proposição.

2.3.4.2 Curvas Interpolantes

Para aplicar a fórmula da curvatura é preciso conhecer a curva paramétrica que interpola a curva discreta C. Encontrar curvas paramétricas que interpolam uma grande quantidade de pontos, embora não seja uma tarefa difícil, pode trazer um resultado indesejado como curvas com intensas oscilações. Por isso é mais prático interpolar por partes, dois a dois (retas) ou três a três pontos (curvas de grau 2), dessa forma minimizando-se efeitos indesejados.

Um modelo de curva simples é constituído pelas polinomiais da forma:

$$q(t) = \sum_{i=0}^{n} a_i t^i.$$

Para interpolar três pontos com este modelo de curva, polinomiais quadráticas são suficientes. Considere dados $\boldsymbol{p}_{i-1} = (x_{i-1}, y_{i-1})$, $\boldsymbol{p}_i = (x_i, y_i)$ e $\boldsymbol{p}_{i+1} = (x_{i+1}, y_{i+1})$ para encontrar uma curva na forma

$$\begin{aligned}x(t) &= a_2t^2 + a_1t + a_0,\\ y(t) &= b_2t^2 + b_1t + b_0,\end{aligned}$$

Com estes, encontrar uma curva torna-se solucionar os sistemas:

$$\begin{bmatrix} t_{i-1}^0 & t_{i-1}^1 & t_{i-1}^2 \\ t_i^0 & t_i^1 & t_i^2 \\ t_{i+1}^0 & t_{i+1}^1 & t_{i+1}^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_{i-1} \\ x_i \\ x_{i+1} \end{bmatrix}, \qquad \begin{bmatrix} t_{i-1}^0 & t_{i-1}^1 & t_{i-1}^2 \\ t_i^0 & t_i^1 & t_i^2 \\ t_{i+1}^0 & t_{i+1}^1 & t_{i+1}^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_{i-1} \\ y_i \\ y_{i+1} \end{bmatrix},$$

onde a escolha dos t's deve seguir $t_{i-1} < t_i < t_{i+1}$ para que a curva passe nos pontos na ordem correta. As quantidades $|t_i - t_{i-1}|$ e $|t_{i+1} - t_i|$ estão relacionadas ao comprimento de curva entre os pontos $\boldsymbol{p}_{i-1}$ e $\boldsymbol{p}_i$, e $\boldsymbol{p}_i$ e $\boldsymbol{p}_{i+1}$. Por isso uma escolha adequada de t deve levar em conta a distância entre os pontos. Escolhe-se t de forma que simplifique o sistema e seja coerente com o comprimento de curva:

$$\begin{aligned} t_{i-1} &= -|\boldsymbol{p}_i - \boldsymbol{p}_{i-1}| = -\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \\ t_i &= 0, \\ t_{i+1} &= |\boldsymbol{p}_{i+1} - \boldsymbol{p}_i| = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}. \end{aligned}$$

Sejam

$$T = \begin{bmatrix} 1 & t_{i-1} & t_{i-1}^2 \\ 1 & 0 & 0 \\ 1 & t_{i+1} & t_{i+1}^2 \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}, \qquad \boldsymbol{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}, \boldsymbol{x} = \begin{bmatrix} x_{i-1} \\ x_i \\ x_{i+1} \end{bmatrix}, \qquad \boldsymbol{y} = \begin{bmatrix} y_{i-1} \\ y_i \\ y_{i+1} \end{bmatrix}.$$

Como a segunda linha de T é $[1 \quad 0 \quad 0]$, é fácil encontrar a_0 e b_0:

$$\begin{aligned} a_0 + 0a_1 + 0a_2 = x_i, \qquad &\rightarrow \qquad a_0 = x_i, \\ b_0 + 0b_1 + 0b_2 = y_i, \qquad &\rightarrow \qquad b_0 = y_i. \end{aligned}$$

Conhecidos a_0 e b_0, pode-se reduzir os sistemas:

$$\begin{bmatrix} t_{i-1} & t_{i-1}^2 \\ t_{i+1} & t_{i+1}^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} x_{i-1} - x_i \\ x_{i+1} - x_i \end{bmatrix} \qquad \begin{bmatrix} t_{i-1} & t_{i-1}^2 \\ t_{i+1} & t_{i+1}^2 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} y_{i-1} - y_i \\ y_{i+1} - y_i \end{bmatrix}$$

ou, simplificadamente,

$$\boldsymbol{a}' = T'^{-1}\boldsymbol{x}', \qquad \boldsymbol{b}' = T'^{-1}\boldsymbol{y}',$$

com

$$T' = \begin{bmatrix} t_{i-1} & t_{i-1}^2 \\ t_{i+1} & t_{i+1}^2 \end{bmatrix}, \quad \text{e} \qquad T'^{-1} = \frac{1}{t_{i-1}t_{i+1}^2 - t_{i-1}^2 t_{i+1}} \begin{bmatrix} t_{i+1}^2 & -t_{i-1}^2 \\ -t_{i+1} & t_{i-1} \end{bmatrix}.$$

Assim, explicitamente,

$$a_1 = \frac{1}{t_{i-1}t_{i+1}^2 - t_{i-1}^2 t_{i+1}} \left((x_{i-1} - x_i)t_{i+1}^2 - (x_{i+1} - x_i)t_{i-1}^2 \right)$$
$$a_2 = \frac{1}{t_{i-1}t_{i+1}^2 - t_{i-1}^2 t_{i+1}} \left(-(x_{i-1} - x_i)t_{i+1} + (x_{i+1} - x_i)t_{i-1} \right)$$
$$b_1 = \frac{1}{t_{i-1}t_{i+1}^2 - t_{i-1}^2 t_{i+1}} \left((y_{i-1} - y_i)t_{i+1}^2 - (y_{i+1} - y_i)t_{i-1}^2 \right)$$
$$b_2 = \frac{1}{t_{i-1}t_{i+1}^2 - t_{i-1}^2 t_{i+1}} \left(-(y_{i-1} - y_i)t_{i+1} + (y_{i+1} - y_i)t_{i-1} \right)$$

2.3.4.3 Cálculo das características **abs_k** e **wigg**

Para o cálculo da curvatura, como se conhece a curva paramétrica e suas derivadas,

$$x'(t) = 2a_2 t + a_1, \qquad y'(t) = 2b_2 t + b_1,$$
$$x''(t) = 2a_2, \qquad y''(t) = 2b_2.$$

no ponto $t_i = 0$ a curvatura é:

$$k_i = 2\frac{a_1 b_2 - b_1 a_2}{\left(\sqrt{a_1^2 + b_1^2}\right)^3}.$$

Assim, calculam-se **abs_k** e **wigg**:

$$\mathbf{abs\_k} = \frac{1}{n}\sum_{i=0}^{n-1} |k_i|$$

$$\mathbf{wigg} = \frac{1}{n}\sum_{i=0}^{n-2} \text{wiggliness}(i)$$

onde

$$\text{wiggliness}(i) = \begin{cases} i, & k_i k_{i+1} < 0 \quad e \quad |k_i k_{i+1}| > \epsilon \\ 0, & \text{caso contrário} \end{cases}$$

2.3.5 Cálculo das características len, edge_sparsity e avg_conf

O comprimento de cada curva discreta, sequência de n pontos (x_i, y_i), é a soma das distâncias ponto a ponto,

$$\textbf{len} = \sum_{i=0}^{n-2} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}.$$

Edge Sparsity é a medida da quantidade de *edgels* que não pertencem à curva no entorno desta. Esta característica utiliza informação espacial do mapa de bordas, sendo calculada como descrito pelo algoritmo 1.

Algoritmo 1 - Edgel Sparsity

DOCUMENTAÇÃO
  ENTRADAS
    $E_{i,j}$: Mapa de bordas de tamanho $n \times m$
    C_i: Curva, sequência de pontos $C(i) = (x_i, y_i)$ de tamanho k
    len: Comprimento da curva C_i
    r: Raio do cálculo
  SAÍDAS
    $edgeSparsity$: Edge Sparsity entorno da curva
FIM DOCUMENTAÇÃO
ALGORITMO CÁLCULO *Edge Sparsity*
1. **declarar** $edgeSparsity$ **numérico**
2. **declarar** $M_{i,j}$ **Matriz de tamanho** $n \times m$
3. $edgeSparsity \leftarrow 0$ {*Inicializa com zero*}
4. $M \leftarrow 0$ {*Inicializa com zero*}
5. **para** $(x_i, y_i) \leftarrow C(i)$ **de** $i = 1$ **até** $i = k$, **fazer** {*Para cada ponto da curva*}
6. **para** $l \neq 0, m \neq 0$ **de** $l = m = -r$ **até** $l = m = r$, **fazer**
7. $M(x_i + l, y_i + m) \leftarrow true$
8. **fim para**
9. **fim para**
10. **para de** $i = j = 1$ **até** $i = n, j = m$, **fazer**
11. **se** $(E(i,j) \neq vazio \;\&\&\; M(i,j) == true)$, **então**
12. $edgeSparsity \leftarrow edgeSparsity + 1$
13. **fim se**
14. **fim para**
15. $edgeSparsity \leftarrow edgeSparsity/len$

A intensidade média da curva **avg_conf** é calculada da seguinte forma:

$$\textbf{avg\_conf} = \frac{1}{n} \sum_{i=0}^{n-1} \Delta_i,$$

com Δ_i a intensidade de cada *edgel* e_i pertencente à curva.

2.3.6 Cálculo das características geom_diff e tex_diff

Sejam duas curvas discretas C_1 com n *edgels* e C_2 com m *edgels* com $\boldsymbol{e}_n^1 \in C_1$, $\boldsymbol{e}_m^2 \in C_2$ e $\boldsymbol{e}_n^1 = \boldsymbol{e}_m^2$, onde o subscrito indica o índice do *edgel* e o sobrescrito a curva correspondente. Então o cosseno do ângulo entre estas curvas em $\boldsymbol{e}_n^j$, se $\boldsymbol{p}_i^j$ é o ponto 2D referente ao *edgel* $\boldsymbol{e}_i^j$ e r é o parâmetro livre que define o raio local de cálculo, é calculado por

$$\textbf{geom\_diff} = \frac{(\boldsymbol{p}_n^1 - \boldsymbol{p}_{n-r}^1) \cdot (\boldsymbol{p}_m^2 - \boldsymbol{p}_{m-r}^2)}{|\boldsymbol{p}_n^1 - \boldsymbol{p}_{n-r}^1| \quad |\boldsymbol{p}_m^2 - \boldsymbol{p}_{m-r}^2|}.$$

Para calcular a diferença de textura entre duas curvas são calculados os histogramas da textura localmente à direita e à esquerda de cada curva e então faz-se a distância χ^2 entre os histogramas.

Dada T uma matriz onde cada elemento $T_{i,j}$ é um número inteiro que representa a classe da textura em (i, j), T é chamada mapa de textura relativo a uma imagem I.

Algoritmo 2 - Histograma de Textura

ENTRADAS

$T_{i,j}$: Mapa de textura $n \times m$
C_i: Curva, sequência de pontos $C(i) = (x_i, y_i)$ de tamanho k
r: raio local
tc: Número total de classes

SAÍDAS

U: Histograma à direita da curva
L: Histograma à esquerda da curva

ALGORITMO CÁLCULO HISTOGRAMAS DE TEXTURA

1. **declarar** $U[tc]$ **vetor numérico de tamanho** tc
2. **declarar** $L[tc]$ **vetor numérico de tamanho** tc
3. $U \leftarrow 0$ {*Inicializa com zeros*}
4. $L \leftarrow 0$ {*Inicializa com zeros*}
5. **para** $(x_i, y_i) \leftarrow C(i)$ **de** $i = 1$ **até** $i = k$, **fazer** {*Para cada ponto da curva*}
6. **declarar** xl **índice** x **à esquerda**
7. **declarar** xr **índice** x **à direita**
8. **declarar** yl **índice** y **à esquerda**
9. **declarar** yr **índice** y **à direita**
10. **para** j **de** $j = 1$ **até** $j = r$, **fazer** {*Para o raio local*}
11. $xl \leftarrow x_i - j$
12. $xr \leftarrow x_i + j$
13. $yl \leftarrow y_i - j$
14. $yl \leftarrow x_i + j$
15. $U(T(xr, yr)) \leftarrow U(T(xr, yr)) + 1$
16. $L(T(xl, yl)) \leftarrow L(T(xl, yl)) + 1$
17. **fim para**
18. **fim para**

Sejam U^1, L^1 os histogramas da curva C_1 e U^2, L^2 os da curva C_2, com U o histograma da textura acima da curva e L o histograma da textura abaixo da curva, calculados pelo algoritmo 2, então

$$\textbf{tex\_diff} = \frac{1}{2}\left(\chi^2(U^1, U^2) + \chi^2(L^1, L^2)\right)$$

com

$$\chi^2(X, Y) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}$$

3 AQUISIÇÃO DE DADOS

Neste capítulo são apresentados os dados utilizados no capítulo 4, de experimentos, e as interfaces gráficas utilizadas para pré-processamento dos mesmos. Os dados, em seu estado bruto, são quatro vídeos do tanque de ondas multidirecionais da COPPE/UFRJ, que são processados em várias etapas. Este capítulo é dividido em seções tratando do tanque de ondas e dos vídeos, seleção do subconjunto utilizado e a geração das referências utilizadas no treino e teste dos resultados.

3.1 Tanque de ondas e aquisição dos vídeos

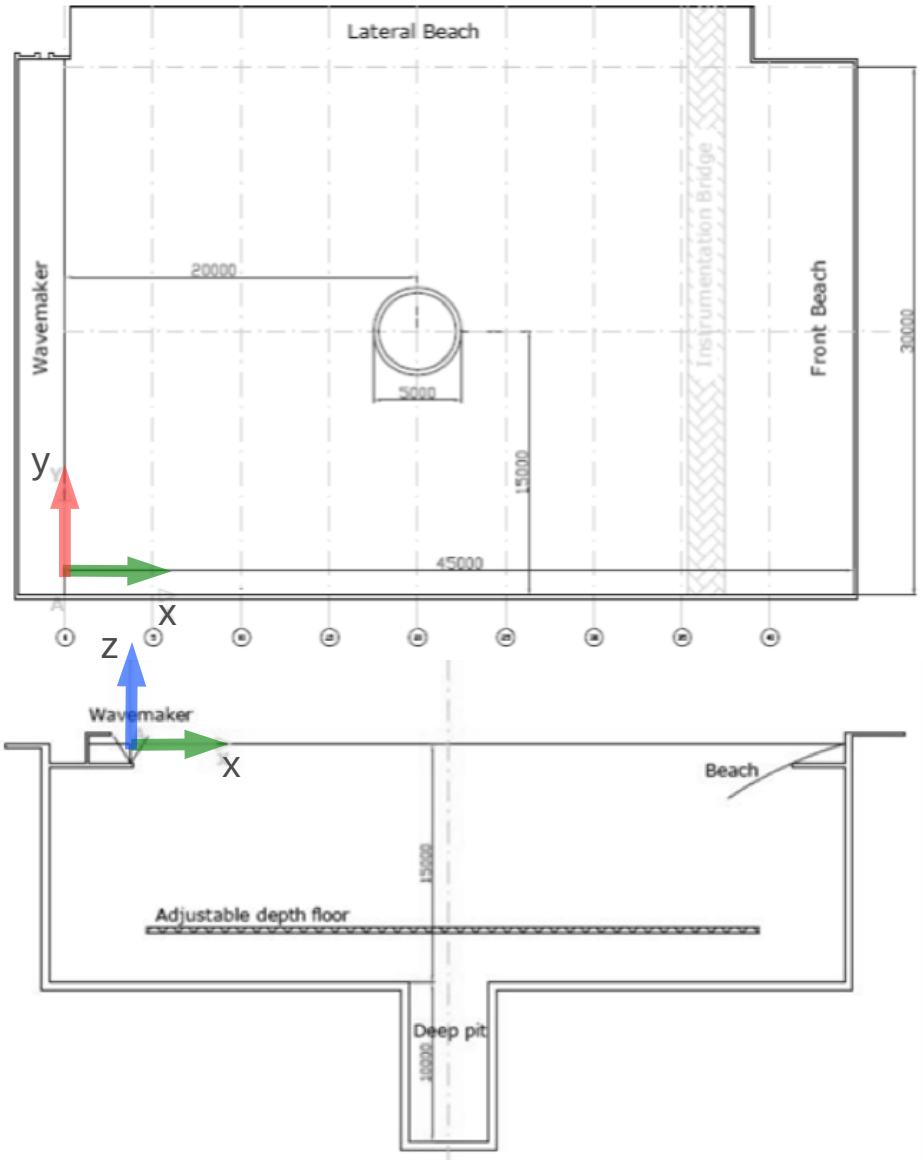
Os vídeos brutos foram produzidos no *LabOceano* que faz parte do Programa de Engenharia Naval e Oceânica da COPPE/UFRJ. Com mais de 2000 projetos já realizados em convênio com a Petrobras, a COPPE/UFRJ é referência internacional em pesquisa e desenvolvimento. O *LabOceano* possui uma equipe técnica altamente qualificada, formada por professores, pesquisadores e engenheiros em regime de dedicação exclusiva, alunos de mestrado e doutorado, além de profissionais e pessoal de apoio. Com equipamentos de última geração e uma infra-estrutura somente encontrada em poucos laboratórios no mundo, o *LabOceano* realiza ensaios em escala reduzida em hidrodinâmica experimental, hidrodinâmica computacional e em modelagem numérica de sistemas navais e oceânicos, além do desenvolvimento de projetos de pesquisa e de treinamento de pessoal.

O tanque de ondas multidirecional possui as dimensões mostradas na Figura 25, tendo $45m$ de comprimento, $30m$ de largura e $15m$ de profundidade, com um poço central com $10m$ adicionais localizado a $20m$ de distância do gerador de ondas. É dotado ainda de duas praias absorventes, uma localizada no final de seu comprimento e outra em um dos lados.

Os quatro vídeos provêm de quatro câmeras sincronizadas, formando duas cabeças estéreo espaçadas por 45º entre si, Figura 26 . As cabeças estéreo estão dispostas verticalmente. Além das câmeras, existem sondas que medem as ondas em um único ponto, as quais podem ser usadas para fins de validação.

Cada vídeo tem duração de 179 segundos ou 2min e 59s, resolução de 1920x1080 (fullHD) e bitrate de vídeo de aproximadamente 12000 kbps (kilobit por segundo) totalizando aproximadamente 265 MByte cada vídeo. Foram extraídos 182 quadros ordenados e igualmente espaçados no tempo de cada vídeo, definindo assim um subconjunto representativo de todos os possíveis quadros.

Figura 25 - Dimensões do tanque de ondas *LabOceano* COPPE/UFRJ



Legenda: Dimensões do tanque de ondas *LabOceano* COPPE/UFRJ
Fonte: *LabOceano* COPPE/UFRJ, 2017.

Tabela 2 - Posições das câmeras.

Câmera	ângulo(°)	$x(mm)$	$y(mm)$	$z(mm)$
1	0	14710	7500	480
2	0	14710	7500	240
3	45	14975	15000	480
4	45	14975	15000	240

Fonte: O autor, 2017.

Figura 26 - Quadros representativos extraídos de cada video.

Legenda: Quadros representativos extraídos de cada video.
Fonte: O autor, 2017.

Tabela 3 - Tamanhos dos cortes realizados e posições do canto superior esquerdo de onde se iniciam os cortes em cada vídeo.

Câmera	Comprimento	Largura	x	y
1	1340	230	575	850
2	1350	224	570	855
3	1346	240	510	450
4	1344	242	576	744

Fonte: O autor, 2017.

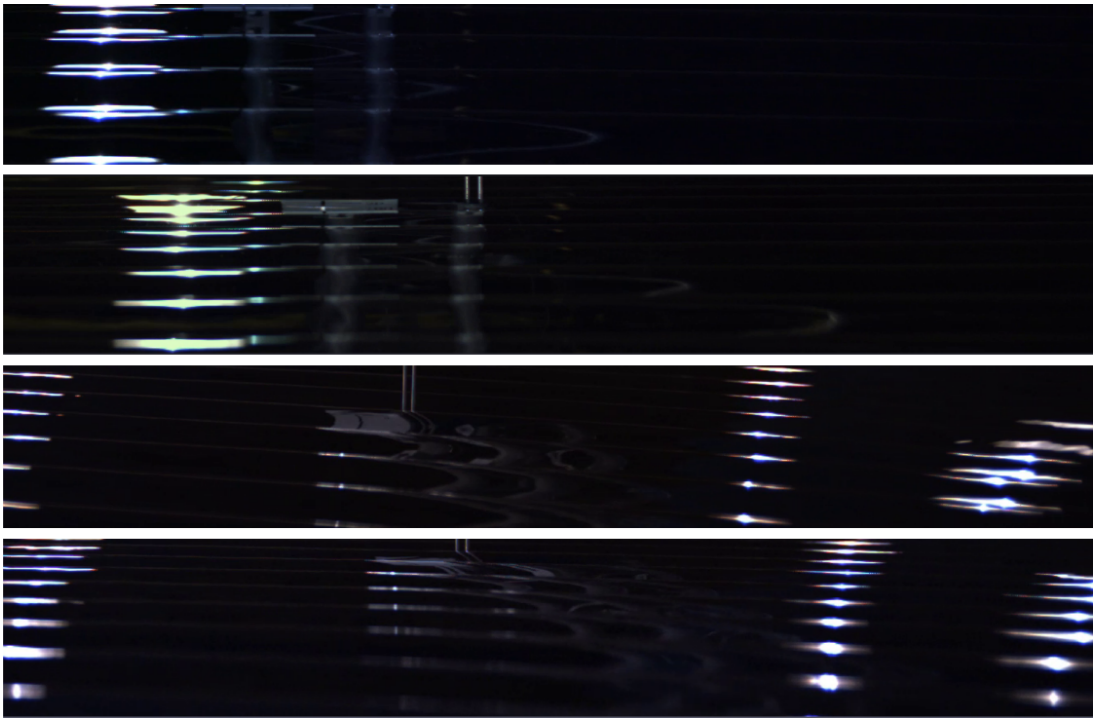
3.2 **Área de interesse**

Verifica-se que em geral, como exemplificado na Figura 26, as ondas estão presentes apenas na parte inferior de cada quadro. Assim foi feito um corte de cada quadro contendo apenas ondas. O corte foi definido com tamanho 1340×230 píxels como na Figura 27. A escolha por diferentes locais em cada vídeo busca manter pontos em comum entre eles, dado que estes foram tomados de diferentes posições. Este é um requisito para a reconstrução 3D e que não impõe perda de generalidade nos dados. Dessa forma tem-se uma área de interesse bem definida e representativa do conjunto, podendo ser estendida para outras partes de cada vídeo e para os quadros intermediários.

Figura 27 - Cortes da área de interesse

Legenda: Em laranja, área de corte em cada vídeo
Fonte: O autor, 2017.

Figura 28 - Imagens das áreas selecionadas.



Legenda: Imagens das áreas selecionadas.
Fonte: O autor, 2017.

3.3 Dados de referência

Os dados de referência, que constituem parte fundamental do sistema, são utilizados tanto no aprendizado como na comparação de resultados obtidos. Desta forma a engenharia destes influencia a qualidade dos resultados. Estas referências estão em um formato compatível com o código implementando, mas podem ser lidas facilmente com qualquer editor de texto e podem ser convertidas para outros formatos facilmente, podendo assim estas podem ser utilizadas em outros experimentos, no desenvolvimento de outras técnicas e em comparações diretas com os resultados obtidos nesta dissertação. A disponibilização dessas será feita posteriormente, possibilitando também a validação externa dos resultados, o que é fundamental na concretização de novas técnicas e metodologias.

Essas referências serão utilizados para treino (aprendizado) e teste dos resultados obtidos posteriormente. Tais dados consistem de curvas anotadas à mão para cada quadro utilizado, seguindo os critérios de totalidade, distinção e objetividade conforme apresentados na seção 1.1, página 28. Estes dados serão utilizados pelos métodos de aprendizagem de máquina propostos na seção 2.3.

A aquisição de referências totalmente anotadas à mão requer um imenso trabalho, necessitando do desenho de curvas feitas manualmente por várias pessoas e utilizando as curvas mais anotadas. Para reduzir o trabalho necessário, nesta aquisição foram seguidas as seguintes etapas na construção das referências:

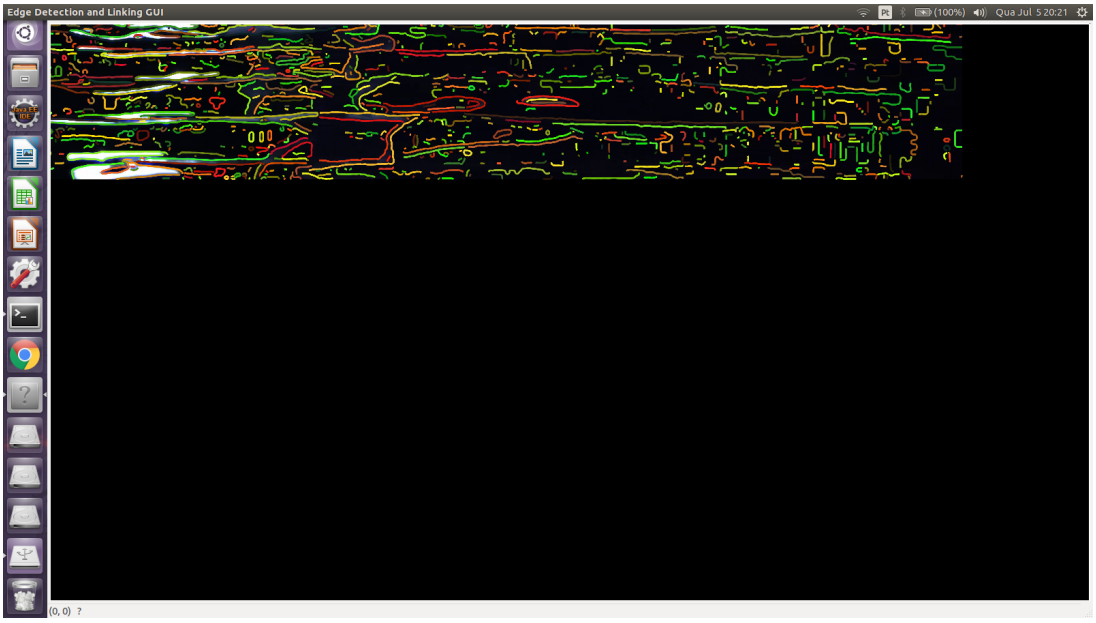
- aquisição automática de possíveis curvas;
- separação manual de curvas pertencentes a contornos distintos;
- junção manual de curvas pertencentes a um mesmo contorno;
- remoção manual de curvas não verídicas.

A aquisição automática de possíveis curvas é feita usando as técnicas apresentadas nas seções 2.1 e 2.2 com limiares muito baixos, ou seja, captando grande parte das curvas verídicas, porém também muitas outras não verídicas com baixa qualidade de totalidade, distinção e objetividade.

Para a parte manual, foi construída uma interface gráfica para visualização e edição de curvas, Figura 29, permitido quebrar curvas discretas em qualquer de seus pontos, Figura 30, unir curvas com extremidades suficientemente próximas, Figura 31, e remover curvas espúrias, Figura 32. Cada etapa gera referências próprias, ou seja, há referências apenas para quebra, outras para junções e finalmente para a remoção. Dessa forma é possível analisar cada etapa separadamente.

Como as imagens de onde as curvas são extraídas apresentam muitas singularidades, reflexões e transparência, as curvas editadas procuram manter como verídico tudo

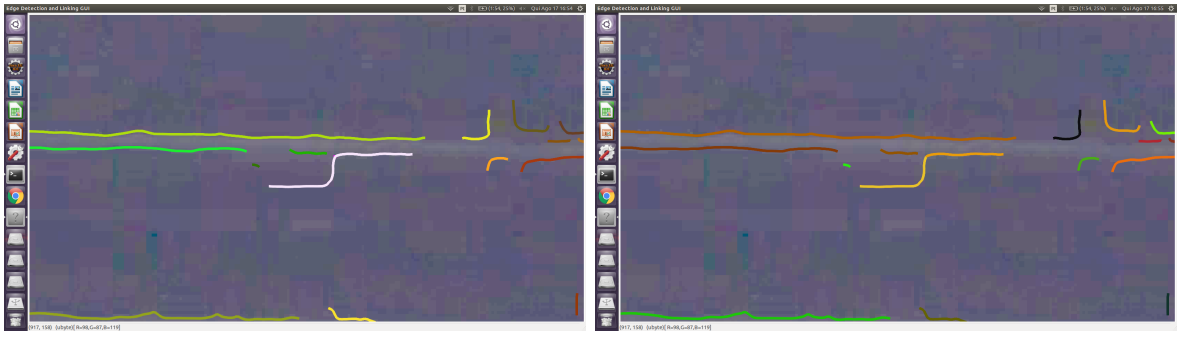
Figura 29 - Interface gráfica para visualização e edição de curvas.



Legenda: Interface gráfica para visualização e edição de curvas.

Fonte: O autor, 2017.

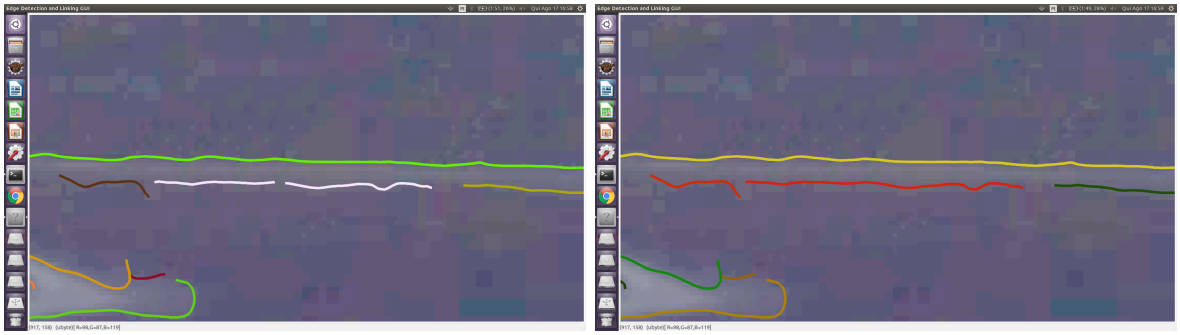
Figura 30 - Quebra de uma curva na interface gráfica.



Legenda: Quebra de uma curva na interface gráfica.
[Esquerda] Curva selecionada em lilás, ponto selecionado em rosa.
[Direita] Curva dividida em duas.

Fonte: O autor, 2017.

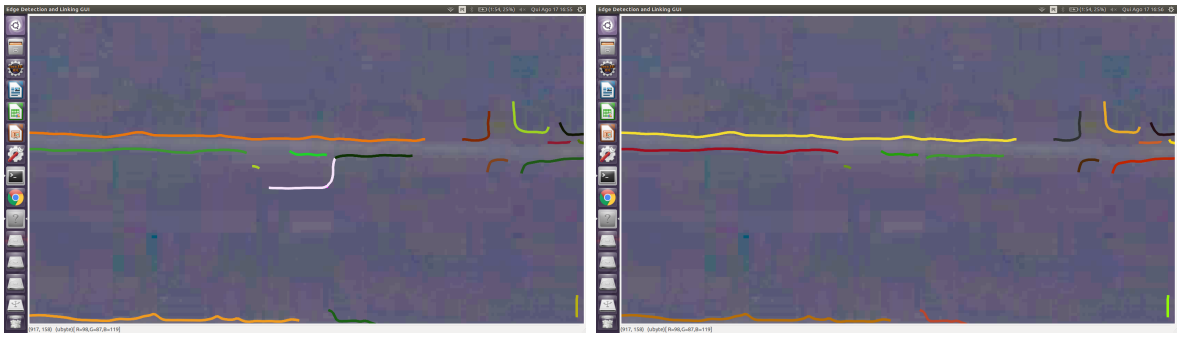
Figura 31 - Junção de duas curvas na interface gráfica.



Legenda: Junção de duas curvas na interface gráfica.
[Esquerda] Curvas selecionadas em lilás. [Direita] União das duas curvas.

Fonte: O autor, 2017.

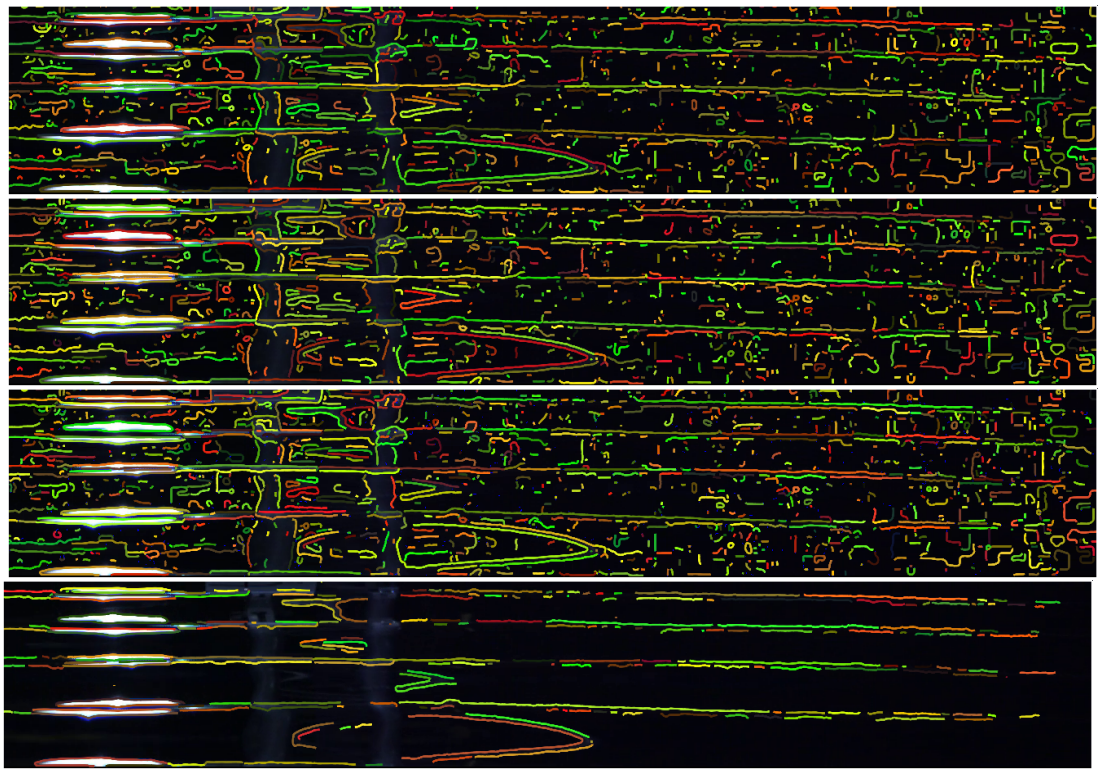
Figura 32 - Remoção de uma curva na interface gráfica.



Legenda: Remoção de uma curva na interface gráfica.
[Esquerda] Curva selecionada em lilás.
[Direita] Curva removida.

Fonte: O autor, 2017.

Figura 33 - Curvas extraídas automaticamente do quadro 140 da câmera 1 e referências.



Legenda: De cima para baixo, curvas extraídas automaticamente do quadro 140 da câmera 1, e as referências manuais de quebra, referência de união e referência de remoção.

Fonte: O autor, 2017.

que está diretamente ligado à geometria das ondas, como cristas e ondulações, e remover curvas geradas por ruído e outros efeitos ópticos indesejados. O conjunto de referências é composto por 18358 fragmentos de curva, utilizadas no aprendizado de seleção, e 1566 junções, utilizadas no aprendizado de quebra e junção, ambas foram extraídas de 12 quadros, sendo 3 quadros de cada vídeo.

Na Figura 33 pode ser vista a mudança gradual das curvas obtidas automaticamente até o último estágio da referência. Embora haja alguns fragmentos distintos que façam parte de um mesmo contorno estes só podem ser unidos se suas extremidades são próximas o suficiente, uma limitação imposta pela obtenção automática, mas que pode ser removida adicionando um modelo de curva para completar estas lacunas coerentemente com *edgels* contendo posição e orientação.