Features de uma imagem

Augusto de Holanda B. M. Tavares

Universidade Federal da Paraíba
Centro de Informática
Departamento de Sistemas de Computação

28 de agosto de 2024

O que são *features*?

- **Features** ou **características** de uma imagem são pontos ou regiões importantes para a caracterização dos elementos contidos nesta.

Exemplo

- Considere o problema de montar um quebra-cabeça. Como você sabe quais partes conectam com quais outras?

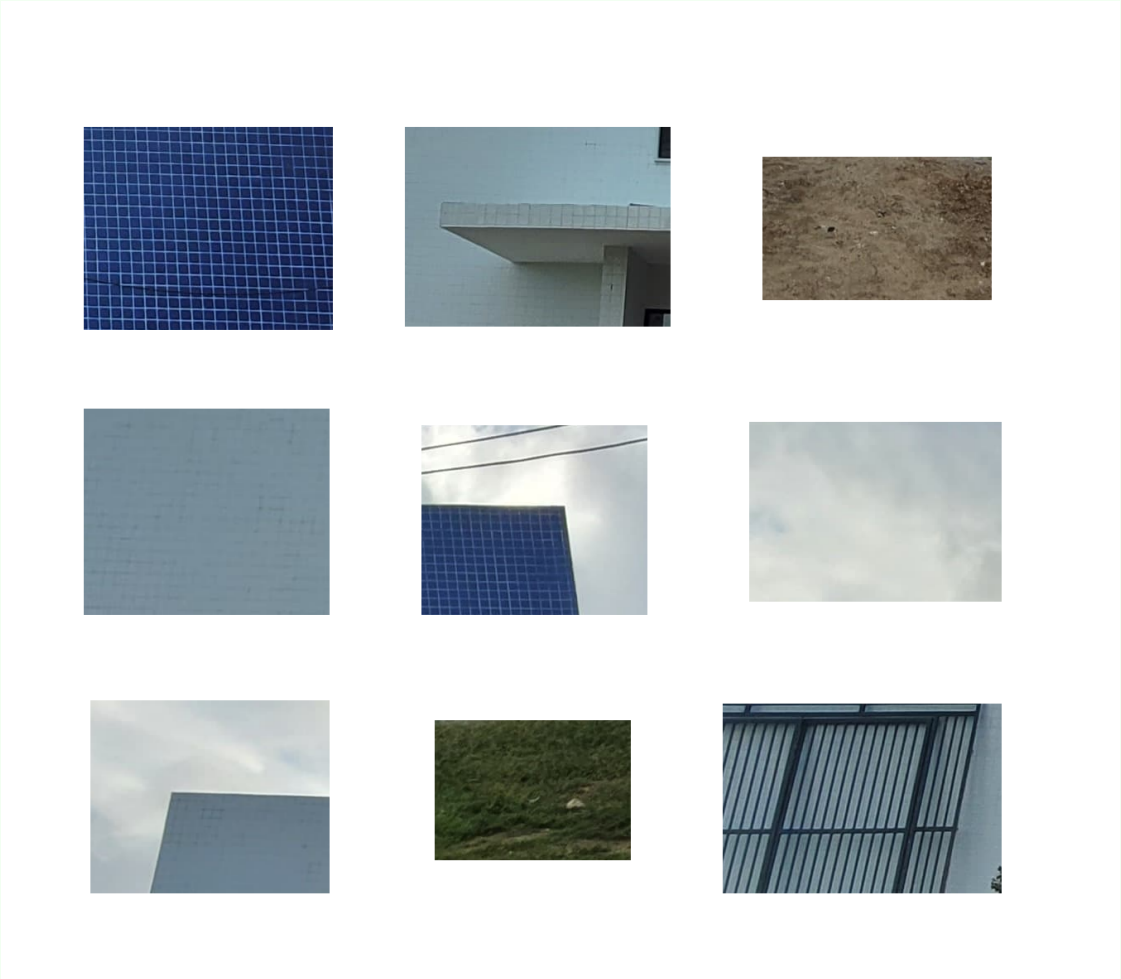
- A idéia é que ao se determinar elementos importantes na imagem, é possível:
  - Procurar estes elementos em diferentes imagens.
  - Verificar se um ou mais destes elementos mudou de posição ou foi rotacionado em uma sequência de quadros.

Características de uma fachada

Exemplo

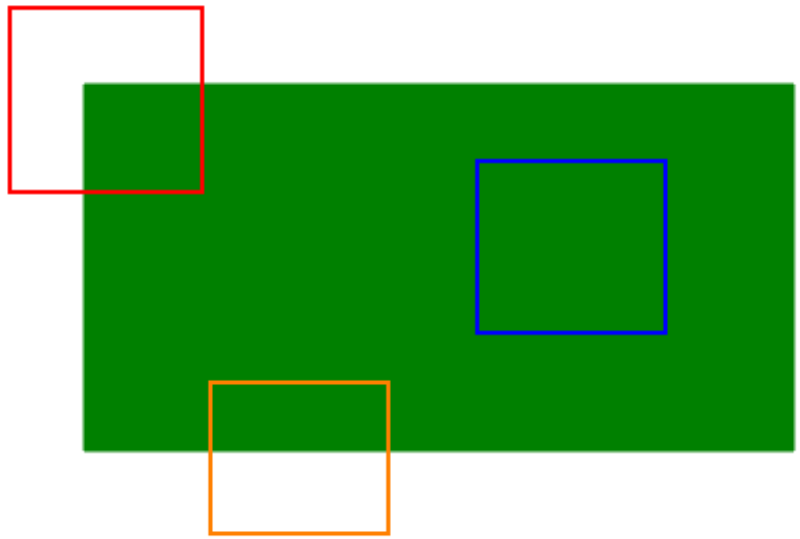
Características de uma fachada

Exemplo



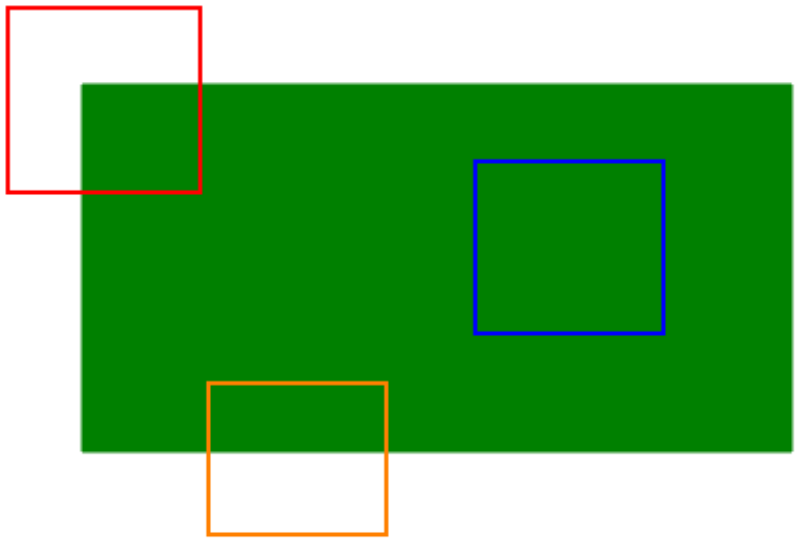
Características de uma imagem

- Considere a imagem abaixo:



- Área azul: Área contínua, sem variações.
- Área azul: Borda.
- Área azul: Canto.

Características de uma imagem



- Área azul: Não há mudanças com movimento nesta área.
- Área azul: Mudanças para movimento vertical, sem mudanças para movimento horizontal.
- Área azul: Mudanças para qualquer movimento.

Características de uma imagem

- Via de regra, os **cantos** são as características/*features* mais determinantes de uma imagem.
- Outra característica comum são as **manchas**/**blobs**.
- Tendo isto em mente, como se localiza uma *feature* em uma imagem?

Exemplo

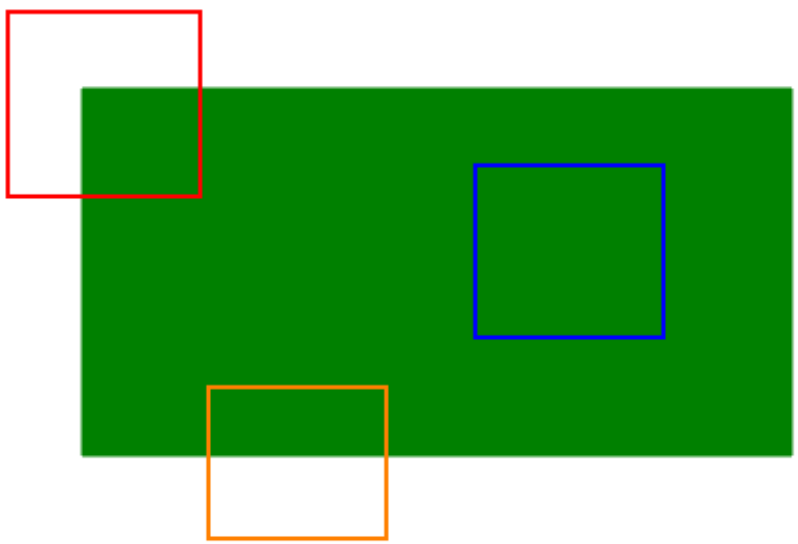
- A região em torno do canto do prédio tem céu na parte de cima, abaixo as pastilhas do prédio, sem a presença de janelas. Há alguns fios acima do do canto. As pastilhas nesta região são azuis.

Características de uma imagem

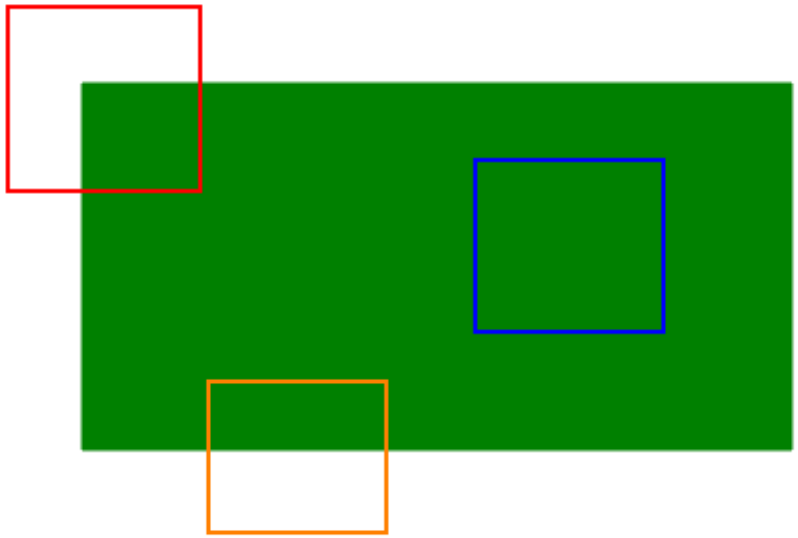
- Este processo é a **descrição de features**. Isto gera um conjunto de **descritores**.
- Uma *feature*, então, pode ser caracterizada pelo seu conjunto de descritores.
- Para determinar se uma *feature* qualquer está presente em uma imagem basta procurar os seus descritores. Caso todos (ou a maior parte deles) estejam na imagem, a *feature* provavelmente estará.

Características de uma imagem

- No entanto, até este ponto foi negligenciada uma pergunta fundamental: como se determina que uma parte da imagem é uma *feature*?
- Para tanto, retornemos ao exemplo do caso abstrato.



Características de uma imagem



- Área azul: Movimentos pequenos desta área resultam em variações desprezíveis.
- Área azul: Movimentos pequenos desta área resultam em variações razoáveis.
- Área azul: Movimentos pequenos desta área resultam em variações próximas do máximo.

Detector de cantos de Harris

- Este conceito de detectar a máxima variação com o deslocamento é representado no artigo **"A Combined Corner and Edge Detector (1988)"**:

$$E(u,v) = \sum_{x,y} w(x,y)\,[I(x+u, y+v) - I(x,y)]^2$$

- $E(u,v)$ mensura o deslocamento (u,v) em todas as direções. $I(x,y)$ é a intensidade no ponto desejado, e $w(x,y)$ é uma função de janelamento, tipicamente retangular ou gaussiana.

Detector de cantos de Harris

- Se o objetivo é maximizar $E(u, v)$ a diferença do segundo termo deve ser maximizada. Aplicando uma expansão de Taylor, chega-se a uma forma simplificada da equação anterior:

$$E(u, v) \approx \begin{bmatrix} u & v \end{bmatrix} \boldsymbol{M} \begin{bmatrix} u \\ v \end{bmatrix}$$

- Onde:

$$\boldsymbol{M} = \sum_{x,y} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

- Sendo I_x e I_y as derivadas de $I(x, y)$ em x e y, respectivamente. Elas podem ser obtidas com um filtro de Sobel.

Detector de cantos de Harris

- Sejam, então, λ_1 e λ_2 os autovalores da matriz $\boldsymbol{M}$.
- É definida uma pontuação R tal que:

$$R = \det(\boldsymbol{M}) - k\ \mathrm{trace}(\boldsymbol{M})^2$$

- O determinante de uma matriz é o produto dos autovalores e o traço é a sua soma.
- Ou seja:

$$R = \lambda_1\lambda_2 - k\,(\lambda_1 + \lambda_2)^2$$

Detector de cantos de Harris

- De onde se tem que:
  - Se $|R|$ é pequeno, λ_1 e λ_2 são pequenos, indicando que a região da imagem é plana.
  - Se $R < 0$, ou $\lambda_1 << \lambda_2$ ou $\lambda_2 << \lambda_1$, indicando uma borda na imagem.
  - Quando $R >> 1$, indicando que $\lambda_1 >> 1$, $\lambda_2 >> 1$ e $\lambda_1 \approx \lambda_2$, a região é um canto.

Detector de Shi-Tomasi

- A função de pontuação do detector de Harris foi modificada por Shi-Tomasi para:

$$R = \min(\lambda_1, \lambda_2)$$

- Se R for maior do que um determinado limiar ϵ, o ponto em questão é um canto.

Detector de Shi-Tomasi

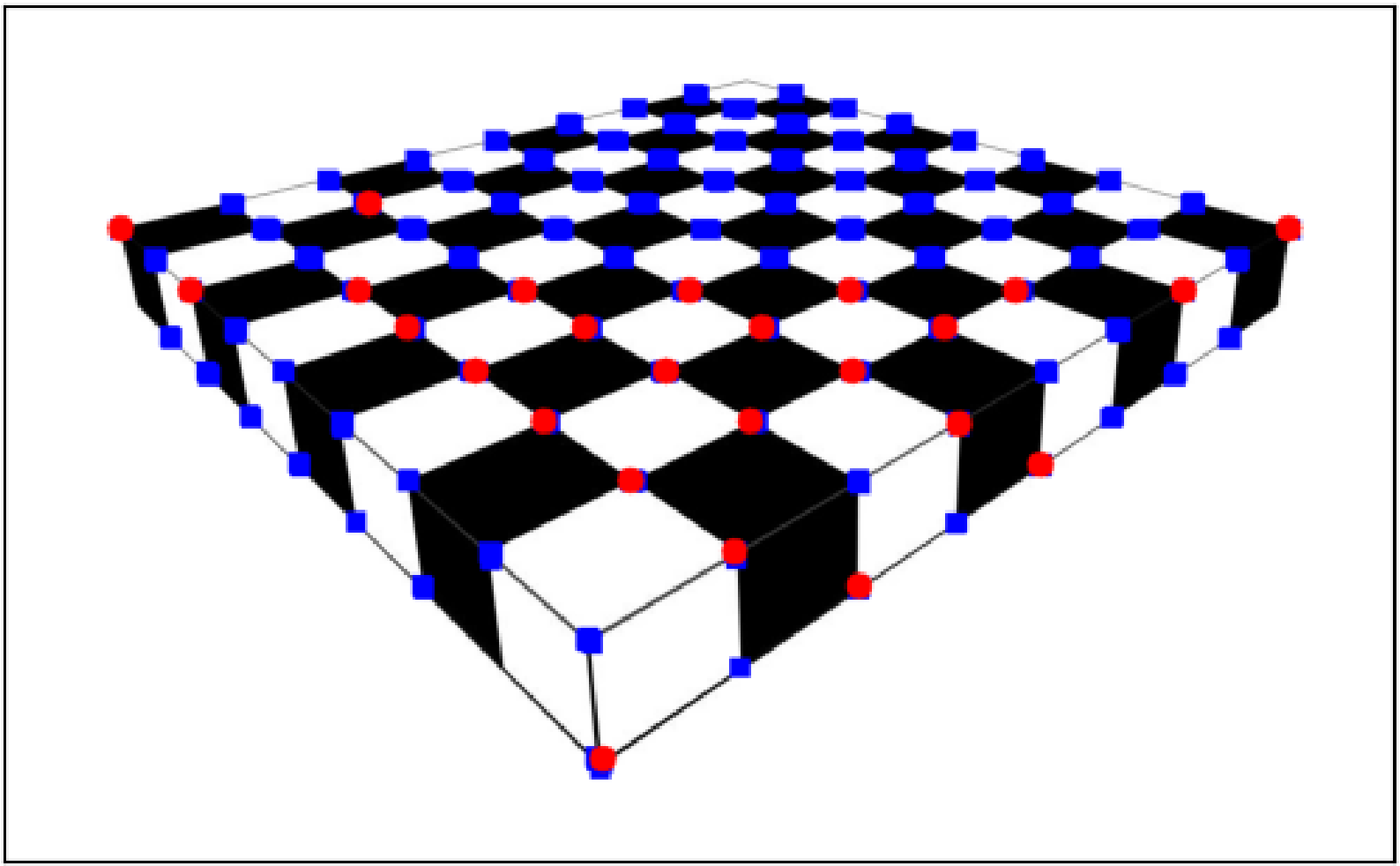
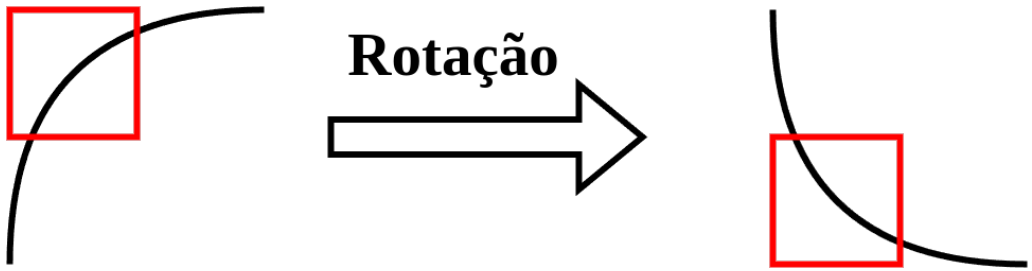
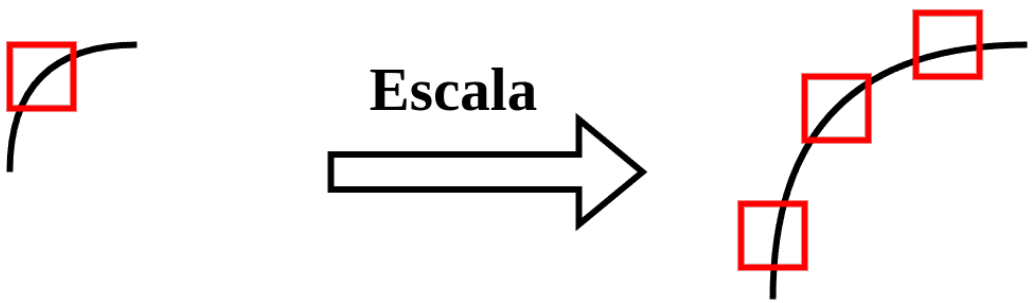


Figura: *Features* detectadas por Harris em azul e por Shi-tomasi em vermelho (segundo caso limitado às melhores).

Variações nas *features*

- Os detectores de Harris e de Shi-tomasi são resistentes à rotação da imagem, mas perdem as suas características caso ocorram transformações de escala.





Scale-Invariant Feature Transform (SIFT)

- Em 2004 é publicado o artigo "*Distinctive Image Features from Scale-Invariant Keypoints*", de D. Lowe, que é capaz de detectar as características da imagem de maneira resistente à rotação e à mudanças de escala.
- O primeiro passo é calcular a diferença entre versões da mesma imagem filtradas por gaussianas para diferentes valores de σ. O σ das curvas gaussianas neste caso tem relação com a escala da imagem.
- Este processo localiza manchas nas imagens. Então, os valores de máximo são selecionados, indicando que naquele ponto (x, y) em determinada escala σ há um ponto-chave em potencial.

Scale-Invariant Feature Transform (SIFT)

- Até aqui há dois problemas: algumas das *features* detectadas são falsos positivos, e as bordas também são destacadas pela diferença de gaussianos.
- Um método de exclusão por autovalores da matriz Hessiana similar ao método de Harris é utilizado para excluir as bordas.
- Dentre os pontos restantes são ignorados aqueles cujo valor no processo fica abaixo de um determinado limiar.

Scale-Invariant Feature Transform (SIFT)

- O gradiente da vizinhança dos pontos restantes é calculado, de modo a definir a orientação daquela *feature*.
- São criados os descritores da *feature*: a vizinhança em torno do ponto é dividida em sub-vizinhanças, a partir de onde são construídos histogramas de orientação.
- Estes descritores são comparados quando se deseja detrminar se um ponto chave está presente em duas ou mais imagens com escalas diferentes.

Scale-Invariant Feature Transform (SIFT)

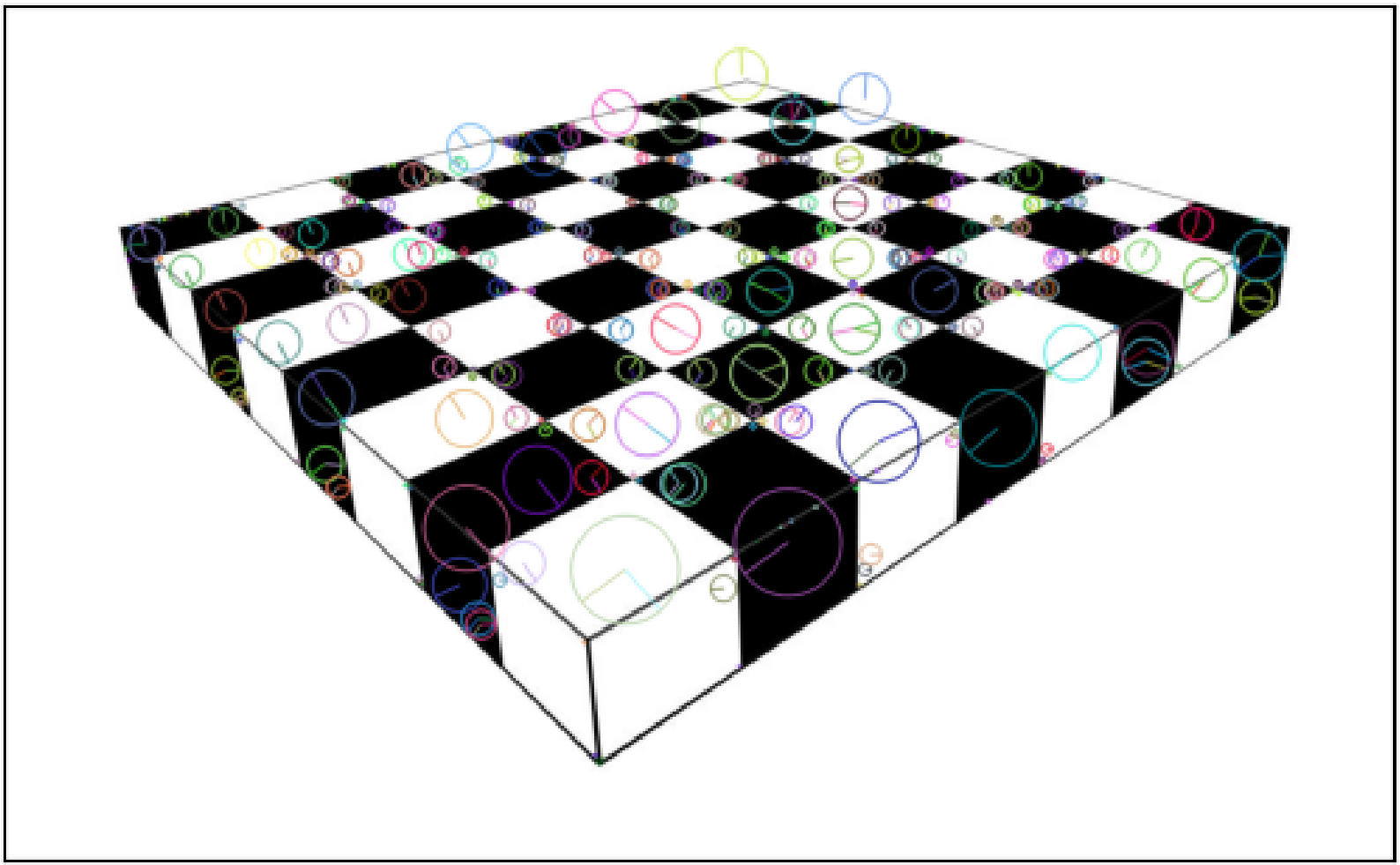


Figura: Pontos chave identificados utilizando o algoritmo SIFT.

Scale-Invariant Feature Transform (SIFT)

Figura: SIFT.

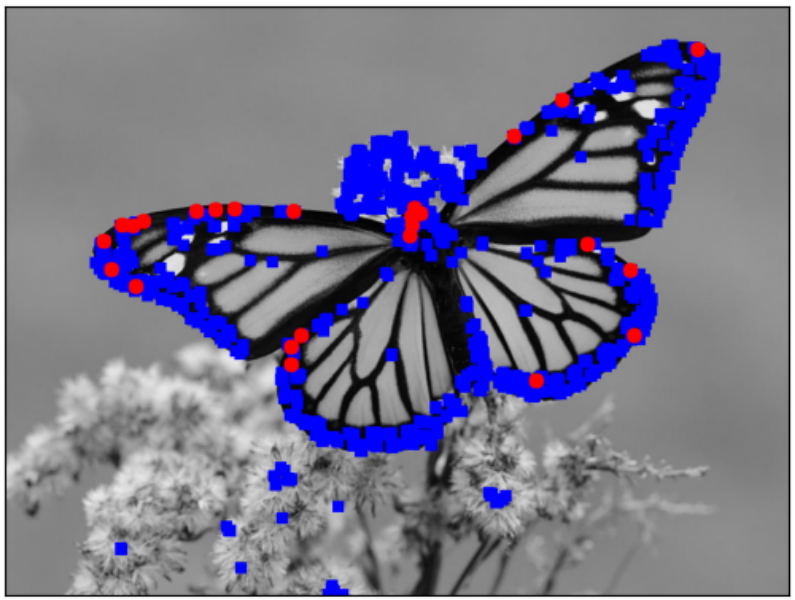


Figura: Harris e Shi-Tomasi.

Speeded-up Robust Features (SURF)

- Embora robusto, o algoritmo SIFT ainda é demasiadamente lento.
- O SURF é proposto como uma alternativa, utilizando aproximaçoes criteriosas para obter resultados comparáveis em menos tempo.
- A principal diferença é a substituição das curvas gaussianas por filtros quadrados.
- Ainda, são utilizadas *wavelets* para determinar a orientação e gerar os descritores das *features*.

Features from Accelerated Segment Test (FAST)

- O algoritmo FAST é uma alternativa ao SURF para aplicações em tempo real, como SLAM em robótica.
- O parâmetro principal é um valor limite t.
- Para um *pixel* de intensidade I_p são verificados os *pixels* pertencentes a um círculo de 16 *pixels* no seu entorno.
- Se todos os n *pixels* de um conjunto contínuo no círculo são mais escuros que $I_p - t$ ou mais brilhantes que $I_p + t$, o *pixel* em questão é um ponto-chave.
- Adicionalmente, apenas os *pixels* nas direções cardinais podem ser verificados, avaliando se há uma maioria simples de *pixels* mais claros ou mais escuros.

Features from Accelerated Segment Test (FAST)

- Esta abordagem é rápida, mas ainda é suscetível a erros caso o parâmetro n seja mal escolhido, além de ser dependente da ordenação das classificações e da concentração de pontos-chave na imagem.
- Estas limitações são compensadas através de aprendizado de máquina.
- Um banco de imagens é utilizado para treinar um classificador que vai determinar a validade de pontos-chave em imagens futuras.

Features from Accelerated Segment Test (FAST)

- Outro problema do FAST é a detecção de múltiplos pontos-chave em locais adjacentes.
- Isto é resolvido com a supressão de pontos não-máximos.
- Essencialmente, é calculada uma métrica V para cada ponto-chave. Caso dois ou mais pontos-chave sejam adjacentes, apenas o de maior valor V será considerado.
- Com estas modificações, o FAST executa rapidamente, mas ainda é suscetível à níveis de ruído elevados, além de depender de um bom ajuste do parâmetro limiar.

Features from Accelerated Segment Test (FAST)

Figura: FAST com máximos suprimidos.

Features from Accelerated Segment Test (FAST)

Figura: FAST sem máximos suprimidos.

Binary Robust Independent Elementary Features (BRIEF)

- Um outro problema encontrado é o tamanho do espaço de memória necessário para armazenar os descritores das *features*. O SIFT utiliza um vetor com 128 elementos, enquanto o SURF utiliza um mínimo de 256.
- Para uma imagem com milhares de pontos chave, rapidamente são alcançados os limites para aplicações com recursos limitados, especificamente em sistemas embarcados.
- Ainda, no processo de identificação de pontos chave, que é utilizado no rastreio de objetos, quanto mais memória é utilizada para os descritores, maior será o tempo de busca.

Binary Robust Independent Elementary Features (BRIEF)

- Uma alternativa é converter o conjunto de descritores em uma *string* de *bits* através de uma lógica de *hash*. A partir daí basta comparar as *strings* resultantes por distância de Hamming para verificar se dois pontos-chave são iguais.
- Isto resolve o problema do tempo para identificação, mas ainda exige o cálculo dos descritores completos.
- O BRIEF é um algoritmo que a partir de um conjunto de pontos chave consegue gerar estas *strings* de *bits* sem a necessidade de calcular os descritores tradicionais.
- Note que o BRIEF fornece descritores mais eficientes a partir de um conjunto de pontos-chave, mas não calcula os pontos em si.

Oriented FAST and Rotated BRIEF (ORB)

- Para além dos problemas já descritos, os algoritmos de detecção de *features* SIFT e SURF possuem uma desvantagem fundamental: eles são patenteados, e o seu uso tecnicamente exige o pagamento de *royalties*.
- O ORB surge como uma alternativa aberta, formulada no OpenCV Labs, que combina a detecção de pontos-chave FAST com o gerador de descritores BRIEF, com algumas modificações, para obter um algoritmo capaz de detectar pontos chave e gerar descritores rapidamente.

Oriented FAST and Rotated BRIEF (ORB)

- A primeira modificação é o uso de uma pirâmide classificatória para extrair *features* invariantes com a escala a partir do FAST.
- A orientação do ponto-chave é obtida a partir do cálculo do centróide da região da imagem centrada no ponto-chave, sendo correspondente à direção do vetor entre o centro da região e o centróide.

Oriented FAST and Rotated BRIEF (ORB)

Pareamento de *features*

- Relembrando: os pontos-chave marcam as regiões da imagem que a caracterizam.
- Consequentemente, conhecendo um ponto-chave e os seus descritores, deve ser possível verificar se este ponto-chave se encontra em uma imagem qualquer.
- Considerando os métodos de localização mais avançados, isto permite o rastreio de objetos robusto a variações de orientação e escala.

Pareamento de *features*

- O primeiro método a ser explorado é o pareamento por força bruta.
- O seu funcionamento é simples:
  1. É selecionada uma *feature* de um primeiro conjunto de pontos-chave.
  2. É tentado um pareamento com todas as *features* de um segundo conjunto, utilizando alguma métrica de distância. São selecionados os pares com a menor distância entre si.
- Para o método de força bruta, a distância normal é apropriada para algoritmos como SIFT e SURF, enquanto para o ORB deve ser utilizada a distância de Hamming.
- Ainda, é possível verificar se o pareamento funciona em "mão dupla". Isto é, se o melhor pareamento do descritor i em A for o descritor j em B, o contrário também deve ser verdadeiro.

Pareamento de *features*

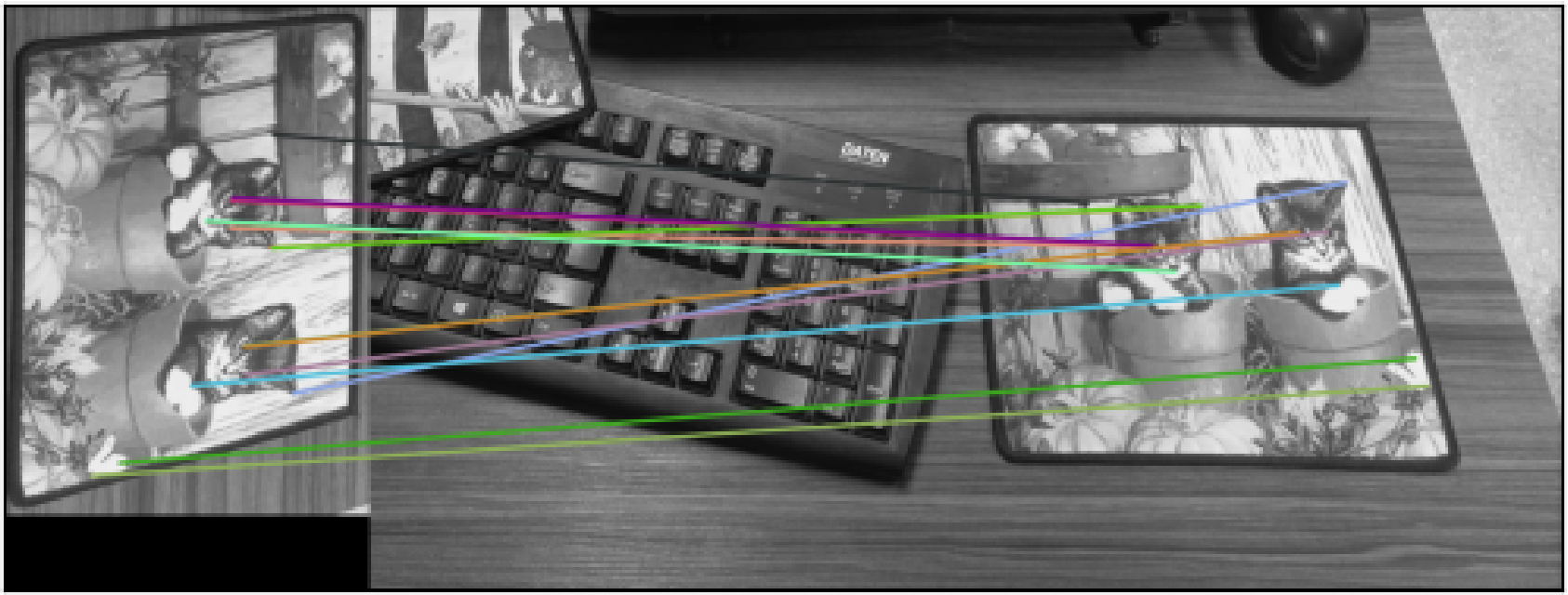


Figura: Pareamento de *features* a partir de descritores ORB utilizando força bruta (20 melhores combinações).

Pareamento de *features*

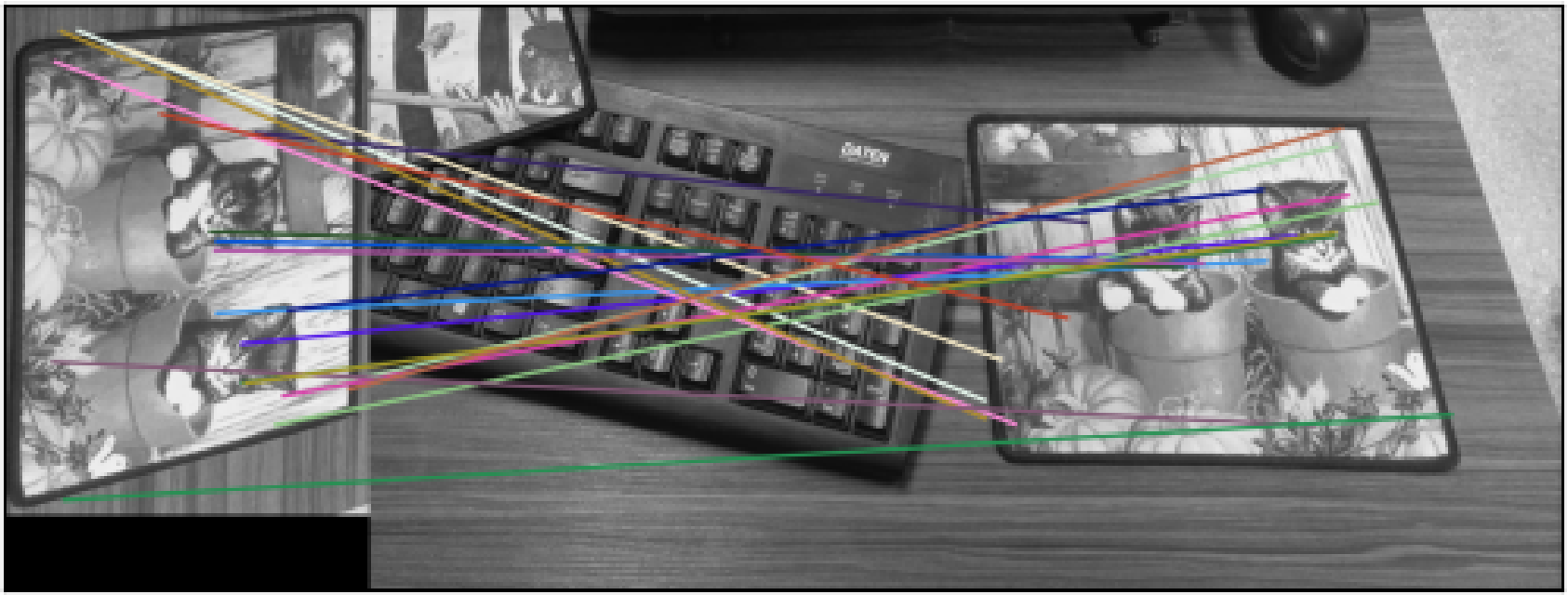


Figura: Pareamento de *features* a partir de descritores SIFT utilizando força bruta (20 melhores combinações).

Pareamento de *features*

- Uma maneira de melhorar o processo de pareamento é a utilização de uma abordagem de k-ésimo vizinho mais próximo (kNN) com o teste de proporção de Lowe.
- A motivação aqui é que eventualmente um par de descritores pode apresentar uma distância pequena entre si, aparentando formar um bom pareamento, quando na prática este não é o caso.
- Isto pode ocorrer por quê há um grande número de descritores por *feature*, e pode haver uma pequena distância entre vários deles que não são relevantes para o pareamento em um determinado caso.
- Este problema poderia ser resolvido através e uma ponderação dos descritores, mas a solução proposta por Lowe é muito mais simples, como será visto a seguir.

Pareamento de *features*

- O primeiro passo é, então, ao invés de identificar apenas o descritor de menor distância na imagem alvo, identificar pelo menos os dois melhores pareamentos pelo critério de distância.
- A suposição do pareamento é que há apenas um correspondente na imagem analisada para uma determinada *feature* da imagem original. Logo, dentre estes dois melhores pareamentos um deles estará necessariamente incorreto.
- O critério de Lowe é baseado na idéia de que se o pareamento de menor distância dentre os dois selecionados, que seria o pareamento correto, não for suficientemente distinto do segundo pareamento, que seria uma representação de um ruído indesejado, então este pareamento deve ser rejeitado.

Pareamento de *features*

- Uma descrição em pseudocódigo do processo seria:

```
matches = get_descriptor_pairs(img1,img2)
good_matches = []
for m,n in matches:
   if m.distance < k*n.distance:
      good_matches.append(m)
```

- Onde $0 \leq k \leq 1$ é um parâmetro de ajuste do quão estrita é a seleção de boas combinações (quanto menor o k, mais estrita).

Pareamento de *features*

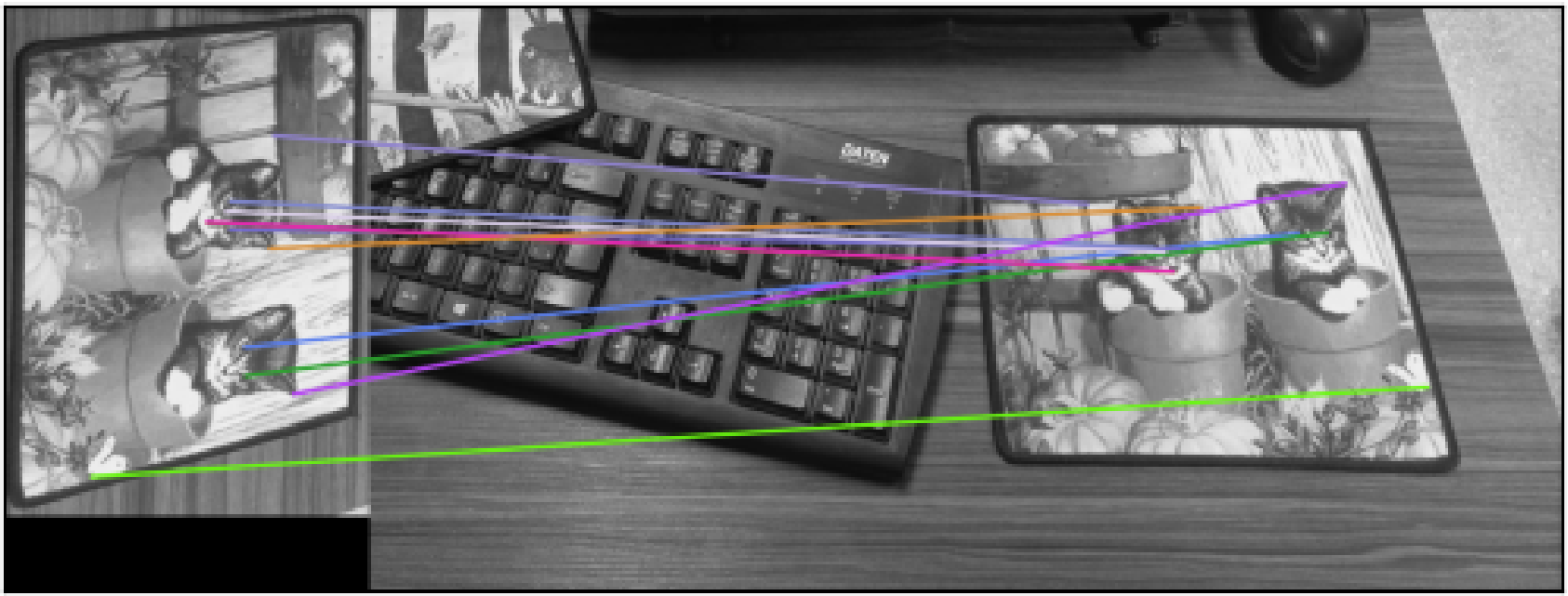


Figura: Pareamento de *features* a partir de descritores ORB utilizando força bruta com kNN e critério de Lowe. Para $k = 0.55$, 12 combinações atenderam ao critério de Lowe.

Pareamento de *features*

- O pareamento por força-bruta é ineficiente para conjuntos de dados grandes e/ou para *features* muito grandes.
- Foi desenvolvia uma biblioteca para busca aproximada de vizinho mais próximo (*Fast Library for Approximate Nearest Neighbors* (FLANN)) que implementa métodos mais apropriados para estes casos.

Pareamento de *features*

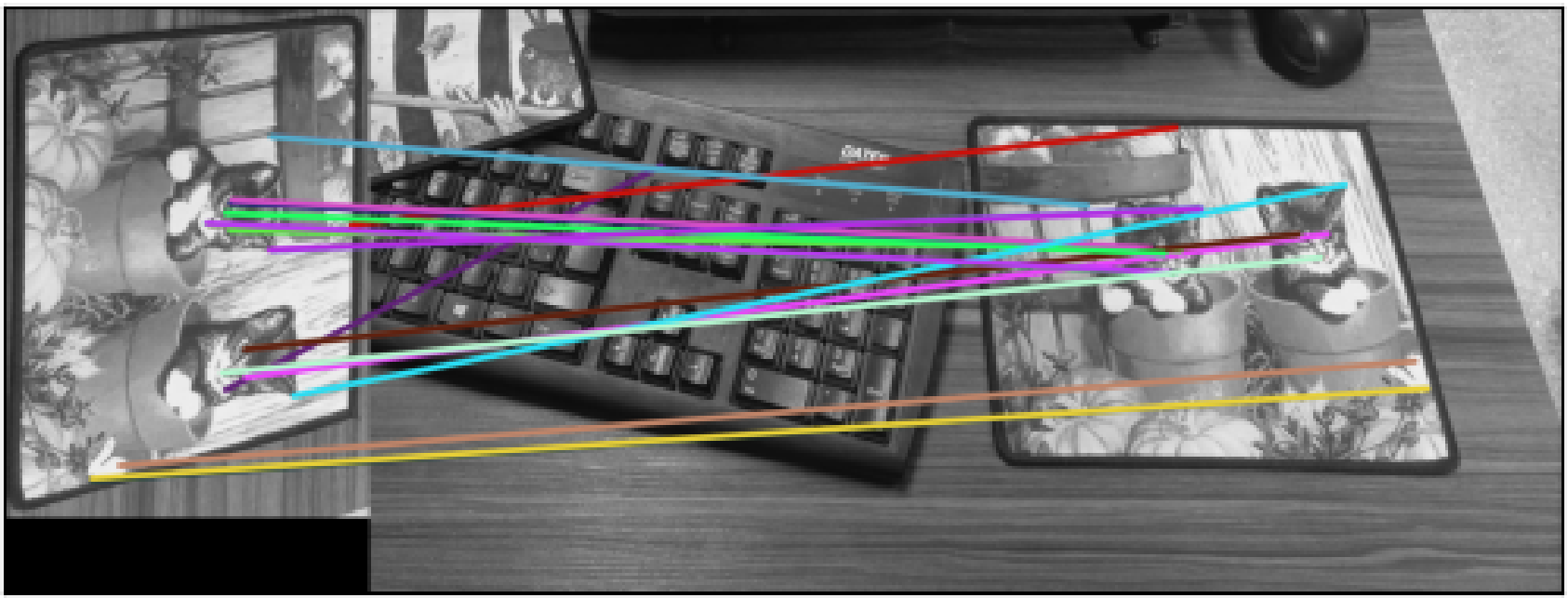


Figura: Pareamento de *features* a partir de descritores ORB utilizando FLANN com critério de Lowe.