Universidade de São Paulo
Instituto de Matemática e Estatística
Bachalerado em Ciência da Computação

Nícolas Nogueira Lopes da Silva

Estudo de Métodos de Detecção de Manipulação Cópia-Colagem em Imagens Digitais

São Paulo
Dezembro de 2018

Estudo de Métodos de Detecção de Manipulação Cópia-Colagem em Imagens Digitais

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Prof. Dr. Paulo André Vechiatto de Miranda

São Paulo
Dezembro de 2018

Resumo

Neste trabalho estudamos métodos para detecção de manipulações em imagens digitais do tipo cópia-colagem. Dentre os métodos pesquisados, investigamos a abordagem baseada no método Patchmatch Generalizado, que busca encontrar semelhanças entre patches (blocos de pixels de tamanhos definidos) de uma imagem baseado na distância de seus histogramas. Após a análise em busca de semelhanças, verificamos estas correspondências para decidir se elas fazem parte de uma região duplicada na imagem analisada. Com o uso de um dataset próprio e outros já existentes, observando o comportamento do método estudado e implementado.

Palavras-chave: Algoritmo de detecção de cópia-colagem, imagens digitais, visão computacional, processamento de imagens.

Abstract

In this work, we studied the methods toward Copy-move Forgery detection in digital images. Among the methods studied, an approach based on the Generalized Patchmatch method, which aims to find similar patches (blocks of pixels of selected sizes) of an image based on the distance of its histograms. After an analysis looking for similar patches, we check these matches to decide whether they are part of a duplicate region in the analyzed image. With the use of a personal and other existing datasets, observing the behavior of the method studied and implemented.

Keywords: Copy-move Forgery detection algorithm, digital images, computer vision, image processing.

Sumário

Lista de Abreviaturas **vii**

1 Introdução **1**

2 Detecção de manipulações em imagens **3**

3 Detecção de manipulação de Cópia-Colagem **5**

3.1 Cópia-Colagem 5

3.2 Fluxo de trabalho 6

3.3 Principais abordagens existentes 7

3.3.1 Abordagem de Fridrich et al. 7

3.3.2 Abordagem de Popescu e Farid 7

3.3.3 Abordagem de Langille e Gong 7

3.3.4 Abordagem de Luo et al. 7

3.3.5 Abordagem de Mahdian e Saic 8

3.3.6 Abordagem de Myna et al. 8

3.3.7 Abordagem de Li et al. 8

3.3.8 Abordagem de Huang et al. 8

3.3.9 Abordagem de Irene et al. 8

3.3.10 Abordagem de Zhang et al. 8

3.3.11 Abordagem de Sergio e Asoke 9

3.3.12 Abordagem de Wang et al. 9

3.3.13 Abordagem de Chen e Hsu 9

3.3.14 Abordagem de Bianchi et al. 9

3.3.15 Abordagem de Bianchi e Piva 9

3.3.16 Abordagem de Barnes et al.Silva 9

4 PatchMatch **11**

5 PatchMatch Generalizado **15**

6 Detecção de Cópia-Colagem utilizando o PatchMatch Generalizado **17**

7 Experimentos e Validação **19**

8 Conclusões **21**

Referências Bibliográficas **23**

Lista de Abreviaturas

CMFD	Detecção de Modificação Cópia-Colagem (*Copy-Move Forgery Detection*)
DWT	Transformada discreta de Wavelet (*Discrete Wavelet Transform*)

Capítulo 1

Introdução

A imagem é um dos principais meios de comunicação atualmente, com seu amplo uso em redes sociais. Ao contrário de outros tipos de dados digitalizados, imagens podem transportar mensagens ou ideias de forma rápida e eficiente. Com o vasto uso de imagens digitais nos últimos anos, a manipulação destas foi facilitada pela disponibilidade de aplicativos de edição de imagens, sejam gratuitos como o GIMP[1] ou até mesmo pagos como o Adobe Photoshop[2]. Com o uso de tais aplicativos, se tornou fácil duplicar e manipular o conteúdo de imagens digitais sem grandes perdas de qualidade e sem deixar vestígios de manipulações (dependendo da habilidade do manipulador assim como a aplicação utilizada). As imagens amplamente compartilhadas na Internet podem ter passado por modificações com intenções maliciosas. Sendo assim, provar a integridade e a autenticidade de imagens digitais se torna um problema real, principalmente em situações em que envolvem casos judiciais, imagens divulgadas em noticiários, imagens médicas, etc.

Dentre as diversas formas de manipulação de imagens, as mais comuns são do tipo Cópia-Colagem (também conhecida como *Copy-Move Forgery* ou Clonagem). Neste tipo de manipulação, uma região da imagem é copiada e colada (com ou sem manipulação da porção copiada da imagem) em uma ou mais regiões da mesma ou de outras imagens. Um dos possíveis objetivos deste tipo de manipulação é o de esconder ou duplicar alguns objetos ou sub-porções da imagem. Geralmente, para aprimorar o resultado de uma manipulação, emprega-se operações de compressão, rotação, redimensionamento, espelhamento e suavização nas regiões duplicadas o que propicia a eliminação de vestígios de adulteração e dificultando o processo de detecção.

Com a aplicação de manipulações no conteúdo de imagens digitais, surge o tópico de estudo de detecção de manipulação de Cópia-Colagem (também chamada de *Copy-Move Forgery Detection* ou CMFD), que envolve técnicas que possam encontrar as regiões semelhantes ou duplicadas da imagem. De acordo com Warif et al. (2016), na literatura de técnicas de manipulação de imagens, a manipulação e detecção para Cópia-Colagem são os tópicos mais amplamente estudados, com um total de 84 artigos científicos com o assunto CMFD indexado pelo *Web of Science*[3] publicados entre 2007 e 2014.

Com o amplo estudo neste campo, diversos métodos foram desenvolvidos para explorar os variados aspectos envolvidos no problema de detecção de manipulação, como quando operações foram realizadas para dificultar o processo de detecção. Segundo Silva (2012), um método para identificação de uma clonagem na qual o segmento duplicado também foi rotacionado pode não ser eficaz na detecção de uma clonagem que envolveu compressão

[1] https://www.gimp.org/
[2] https://www.adobe.com/br/products/photoshop.html
[3] http://wokinfo.com/

JPEG.

Neste trabalho, investigamos a fundo uma abordagem para detecção de manipulações de cópia-colagem em imagens digitais. Esta abordagem foi apresentada por Silva (2012) e é baseada no *PatchMatch Generalizado*, cuja proposta é encontrar correspondências de *patches* (blocos de *pixels* de tamanho definido). A aplicação desta abordagem busca encontrar, para cada *patch* de uma imagem, um conjunto de correspondências. Este conjunto possui *patches* similares com base em um parâmetro de distância, para este trabalho estamos considerando a distância dos histogramas dos *patches* comparados. O método produz um mapa de correspondências, marcando áreas de suspeita de manipulação, auxiliando um perito a tomar uma decisão se houve manipulação na imagem com base em um exame visual posterior sobre as áreas marcadas e suas vizinhanças.

Implementamos o método baseado no *PatchMatch Generalizado* e validamos de maneira qualitativa (visual) e quantitativa, com uso de um dataset pessoal composto por 5 imagens e com o uso de 10 imagens do dataset CoMoFoD[4], que é especificamente composto por imagens manipuladas através de operações de clonagem com alto grau de realismo.

[4] http://www.vcl.fer.hr/comofod/comofod.html

Capítulo 2

Detecção de manipulações em imagens

De acordo com Warif et al. (2016), os métodos de detecção de manipulação de imagens podem ser amplamente categorizados em abordagens ativas e passivas, de acordo com a presença de informação adicional. A abordagem ativa é baseada na informação vinculada na imagem digital para detecção de adulteração como marcas d'água e assinaturas digitais. Estas informações podem ser usadas para assegurar a autenticidade de uma imagem, entretanto, esta abordagem exige que estas informações sejam vinculadas durante o processo de captura ou posteriormente por pessoas autorizadas. A grande parte das imagens na internet não possuem informação sobre sua origem, nestes casos este tipo de abordagem é impossível ou ineficaz.

Por outro lado, a abordagem passiva é capaz de detectar manipulações sem informação adicional, extraindo características intrínsecas à imagem levando em consideração a detecção de adulteração e identificação do dispositivo de origem. As adulterações ainda podem ser classificadas em dependentes ou independentes, no caso as adulterações do primeiro tipo utilizam regiões de uma imagem inicial com cópia-colagem na mesma imagem ou para outras (*splicing*), as do segundo tipo não dependem de uma imagem inicial, como compressões e reamostragem. Com a ampla quantidade de artigos envolvendo detecção de manipulações de imagens, muitas classificações foram apresentadas, a visão geral abordada pela maioria dos autores está representada na figura 2.1.

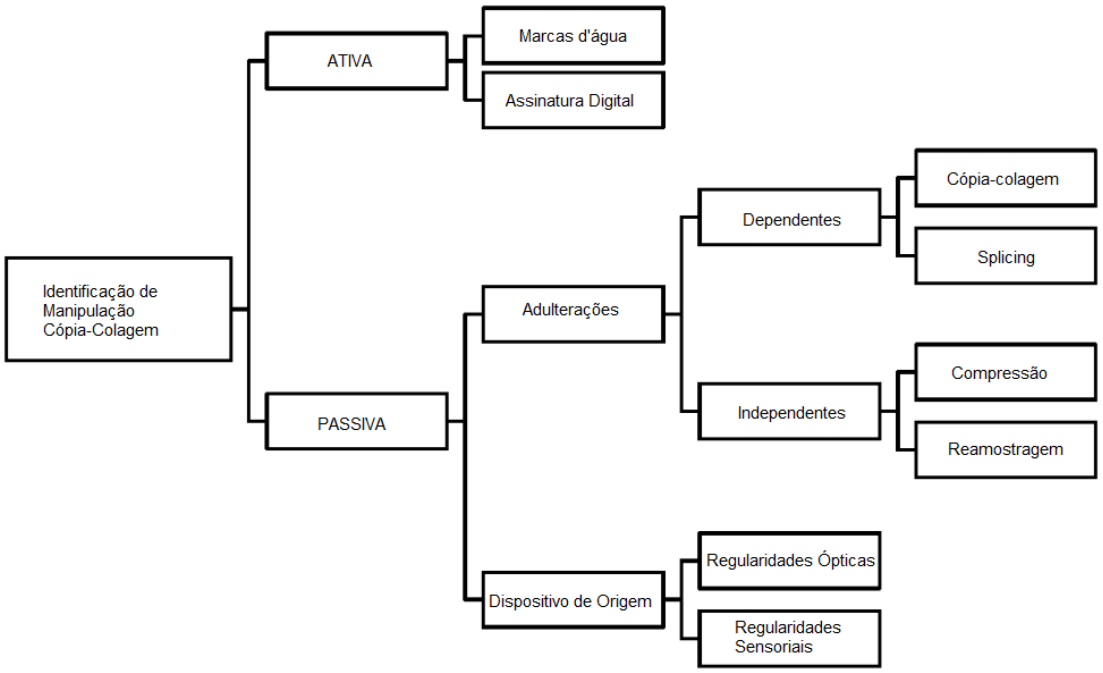

Figura 2.1: *Visão geral da classificação das manipulação em imagens digitais. Adaptado de Warif et al. (2016).*

Capítulo 3

Detecção de manipulação de Cópia-Colagem

Neste capítulo, apresentamos uma visão geral da manipulação cópia-colagem, o fluxo de trabalho envolvendo os métodos que buscam detectá-las, assim como os desafios enfrentados e como os principais métodos existentes na literatura lidam com eles.

3.1 Cópia-Colagem

Uma manipulação de cópia-colagem, também conhecido como clonagem, consiste em clonar regiões em uma imagem. É possível utilizar este tipo de manipulação para ocultar ou multiplicar elementos na imagem, como pessoas, objetos, placas ou mensagens, etc. Para isso, seleciona-se uma região da imagem e, caso necessário, aplicam-se operações sobre este segmento (e.g. rotação). Em seguida, a região selecionada é posicionada sobre os elementos que se busca ocultar, duplicando a região original. Um exemplo deste tipo de manipulação pode ser visto na figura 3.1.

Figura 3.1: *Exemplo de uma imagem que sofreu manipulação do tipo cópia-colagem.*

Dentre as operações que podem ser efetuadas sobre a região selecionada (adulteração dependente), estão o redimensionamento (ou ajuste de escala), espelhamento (vertical ou horizontal), rotação, suavização de bordas, aplicação de filtros, ajuste de iluminação, etc. Contudo, após o reposicionamento da região selecionada, operações globais ainda podem ser aplicadas (adulteração independente) como adição de ruídos gaussianos e compressões. Essas operações apresentam desafios aos métodos que buscam identificar estas adulterações, pois modificam os valores dos *pixels* dos segmentos duplicados e da imagem como um todo, alteram a disposição dos elementos em relação aos segmentos originais ao eliminar elemen-

tos já existentes na sobreposição causada pelo segmento duplicado. A figura X ilustra os diferentes efeitos das operações sobre uma imagem.

Os métodos de detecção buscam identificar as regiões similares dentro de uma imagem. As regiões similares, contudo, podem ter sofrido das operações apresentadas anteriormente, além disso, regiões na imagem podem ser naturalmente similares como um céu azul, asfalto, superfícies de cor uniforme e elementos fisicamente parecidos na imagem. Os métodos de detecção de clonagem envolvem técnicas de detecção do tipo ativa envolvendo adulterações dependentes e independentes, como vistas no capítulo 2. A seguir, é apresentado o fluxo de trabalho presente na maioria das técnicas de detecção de clonagem em imagens digitais.

3.2 Fluxo de trabalho

Na CMFD, o fluxo de trabalho (*workflow*) consiste em quatro etapas: pré-processamento, extração de características, correspondência e visualização.

O primeiro estágio do processo da CMFD é tipicamente o pré-processamento, que é uma etapa opcional. No pré-processamento, procura-se melhorar a imagem por suprimir distorções indesejáveis e aprimorar as características da imagem. Imagens coloridas que possuem canais RGB (R para vermelho, G para verde e B para azul) podem ser convertidas para imagens em tons de cinza (*greyscale*), este tipo de pré-processamento parece ser o mais utilizado entre os autores na literatura. A conversão pode ser utilizada ao somar os canais de cores com pesos na imagem, produzindo um único canal de cor variando de 0 a 255. A equação a seguir representa o cálculo realizado ao efetuar esta operação:

$$I = 0,228R + 0,587G + 0,114B$$

Este tipo de operação é realizada para reduzir a dimensionalidade dos dados e aprimorar a distinção visual de elementos na imagem. Este processo também faz com que a complexidade do processamento possa ser reduzida e a velocidade de processamento seja aumentada. Além da conversão de cores, a divisão em blocos é usada como parte do pré-processamento na CMFD. A divisão em blocos é um método que divide a imagem em um número de blocos que podem se sobrepor ou não. A divisão em blocos busca reduzir o tempo computacional para a etapa de correspondência em comparação com a busca exaustiva *pixel* a *pixel*.

Após o pré-processamento, a extração de características permite selecionar informações relevantes para representar as aspectos de interesse de uma imagem. Métodos comuns de extração de características presentes na literatura são Transformada Discreta do Cosseno (*Discrete Cosine Transform ou DCT*) e Transformada Discreta Wavelet (*Discrete Wavelet Transform ou DWT*), Transformada de características invariantes à escala (*Scale-Invariant Feature Transform ou SIFT*), etc.

A extração de características é seguido por um estágio de correspondência (*matching*) que procurar similaridades entre duas ou mais características na imagem. Nesta etapa, as possíveis manipulações de clonagem são determinadas. A execução das técnicas de correspondência são principalmente baseadas em blocos ou em pontos-chave dependendo das características extraídas. Por exemplo, de acordo com Warif (2016), as características obtidas pela DCT são correspondidas em blocos, enquanto as obtidas pelo SIFT, baseadas em pontos-chave invariantes são correspondidas pela distância do vizinho mais próximo de todos os pontos no espaço de características.

Concluindo o processo, a CMFD pode ser visualizada para mostrar as regiões com possível adulteração na imagem modificada. A visualização de abordagens baseadas em blocos usualmente é representada por colorir as regiões suspeitas com alguma cor específica na ima-

gem analisada ou através da utilização de um mapa de correspondências, marcando como uma máscara as regiões de possível adulteração. Por outro lado, as abordagens baseadas em pontos-chave usualmente apresentam segmentos de reta entre os pontos de correspondência.

3.3 Principais abordagens existentes

Como explicam Birajdar et al. (2013), cópia-colagem é o tipo de técnica de adulteração mais comum devido à sua facilidade e efetividade. Devido ao extenso estudo na literatura, existem diversos algoritmos para detecção de cópia-colagem. Uma abordagem direta consiste na comparação de cada par de *pixels* da imagem. A complexidade deste método é, entretanto, $O(N^2)$ com N sendo a quantidade de *pixels* da imagem, o que torna este método impraticável. Além disso, Farid (2008) argumenta que a quantidade de falsos-positivos (regiões incorretamente identificadas como duplicadas) pode ser elevado.

3.3.1 Abordagem de Fridrich et al.

A primeira tentativa de identificar regiões adulteradas foi investigada por Fridrich et al. (2003). Os autores propuseram um método de detecção de manipulação cópia-colagem utilizando Transformada Discreta do Cosseno (DCT) por blocos sobrepostos e com uma representação lexicográfica (semelhante à ordem alfabética) para evitar alta carga computacional, entretanto sem tratar casos onde ocorreram operações de rotação e redimensionamento.

3.3.2 Abordagem de Popescu e Farid

Popescu e Farid (2004) apresentaram um método que utiliza análise de componentes principais (PCA) para a representação de segmentação da imagens com uso de blocos quadrados sobrepostos, capturando as informações mais representativas de cada bloco e analisando apenas este conjunto reduzido de dados. A detecção baseada em PCA resultou na redução do custo computacional, sendo este $O(N_t N log N)$ onde N_t é a dimensionalidade da representação PCA truncada e N o número de *pixels* da imagem. Os resultados informados pelos autores mostram que o método é eficaz na detecção em imagem comprimidas em JPEG e afetadas pelo uso de Ruído Gaussiano em diversos graus.

3.3.3 Abordagem de Langille e Gong

Para lidar com a complexidade computacional, o uso de uma árvore k-d (árvore k-dimensional) foi proposta por Langille e Gong (2006) no qual um método de busca por blocos com intensidades similares foi utilizado. O algoritmo resultando possui complexidade $O(N_b N_s)$ onde N_s é o tamanho da vizinhança de busca e N_b é a quantidade de blocos. A Correlação Cruzada Normalizada de Média Zero (ZNCC) foi utilizada como métrica de similaridade e resultados de detecção precisos foram obtidos através da pesquisa dentro de no máximo 100 blocos vizinhos no vetor de blocos ordenado.

3.3.4 Abordagem de Luo et al.

Luo et al. (2006) introduziram o método de detecção e localização de clonagem baseado em dividir a imagem em pequenos blocos sobrepostos, e então comparar a similaridade desses blocos e identificar regiões possivelmente duplicadas utilizando características baseadas em intensidade. O algoritmo em questão possui menor complexidade, sendo robusto contra

ataques fortes e de vários tipos provenientes de operações realizadas após a cópia como compressão, adição de ruídos e combinação destas operações, resultando em acurácia de 96,31% e falso-negativo de 9,66% em casos de operações mistas.

3.3.5 Abordagem de Mahdian e Saic

O método para detecção de regiões duplicadas baseado em Momentos Invariantes a Borramento (*Blur Moment Invariants*), PCA e árvores k-d foram descritas por Mahdian e Saic (2007). Na análise de similaridade entre as regiões, cada bloco pertencente à vizinhança das duas regiões também é comparado. O método funciona bem em casos de compressão, mas possui elevado tempo de execução.

3.3.6 Abordagem de Myna et al.

Myna et al. (2007) desenvolveram um método utilizando DWT que detecta e localiza clonagens. A redução de dimensionalidade é obtida ao aplicar a DWT na imagem, uma exaustiva busca é executada para identificar blocos similares na imagem mapeando-os para coordenadas log-polares como uma estratégia de representá-los de maneira robusta a operações de escala e rotação e adotando correlação de fase como critério de similaridade.

3.3.7 Abordagem de Li et al.

Li et al. (2007) apresentaram um método baseado em Detecção em Valores Singulares (*Singular Value Decomposition ou SVD*) para reduzir a dimensionalidade e DWT para a detecção de regiões duplicadas. Regiões duplicadas foram localizadas por ordenação lexicográfica e a detecções de vizinhança para todos os blocos mesmo quando a imagem foi altamente comprimida ou quando sofreu processamento de bordas.

3.3.8 Abordagem de Huang et al.

Descritores SIFT são estáveis em relação à iluminação, rotação e escala e foram utilizados por Huang et al. (2008) para detectar regiões clonadas de uma imagem. O método possui boa acurácia em diferentes tipos de operações como compressão, rotação, ruído, escala e também é robusto na combinação dessas operações.

3.3.9 Abordagem de Irene et al.

Uma nova metodologia baseada no SIFT é avaliada para estimar parâmetros de transformação geométrica (translação horizontal e vertical, fatores de escala e ângulo de rotação) com alta confiabilidade, além de detectar imagens manipuladas doi apresentado por Irene et al. (2011). O método proposto atinge taxa de verdadeiros-positivos de cerca de 100%. A técnica também é utilizada para detecção de *splicing*.

3.3.10 Abordagem de Zhang et al.

A abordagem CMFD baseada em tranformadas wavelet e correlação de fase foi criada para estimar o deslocamento espacial entre uma região copiada e colada foi proposta por Zhang et al. (2008). Mas a performance depende da localização das regiões de cópia-colagem.

3.3.11 Abordagem de Sergio e Asoke

Para executar uma busca eficiente, sobreposições de blocos de *pixels* são mapeados para descritores unidimensionais derivados de mapeamentos log-polar para detecção automatizada e localização de regiões duplicadas afetadas por reflexão, rotação e escala em imagens é o foco da abordagem de Sergio e Asoke (2011).

3.3.12 Abordagem de Wang et al.

Wang et al. (2010) implementou um algoritmo para localizar a região adulterada em uma compactação sem perdas da imagem adulterada quando a sua região inalterada é saída de descompressor de JPEG. O PCA é empregado para separar diferentes ruídos de quantização de frequências espaciais, isto é, ruído de quantização de baixa, média e alta frequência, extraindo as altas frequências para a localização das regiões de adulteração. Entretanto, esse método falha na detecção caso a região de adulteração tenha informação de alta frequência ou quando a imagem de origem foi salva no formato JPEG com qualidade maior que a qualidade da imagem adulterada.

3.3.13 Abordagem de Chen e Hsu

Chen e Hsu (2011) apresentaram uma técnica para detectar recompressão alinhada a blocos ou desalinhada ao formular as características periódicas de imagens JPEG tanto em domínios espaciais como em domínios de transformação. A abordagem é limitada se uma operação global, como ruído branco gaussiano aditivo ou desfoque, for aplicada com um grande nível de distorção antes da recompressão.

3.3.14 Abordagem de Bianchi et al.

Bianchi et al. (2011) aplicaram um teste estatístico para diferenciar entre regiões originais e forjadas em imagens JPEG calculando modelos de probabilidade para os coeficientes DCT de regiões comprimidas individualmente e duplamente juntamente com uma estimativa do fator de quantização primário no caso de compressão dupla.

3.3.15 Abordagem de Bianchi e Piva

Bianchi e Piva (2012) propuseram um método para detectar em uma imagem digital a presença de compressão JPEG não-alinhada baseada na observação de que os coeficientes DCT exibem uma periodicidade inteira quando o DCT em blocos é computado de acordo com a grade da compressão JPEG primária.

3.3.16 Abordagem de Barnes et al.Silva

Barnes et al. (2010) apresenta o *PatchMatch Generalizado* (*Generalized PatchMatch*), que apesar de não possuir estratégia voltada diretamente à solução do problema de clonagem, pode ser utilizada para enfrentar os desafios propostos. Esta técnica é uma expansão do algoritmo *PatchMatch*, proposto por Barnes et al. (2009), que possibilita a localização de correspondências de blocos (*patches*) em uma imagem de maneira rápida. Silva (2012) apresenta uma contribuição para este método para a solução do problema de detecção de manipulação de cópia-colagem, esta contribuição será explorada no capítulo 5.

Capítulo 4

PatchMatch

Neste capítulo será abordada a técnica precursora do *PatchMatch Generalizado* que queremos estudar. Em sua versão original, o *PatchMatch* proposto por Barnes et al. (2009) é uma abordagem randomizada eficiente para o seguinte problema: para cada bloco *pxp* de uma imagem A, encontrar o bloco que seja aproximadamente o vizinho mais próximo na imagem B, minimizando a soma do quadrado da diferença entre as intensidades do *pixels* correspondentes entre os blocos. O Espaço de Vizinhos mais Próximos (*Nearest Neighbor Field NNF*) é uma função $\mathbf{f} : A \rightarrow \mathbb{R}^2$, definida sobre todos os possíveis blocos (localização do centro dos patches) na imagem A, para alguma função distância D entre dois blocos. Dado a coordenada de bloco $\mathbf{a}$ na imagem A e seu correspondente vizinho mais próximo $\mathbf{b}$ na imagem B, $\mathbf{f}(\mathbf{a})$ é simplesmente $\mathbf{b}$. Ao invés de se percorrer a imagem inteira na busca pela correspondência, ou que sejam usadas árvore e técnicas para redução da dimensionalidade (e.g. PCA), o *PatchMatch* adota uma estratégia baseada em Propagação (*Propagation*) de boas correspondências para blocos vizinhos e Busca Aleatória (*Random Search*) na imagem com o objetivo de melhorar as correspondências encontradas, o conceito de correspondências entre imagens pode ser melhor entendido ao observar a figura 4.1.

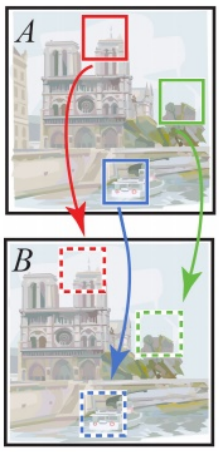

Figura 4.1: *Exemplo de correspondências entre imagens. Retirado de Barnes et al. (2009).*

A métrica de distância utilizada no *PatchMatch* é a Soma dos Quadrados das Diferenças (*Sum of Squared Differences SSD*). A equação a seguir representa o cálculo da SSD entre dois blocos, P_1 e P_2 a serem comparados. A subtração é realizada de forma matricial, a potência é calculada *pixel* a *pixel* da matriz resultante, realizando uma soma de todos os valores da matriz posteriormente:

$$SSD(P_1, P_2) = \sum_{p_1 \in P_1, p_2 \in P_2} (p_1 - p_2)^2$$

O resultado esperado de uma execução do *PatchMatch* é um conjunto de correspondências, uma para cada bloco possível em dada imagem. Esse conjunto é o NNF descrito anteriormente, ele também pode ser entendido como uma matriz, de dimensões idênticas à da imagem original, onde as coordenadas de algum elemento do bloco identifica ele na matriz (usualmente o *pixel* central ou o primeiro do bloco quando visitamos em *scan order* da esquerda para a direita), além de guardar a correspondência, o NNF deve guardar a distância de similaridade adotada entre os blocos da correspondência. As etapas do algoritmo *PatchMatch* consistem por uma Inicialização e por etapas de Iteração que serão descritas a seguir:

1. **Inicialização**: Nesta etapa, inicializamos o NNF, associando uma correspondência aleatória para cada bloco da imagem.

2. **Iteração**: Após a inicialização, o algoritmo executa etapas de aprimoramento do NNF. A cada iteração, o NNF é inspecionado da mesma maneira que a imagem original é inspecionada, em *scan order*. Assim, cada bloco visitado no NNF passa por dois mecanismos de melhoria: propagação e busca aleatória.

   (a) **Propagação**: Nesta etapa, melhoramos as correspondências ao observar os blocos vizinhos ao bloco (x, y) e manter a que representa menor distância, sendo eles o da esquerda $(x-1, y)$ e o de cima $(x, y-1)$ em iterações ímpares, e o da direita $(x-1, y)$ e o de baixo $(x, y-1)$ em iterações pares. Este procedimento ajuda a disseminar correspondências para regiões próximas que estejam em uma mesma estrutura de imagem.

   (b) **Busca Aleatória**: Podemos escapar de ocasionais mínimos locais (devido à possíveis ocorrências de correspondências muito próximas de regiões muito parecidas da imagem, mas que não são as correspondências desejadas) ao realizar uma busca ao redor as correspondências encontradas, testando uma série de blocos de posição aleatória na imagem em distâncias que variam exponencialmente da correspondência presente no início da busca aleatória. Seja v_0 a nossa correspondência inicial, nós tentamos melhorar a correspondência de um bloco que aponta para v_0 ao testar uma série de blocos u seguindo a seguinte equação:

   $$u_i = v_0 + w\alpha^i R_i$$

   onde R_1 é uma coordenada aleatória uniforme em $[-1, 1] \times [-1, 1]$, w é um raio de busca máxima (no nosso caso a maior dimensão da imagem), e α é a razão entre as janelas de busca. Os blocos são examinados para valores de $i = 0, 1, 2, ...$ até que o atual raio de busca $w\alpha^i$ seja menor que 1 *pixel*.

As etapas podem ser melhor compreendidas através da figura 4.2. Um resultado satisfatório (uma condição de parada para as iterações) quantitativo não possui uma definição objetiva, a convergência do método ocorre quando o NNF alcança mapeamento satisfatório para a maioria de seus blocos, de modo que se reconstruíssemos a imagem original a partir do NNF final, a imagem gerada seria muito parecida visualmente com a original. Segundo

Barnes et al. (2009), o *PatchMatch* converge com alta probabilidade em, aproximadamente, cinco iterações.

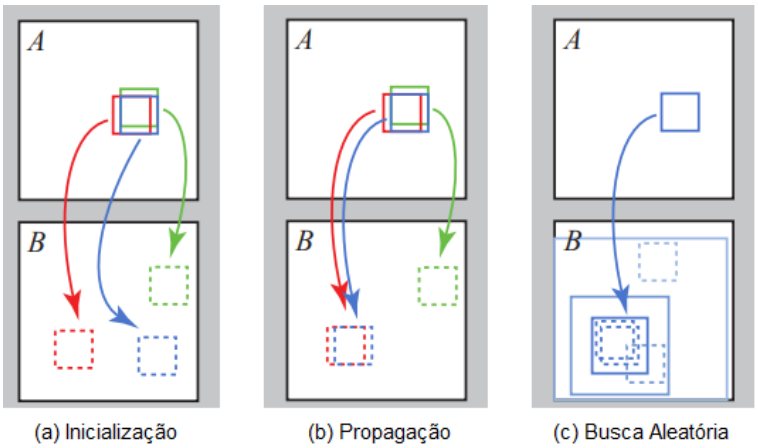

Figura 4.2: *Representação das etapas do método* PatchMatch. *Adaptado de Barnes et al. (2009).*

Capítulo 5

PatchMatch Generalizado

A versão generalizada do *PatchMatch* foi proposta por Barnes et al. (2010) e recebeu esta denominação por se diferenciar do algoritmo original em certos pontos. O primeiro é que o método possui um número pre-definido de K correspondências para cada bloco em sua NNF. O problema passa a tratar os K vizinhos mais próximos para cada bloco. O algoritmo suporta agora a busca de blocos similares em todas as escalas e orientações, os autores passaram a permitir que a comparação com a utilização de descritores SIFT possam ser utilizadas. Duas novas etapas foram inseridas: Enriquecimento Inverso (*Inverse Enrichment*) e Enriquecimento Direto (*Forward Enrichment*). Duas estruturas novas passam a ser utilizadas:

1. **Max-Heap**: Trata-se de uma estrutura para armazenamento de dados baseada no conceito de árvore binária e que possibilita recuperação e inserção eficiente de elementos. No Max-Heap, cada nó armazena as coordenadas x, y e distância adotada (sendo a chave do Max-Heap), e possui chave maior do que seus filhos à esquerda e direita. O elemento com maior valor de chave se encontra na raiz e pode ser recuperado em tempo $O(1)$, e extração em tempo $O(logN)$, assim como a inserção de um elemento. Desta forma a NNF é compreendida agora como uma matriz de Max-Heaps.

2. **Lista Encadeada**: uma lista encadeada simples é utilizada para cada bloco, onde cada elemento guarda valores de coordenadas e distância referentes a outro bloco. É utilizado na etapa de Enriquecimento Inverso.

As etapas do método consistem em: Inicialização e Iteração. Para cada Iteração temos etapa de Propagação, Busca Aleatória, Enriquecimento Inverso e Enriquecimento Direto. As etapas são descritas a seguir:

1. **Inicialização**: Semelhante ao *PatchMatch* original, porém, agora as K correspondências são escolhidas aleatoriamente.

2. **Iteração**: para cada iteração temos etapa de Propagação e Busca Aleatória como antes, porém, temos novas etapas referentes ao enriquecimento, sendo elas o Enriquecimento Inverso e o Enriquecimento Direto.

   (a) **Propagação**: A comparação entre as correspondências vizinhas ao bloco presente no NNF inicialmente passa a ser entre o bloco examinado e os elementos dos heaps que possuem maior distância destes vizinhos, de modo a retirar as piores distâncias de cada heap. Continua a utilizar a estrutura de iterações ímpares observar blocos à esquerda e acima da correspondência original, e em iterações pares observa-se blocos abaixo e à direita.

(b) **Busca Aleatória**: De forma semelhante ao original, porém, passa a aplicar as buscas para cada um dos K elementos de cada heap.

(c) **Enriquecimento**: É uma etapa para tentar otimizar o método, propagando boas correspondências de um bloco para as duas próprias correspondências, considerando um conjunto mais rico de correspondências. O enriquecimento é subdividido em inverso e direto.

- **Enriquecimento Inverso**: Ao inserir ou remover quaisquer elementos do max-heap, agora é necessário realizar a manutenção da outra estrutura de dados que são as listas encadeadas, para dado bloco b está presente em sua lista encadeada as correspondências que apontam para o bloco em questão. Desta forma ao inserir ou remover elementos do max-heap, é necessário remover ou acrescentar as correspondências nas listas encadeadas. No processo de enriquecimento inverso, para dado bloco, se analisa todos os blocos presentes na lista encadeada do bloco em questão.
- **Enriquecimento Direto**: Para cada bloco observa as K^2 correspondências de suas correspondências como possíveis candidatos de melhoria para o mapeamento. A pior correspondência é removida (a raiz, a não ser que todas as K^2 correspondências sejam piores que ela) e a melhor correspondência encontrada é inserida no max-heap do bloco em questão. De acordo com os autores, o método se torna mais eficaz ao realizar o enriquecimento inverso e depois o direto, esta característica foi posteriormente validade no trabalho de Silva (2012).

Assim como no *PatchMatch* original, o método converge quando comparamos a reconstrução à imagem original. Na figura 5.1 temos a comparação de uma das imagens de teste com sua reconstrução utilizando 5 iterações, K = 5 e bloco de lado 7.

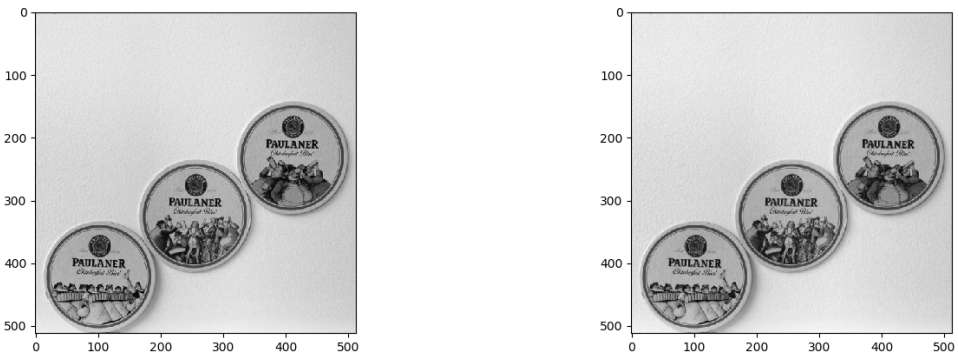

Figura 5.1: *Comparação imagem original (à esquerda) a e reconstrução (à direita) utilizando* PatchMatch Generalizado *em 5 iterações, K = 5 e bloco de lado 7.*

Capítulo 6

Detecção de Cópia-Colagem utilizando o PatchMatch Generalizado

Ao realizar execução do *PatchMatch Generalizado*, o NNF final possui para cada bloco, K boas correspondências, boas o suficiente para reconstruir a imagem original sem diferenças perceptíveis entre a reconstrução e original. Com isso, é possível apontar blocos duplicados na cena, revelando a manipulação. Precisamos estabelecer algumas regras, pois podem existir blocos pode possivelmente estar apontando para eles mesmos. Consideremos um mapa de detecção de tamanho igual ao da imagem original, percorreremos o NNF em *scan order*. Em seguida, para cada bloco comparamos o conjunto de correspondências de cada bloco com o conjunto de correspondências de seus vizinhos (blocos à direita e abaixo no NNF), caso tais correspondências sejam similares, marcamos no mapa de detecção os todos os *pixels* de ambos os blocos. Dois conjuntos de correspondências são similares, caso para qualquer par de blocos $(ax, ay) \in A$ e $(bx, by) \in B$, estes se encontram a uma distância física máxima T um do outro, e ambos possuem uma distância adotada menor do que um limiar D máximo arbitrário.

Silva (2012) ao testar esta versão do *PatchMatch* encontrou problemas para validar a estratégia por encontrar uma alta de taxa de falsos-positivos e falsos-negativos. Concluíram então que poucas informações foram oferecidas pelo autor. Ao tentar implementar desta forma, não consegui replicar a detecção de cópia-colagem, embora a imagem tenha sido reconstruída de forma satisfatória. Silva (2012), propôs um método diferente para a detecção de clonagem:

1. O *PatchMatch Generalizado* é executado da mesma maneira que anteriormente;
2. Percorre-se o NNF em *scan order*;
3. Visita-se o max-heap de correspondências para cada bloco;
4. Caso uma correspondência do bloco esteja a uma distância física inferior a um limiar T do bloco ao qual o max-heap pertence, não prosseguimos com a análise e passamos para a correspondência seguinte. Isto se deve ao fato de blocos poderem ter como correspondência eles mesmos ou regiões muito próxima deles.
5. Comparamos a região ao redor do bloco (x, y) de deslocamento predefinido com as regiões ao redor de cada uma das correspondências, respeitando os limites das bordas de imagem por não comparar vizinhos para blocos que não possam ser acessados. Desta forma as correspondências crescem em direções semelhantes.

6. Ao comparar as regiões correspondentes e elas forem similares i.e. a distância de similaridade entre elas é menor que um limiar D. marcamos as regiões no mapa de detecção.

7. Se uma região já foi marcada, ela não é examinada novamente.

O autor deste último método passa a adotar a similaridade baseada em Soma de Diferenças Absolutas (*Sum of Absolute Differences - SAD*), por também considerar pequenas alterações nas regiões duplicadas.

Capítulo 7

Experimentos e Validação

A implementação do método proposto por Silva (2012) foi feita em Python por considerar relativamente fácil a manipulação de matrizes e imagens. O dataset externo utilizado foi o CoMoFoD (*Image Database for Copy-Move Forgery Detection*) com 10 imagens variadas, e 10 imagens construídas por mim mesmo. As imagens coloridas passaram pelo pre-processamento que as tornou imagens em tons de cinza para reduzir a dimensionalidade do problema. Utilizei inicialmente K = 8, blocos de lado 7, deslocamento de blocos na fase de detecção de cópia colagem igual à metade do lado de um bloco.

Os testes realizados consistiram em avaliar dois pontos: o processo da construção do NNF (ao comparar os resultados obtidos) e avaliar o resultado com a máscara de colagem utilizada para a imagem manipulada.

Ao executar o método, em todas as imagens do dataset houve boa reconstrução da imagem, porém, ao executar a detecção de cópia-colagem, muitos falsos-positivos puderam ser observados em praticamente todas as imagens testadas, como por exemplo na figura 7.1. A máscara original mostra que o método conseguiu determinar blocos referentes à colagem, porém, não conseguiu detectar toda a área necessária (falso-negativos), além de ter detectado área que não participaram da colagem (falsos-positivo). Como os testes apresentaram grande discrepância de resultados identificáveis numericamente (como parâmetros quantitativos de distorção). Eu tentei verificar o que poderia ter ocorrido de errado com minha implementação, ajustar os parâmetros, porém, não obtive grande sucesso, embora eu suspeite que a parte que explora os vizinhos de blocos já detectados não esteja sendo considerada. Os parâmetros de tamanho de blocos não apresentaram grande diferença e os de distância apresentaram grande diferença de forma muito rápida (de acordo com o método estudado).

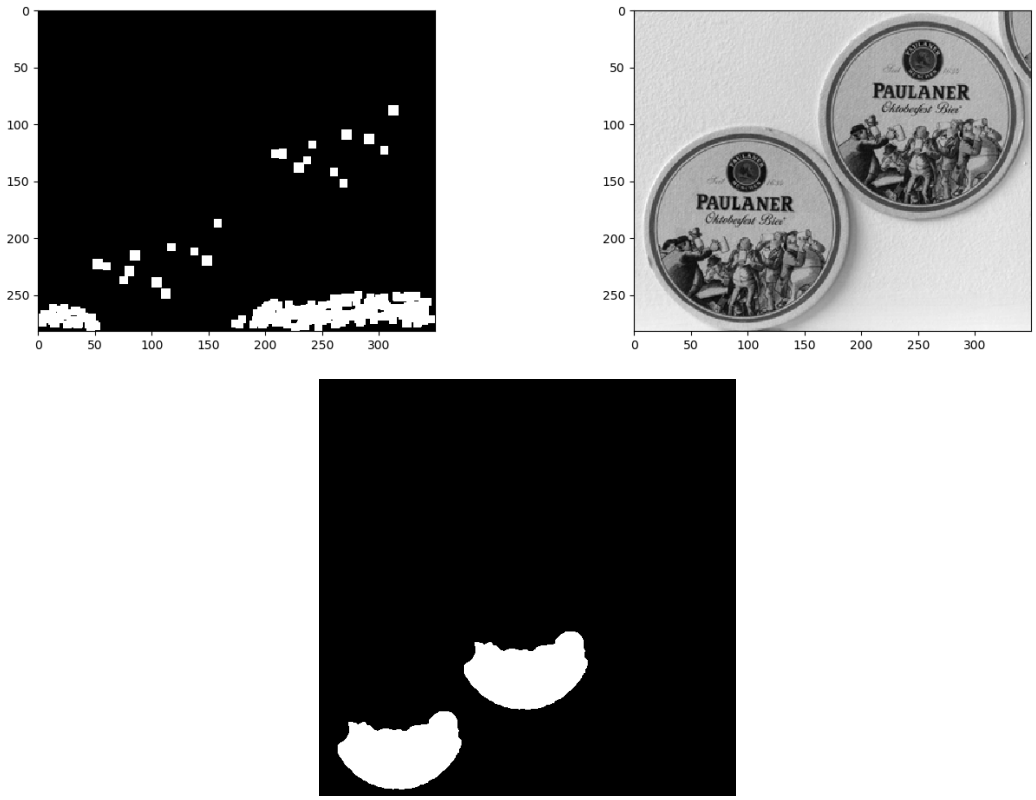

Figura 7.1: *Comparação mapa de detecção obtido (à esquerda-cima), imagem original (à direita cima) e mapa de colagem (centro-baixo) utilizando* PatchMatch Generalizado *em 5 iterações, $K = 8$, bloco de lado 7, limiar de similaridade 10, limiar de distância 20 e deslocamento de bloco na detecção igual à metade do tamanho do bloco truncado)*

Capítulo 8

Conclusões

Apesar de ter implementado corretamente o método *PatchMatch Generalizado*, a análise de detecção possui erros de implementação que não consegui identificar, um possível caminho para prosseguir seria tentar reconstruir todo o processo ou adicionar etapas para a dispersão das correspondências aos vizinhos (visto que foi possível encontrar regiões corretas no mapa de detecção).

Referências Bibliográficas