UNIVERSIDADE FEDERAL DO PAMPA

Douglas Vinicius Ledur Dullius

Representação Paramétrica de Animações Cartoon Utilizando NURBS

Alegrete
2023

Douglas Vinicius Ledur Dullius

Representação Paramétrica de Animações Cartoon Utilizando NURBS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Marcelo Resende Thielo

Alegrete
2023

Douglas Vinicius Ledur Dullius

Representação Paramétrica de Animações Cartoon Utilizando NURBS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em 03 de julho de 2023.
Banca examinadora:

Prof. Dr. Marcelo Resende Thielo
Orientador
UNIPAMPA

Prof. Dr. Bruno Boessio Vizzotto
UNIPAMPA

Prof. Dr. Fabio Paulo Basso
UNIPAMPA

Assinado eletronicamente por **MARCELO RESENDE THIELO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/07/2023, às 02:29, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **BRUNO BOESSIO VIZZOTTO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/07/2023, às 06:41, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **FABIO PAULO BASSO**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 14/07/2023, às 13:12, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1171868** e o código CRC **41F54221**.

Este trabalho é dedicado a minha família,
que me apoiou em todos os momentos de minha vida.

AGRADECIMENTOS

Agradeço primeiramente a minha família, pai, Ademar Luis Dullius, mãe, Rosemeri Ledur e irmã, Jéssica Leticia Ledur Dullius, que sempre estiveram ao meu lado, me apoiando e incentivando incondicionalmente em todos os momentos. Estes que participaram ativamente durante toda a minha formação acadêmica, sempre ajudando com qualquer tipo de necessidade. Devo tudo que tenho a eles e não conseguiria concluir esta etapa de minha vida sem a sua ajuda, portanto, meu maior agradecimento é feito a eles. Agradeço também a meus amigos, Rafael de Oliveira Jarczewski, Uéslei Bervanger Brandt, Gustavo Lopes Tamiosso, Mariana Toledo Costa, Vinicius de Souza da Silva, Vitor Xavier Siqueira e Mágino Antero Corrêa Leguissamo Filho, que me acompanharam durante toda essa fase da minha vida, me ajudando com estudos, conversas e com certeza mudando minha visão de mundo, me tornando uma pessoa melhor. Gostaria de agradecer também a meus professores, que me ensinaram tudo que sei desta área, em especial ao meu orientador Marcelo Resende Thielo, que sempre me ajudou em questões que envolvessem ou não a graduação, além de sempre me incentivar ao aprendizado.

“Abbreviation is a necessary evil and the abbreviator’s business
is to make the best of a job which, though intrinsically bad,
is still better than nothing.“
(Aldous Huxley)

RESUMO

Animações e imagens no estilo *cartoon*, como, por exemplo, as artes no estilo celulóide, são um meio de expressão artística que vêm crescendo cada vez mais. Em função disto, cresce também a sua demanda por novas tecnologias relacionadas. Por exemplo, um grande esforço é demandado dos animadores na criação de animações *cartoon* devido à sua criação ser realizada quadro a quadro. Há a necessidade de meios para a interpolação de novos quadros a partir dos originais, a fim de diminuir esse esforço. Este trabalho visa ter como contribuição principal a criação de uma representação paramétrica de animações cartoon utilizando *non-uniform rational basis spline* (NURBS). Esta criação será independente de taxa de quadros. Além disso, na maioria das vezes irá necessitar de uma menor quantidade de dados para seu armazenamento, devido à baixa complexidade das informações de entrada. Este trabalho visa explorar a propriedade vetorial dessas animações, devido à sua baixa complexidade de cores e linhas de contorno para que, então, se consiga montar um objeto tridimensional através dos pontos de controle que o representem, utilizando NURBS. Neste caso, a terceira dimensão será a representação temporal deste objeto no decorrer da animação, possibilitando assim a interpolação de novos quadros a partir dos originais. Neste trabalho iremos assumir que os pontos de controle que representam o objeto já estão coletados e são necessários para a criação da representação. Os resultados nos mostram que essa apresenta potencial para se tornar uma alternativa viável às redes neurais profundas, que até o presente momento eram utilizadas para a interpolação de novos quadros. Sendo assim, tornar-se-à possível suavizar a animação de entrada com diversos quadros extras, e ainda manter fidelidade à forma original do objeto. Devido à necessidade de recebermos apenas os pontos de controle que representam o objeto e suas *flags* de término de NURBS e quadro para a criação de nossa representação, observamos que a representação proposta conseguiu alcançar 84,23% de compressão dos dados em relação a representação original em video para os casos estudados.

Palavras-chave: Processamento de animações *cartoon*, compressão de animações *cartoon*, representação vetorial de animações *cartoon*, NURBS, representação independente de taxa de quadros.

ABSTRACT

Cartoon-style animations and images, such as celluloid-style art, are a growing medium of artistic expression. As a result, the demand for new related technologies also grows. For example, a lot of effort is required from animators when creating cartoon animations due to their creation being done frame by frame. There is a need for means for interpolating new frames from the originals in order to lessen this effort. This work aims to have as main contribution the creation of a parametric representation of cartoon animations using NURBS. This creation will be frame rate independent. In addition, most of the time a smaller amount of data will be required for its storage, due to the low complexity of the input information. This work aims to explore the vector property of these animations, due to their low complexity of colors and contour lines, so that, then, it is possible to assemble a three-dimensional object through the control points that represent it, using NURBS. In this case, the third dimension will be the temporal representation of this object in the animation progress, thus allowing the interpolation of new frames from the original ones. In this work we will assume that the control points that represent the object are already collected and are necessary for the creation of the representation. The results show us that it has the potential to become a viable alternative to deep neural networks, which until now have been used for the interpolation of new frames. Thus, it will become possible to smooth the incoming animation with several extra frames, and still maintain fidelity to the original shape of the object. Due to the need to receive only the control points that represent the object and its flags term from NURBS and frame for the creation of our representation, noting that the proposed representation reached 84.23% of data emotions in relation to the original video representation for the cases studied.

Key-words: Cartoon animation processing. Cartoon animation compression. Vector representation of cartoon animations, NURBS, frame rate independent representation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D duas dimensões

3D três dimensões

B-spline *basis spline*

BD banco de dados

GMM *gaussian mixture model*

JBIG *joint bi-level image experts group*

JPEG *joint photographic experts group*

JPEG2000 *joint photographic experts group 2000*

LBP *local binary pattern*

NURBS *non-uniform rational basis spline*

PSNR *peak signal-to-noise ratio*

RGB *red green blue*

SIFT *scale-invariant feature transform*

SRKDA *spectral regression-kernel discriminant analysis*

WWW *world wide web*

SUMÁRIO

1 INTRODUÇÃO

Animações *cartoon*, como por exemplo as tradicionais animações em celulóide, consistem em uma forma de arte muito popular. Essas animações alcançam os mais diversos públicos, produzindo uma grande quantidade de material todos os dias. Além disso, a construção dessas animações pode gerar um cenário que demande muito esforço por parte dos desenhistas, visto que essas animações devem ser construídas quadro a quadro. Essa demanda por um maior esforço pode ocasionar em uma menor quantidade de quadros, onde em certos casos, a qualidade da animação pode ser comprometida.

Como alternativa para isto, esforços estão sendo aplicados no desenvolvimento de novas representações independentes de resolução ou taxa de quadros. Como exemplo, temos (BAO, 2019), que utiliza técnicas de aprendizado de máquina, mais especificamente redes neurais profundas para a realização da interpolação de novos quadros a partir dos originais. Porém, a criação de uma representação independente de taxa de quadros, assim como a criação de novos quadros a partir de um conjunto de quadros iniciais está em sua exploração inicial. Assim como (BAO, 2019), as soluções atuais utilizam redes neurais profundas para sua resolução, trazendo consigo os problemas desta abordagem. Dentre eles, podemos citar principalmente a sua natureza de difícil compreensão humana, que pode gerar cenários de erros de difíceis soluções por conta da alta complexidade interpretativa. Além disto, não há soluções específicas para animações no estilo cartoon, que buscam explorar as suas principais características.

O aumento da circulação e transmissão de dados digitais, também pode impactar diretamente a qualidade das animações *cartoon*. Visto que animações mais complexas e com uma maior quantidade de quadros, exigem também, uma maior necessidade de dados em sua representação. Ocasionando assim, em cenários onde pode haver a diminuição da qualidade destas animações visando se adequar aos fluxos de banda nas redes. Existem trabalhos que buscam soluções específicas para este problema em animações *cartoon*. Por exemplo, (RAJ, 2019) e (MAINBERGER, 2011), que buscam desenvolver compressões para imagens no estilo *cartoon*. Além disso, existem diversas técnicas de compressão e representação baseadas em arestas, que podem ser aplicadas de maneira ideal em imagens *cartoon*. Dado que elas apresentam naturalmente uma representação vetorial. Isto se da pelas suas características únicas, como linhas de contorno, coloração simplificada, baixa complexidade e uma relação bem definida entre objetos em primeiro plano e seu respectivo plano de fundo.

Atualmente as animações e imagens *cartoon* não vem sendo muito exploradas. Com poucos trabalhos relacionados a processamento de imagens e animações *cartoon*. Isso faz com que esta área conte com poucas soluções específicas, dependendo muitas vezes de soluções genéricas que podem não gerar os melhores resultados. Com isso, há uma vasta possibilidade de melhorias a serem exploradas.

O objetivo deste trabalho é propor um método de representação vetorial de ani-

mações *cartoon* com perda. Essa representação é independente de taxas de quadros e resolução. Sendo uma alternativa às redes neurais profundas na interpolação de quadros, desenvolvendo uma solução específica para animações *cartoon*. Além disso, essa representação é menos complexa e necessita de menos dados armazenados que as representações comuns de animações. Visto que ela é composta apenas por pontos delimitadores e *flags* que indicam o fim de um quadro e de uma NURBS. Necessitando assim, por muitas vezes, de uma menor quantidade de espaço para armazenamento.

Para a criação desta representação, será considerado que os pontos delimitadores já estão, juntamente com as *flags* que descrevem o final de uma NURBS e de um quadro, em um arquivo. Estes pontos delimitadores podem ser coletados de quadros originais de uma animação, caso o objetivo seja interpolar novos quadros. Mas também há a possibilidade de criar animações novas a partir de pontos delimitadores quaisquer. No caso de serem coletados de uma animação já existente. A animação já deve estar devidamente separada em suas cenas. Onde cada cena trata-se de uma sequência de quadros com apenas um plano de fundo. Com a obtenção destes pontos delimitadores, criamos uma NURBS que será a representação temporal deste objeto. Nessa representação, os eixos x e y serão a representação do objeto em um quadro. Enquanto o eixo z será utilizado para representar as movimentações do objeto na cena. Para isto, utilizaremos da função de NURBS para gerar novos quadros interpolados entre os representados inicialmente no arquivo de entrada, escolhendo coordenadas intermediárias no eixo z.

O restante deste trabalho está dividido da seguinte forma: A seção 2 aborda os temas da fundamentação teórica deste trabalho. A seção 3 apresenta os trabalhos relacionados a representações de imagens *cartoon*, processamento de imagens em *cartoon* e compressão de dados em imagens *cartoon*. Na seção 4 aborda-se o desenvolvimento deste trabalho através da definição das ferramentas a serem utilizadas, de funcionalidades da aplicação e a proposta de modelo para representação de animações *cartoon*. Na seção 5 avalia-se os resultados obtidos. Por fim, na seção 6 apresentamos a conclusão deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção abordam-se os fundamentos teóricos necessários para o entendimento do restante deste trabalho. Os fundamentos introduzidos serão os seguintes: representação computacional de imagens digitais, processamento de imagens digitais, compressão de imagens, Python, curvas paramétricas e NURBS.

2.1 Representação computacional de imagens digitais

Imagens digitais são adquiridas ao discretizar dados detectados, convertendo estes dados em uma forma digital. Isto implica em dois processos, o processo de amostragem e de quantificação. Para uma determinada imagem f, que tem suas coordenadas x e y contínuas e sua amplitude, convertemos esta para sua forma digital através da amostragem de suas coordenadas e quantização de sua amplitude (GONZALEZ, 2007).

Atualmente há duas formas mais populares de representação de imagens digitais, a primeira delas trata-se de imagens rasterizadas e a segunda de imagens vetoriais. Portanto discutiremos estas nos tópicos a seguir.

2.1.1 Imagens rasterizadas

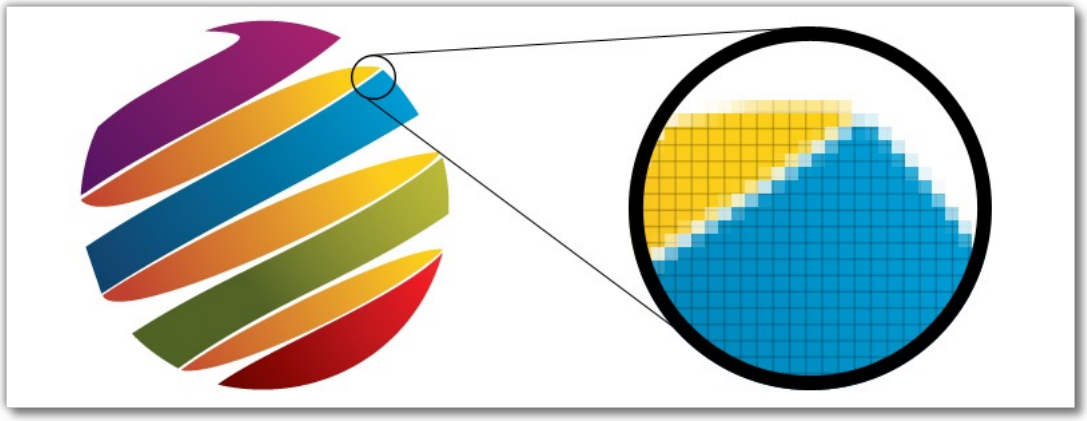
Esta é a representação matricial das imagens. Neste formato a imagem passa a ser descrita por um espaço bidimensional composto por um conjunto de células denominadas *pixels*. Estes representam a menor unidade presente na imagem matricial. As imagens deste formato são formadas através da utilização destes *pixels*, seja através do formato *red green blue* (RGB), onde cada pixel contém sua devida coloração ou através de *bitmap*, trabalhando com definições das áreas que devem ser coloridas ou não. Essa é a representação mais comum de imagens e é empregada na formação de imagens nas memórias e telas de computadores. Também é utilizada na maioria dos dispositivos de saída gráfica, como por exemplo impressoras. Normalmente quando pensamos em definições de imagens associamos a esta representação (AZEVEDO, 2022).

Como podemos observar pela figura 1, esta representação é dependente de resolução, isto é, ao aproximarmos nossa observação de um certo ponto na imagem podemos notar a diminuição na qualidade da imagem. A queda de qualidade pode ser observada quando estamos trabalhando com imagens com baixas quantidades de *pixels*, deixando-os perceptíveis ao olho humano. Entretanto, trata-se de uma representação simples e de constante uso de memória, isto é, não havendo grandes variações para a representação de imagens de mesmo tamanho.

2.1.2 Imagens vetoriais

Na representação vetorial das imagens, utiliza-se como elemento base pontos, linhas, curvas, superfícies tridimensionais ou sólidos que descrevem o elemento em questão.

Figura 1 – Exemplo de imagem rasterizada.



Fonte: (HOLINEY, 2019)

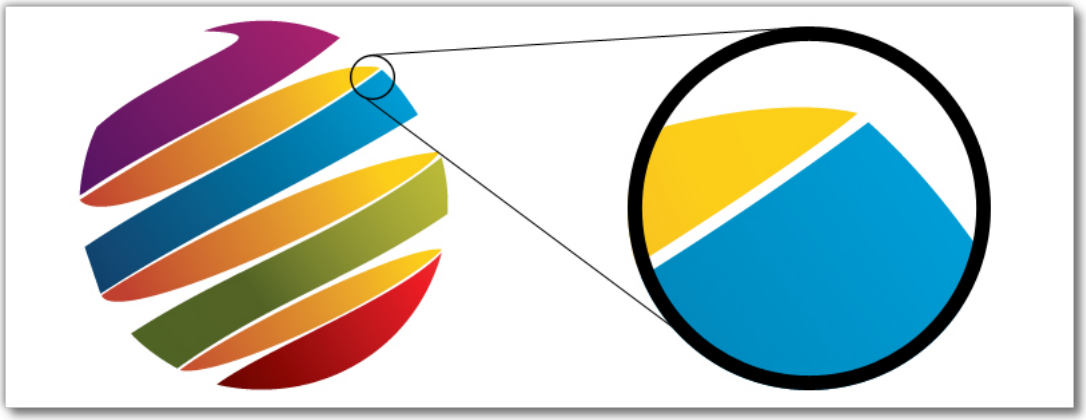
Os elementos são denominados primitivas vetoriais e estão associados a dados que definem sua geometria. Estes dados são pontos de controle e um conjunto de atributos aos quais definem sua aparência. Podemos pensar na representação de um ponto e uma linha reta para uma melhor definição. Um ponto é associado a uma coordenada de posição, o qual é sua geometria, e uma linha reta pode ter sua geometria definida pelos pontos que representam suas extremidades. A coloração do ponto será o que define como este aparecerá na tela, ou seja, seu atributo. Ao mesmo que os atributos de uma linha reta podem ser sua coloração, espessura e se a mesma será representada como uma linha tracejada ou pontilhada. Este formato é principalmente empregado em modelagem de objetos sintéticos (AZEVEDO, 2022).

Como podemos observar através da figura 2, o formato vetorial não apresenta dependência de resolução, devido a sua construção baseada em seus elementos básicos e não em *pixels*. Esta representação consegue se adequar a novas resoluções quando, por exemplo, aproximamos nosso observador de um determinado ponto. Entretanto, trata-se de uma representação de uma maior complexidade envolta, além de não ser o caso ideal em alguns cenários que apresentam imagens com alta variação de cores.

2.2 Processamento de imagens digitais

Segundo (GONZALEZ, 2007), processamento de imagens digitais refere-se a processar imagens digitais através de um computador digital. Neste caso, a imagem digital é composta por um finito número de elementos particulares, aos quais contém valores e localizações. Esses elementos contidos nas imagens são conhecidos como *pixels*. O processamento de imagens digitais opera em casos onde a percepção humana não é capaz devido a limitações físicas. Este cobre quase todo o espectro eletromagnético, operando

Figura 2 – Exemplo de imagem vetorial.



Fonte: (HOLINEY, 2019)

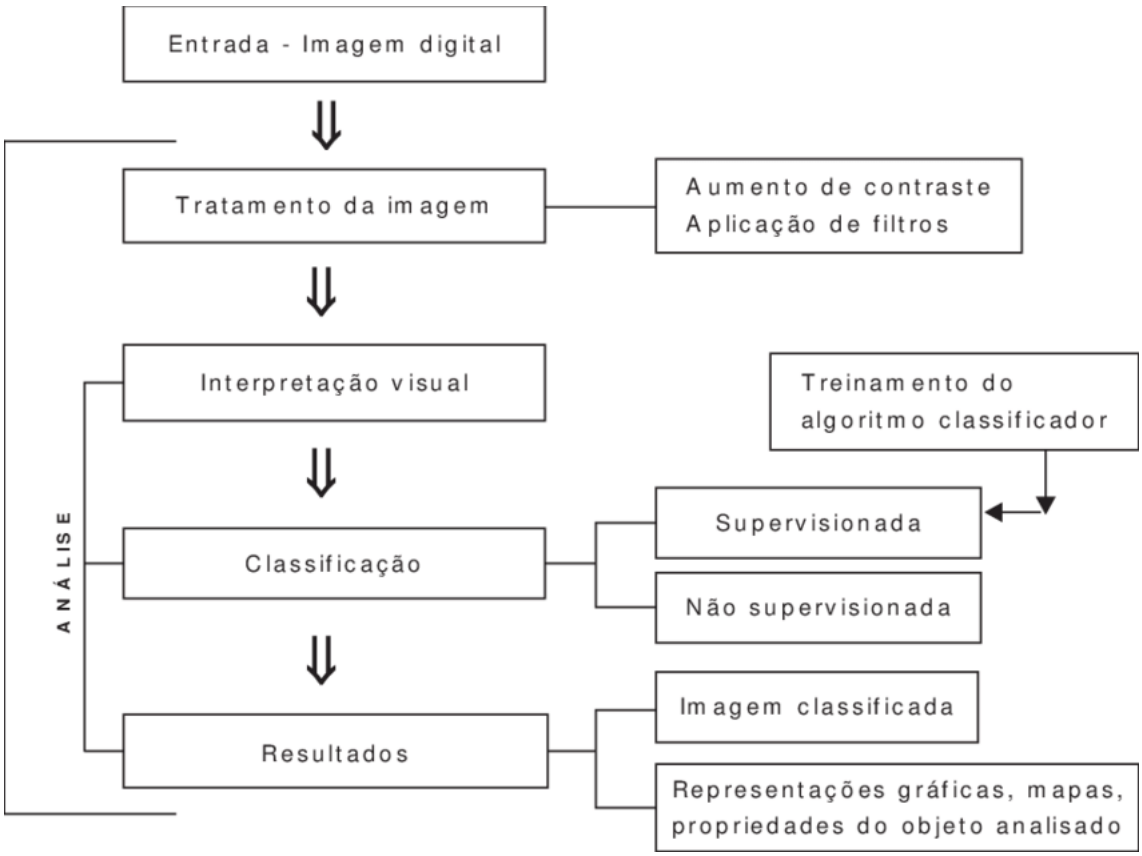
em imagens geradas por origens que humanos não são capazes de associar com imagens. Fazendo com que o processamento de imagens envolva uma variedade de aplicações.

Na literatura atual, não há um consenso bem definido sobre a linha divisória entre processamento de imagens e visão computacional. Embora haja algumas tentativas de divisões destas áreas, nenhuma das apresentadas nos resulta em divisões satisfatórias. Entretanto, (GONZALEZ, 2007) considera processamento de imagem processos de baixo e médio nível de processamento, atribuindo processos de alto nível de processamento a visão computacional. No nível baixo encontramos operações primitivas, como por exemplo pré-processamento de imagens para reduzir ruídos, aprimoramentos de contraste e cortes de imagens. Neste nível recebe-se imagens como entrada e gera-se imagens como saída. Enquanto isso, no nível médio há o envolvimento de tarefas de segmentação, descrição do objeto em questão e classificação de objetos individuais. O nível médio normalmente recebe imagens como entrada e gera como resultado atributos extraídos dessas imagens, como arestas, contornos entre outros. A figura 3 busca demonstrar visualmente possíveis passos no processamento de imagens.

Segundo (GONZALEZ, 2007), o processamento de imagens também pode ser dividido em alguns processos, os quais são:

- Aquisição da imagem.
- Aprimoramento e filtragem da imagem.
- Restauração da imagem.
- Processo de coloração da imagem.
- Processo de multi-resolução e ondas.

Figura 3 – Passos do processamento de imagens digitais.



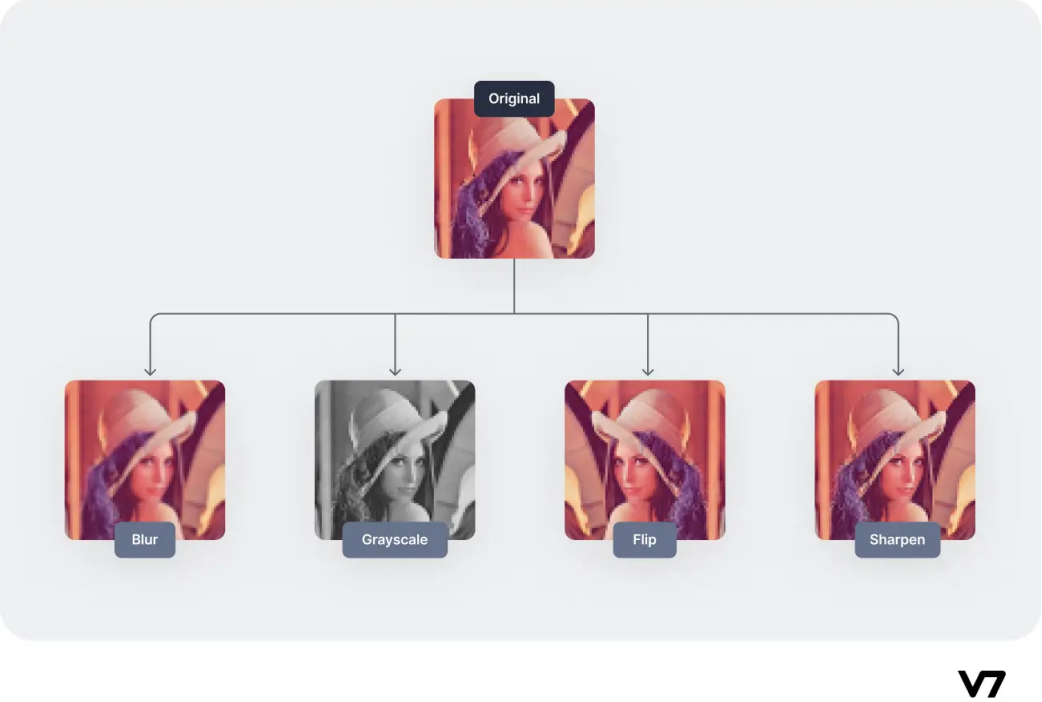
Fonte: (GIOVANINI, 2019)

- Compressão.
- Processo morfológico.
- Segmentação.
- Representação e descrição.
- Reconhecimento de objeto.

Atualmente existem poucas áreas que não são impactadas de alguma forma pelo processamento de imagens digitais. As mais diversas áreas aplicam processamento de imagens digitais de inúmeras formas. Para melhor demonstrar a sua aplicabilidade, podemos avaliar algumas possíveis origens de imagens, sendo elas: Imagens Raio-X, banda ultravioleta, bandas visíveis e infravermelhas, banda de microondas, banda de rádio (GONZALEZ, 2007). Com essas possíveis origens pode-se ter uma melhor compreensão de sua utilização, dado que diversas aplicações atuais utilizam-se de imagens dentre as origens descritas.

A figura 4 busca demonstrar algumas possíveis técnicas de processamento de imagens, com exemplificações gráficas destas para uma melhor compreensão.

Figura 4 – Exemplos de técnicas de processamento de imagens.



Fonte: (KUNDU, 2023)

2.3 Compressão de imagens

A compressão de imagens digitais trata-se de uma área que visa através da utilização de algoritmos diminuir a quantidade de dados necessários para representar determinada imagem. (GONZALEZ, 2007) descreve-a como a arte e ciência da redução dos dados necessários para representação de uma imagem. (GONZALEZ, 2007) também aponta que esta trata-se de uma das tecnologias mais utilizadas atualmente no campo das técnicas de processamento de imagem. Se observarmos mais atentamente, a compressão de descompressão de dados está nos mais diversos lugares, diariamente utilizamos ela constantemente, embora seja uma técnica geralmente invisível aos usuários. Câmeras digitais e transmissões ao vivo de vídeos são exemplos de sua utilização, onde aqui podemos notar suas aplicações comuns, diminuição de dados trafegando em redes e diminuição de memória necessária para armazenamento.

De acordo com (GONZALEZ, 2007) a compressão de dados é o processo de reduzir a quantidade de dados para representar uma mesma quantidade de informações. Nesta

frase, trata-se informação e dados de maneira diferentes, onde os dados são a forma de se transmitir determinada informação. No processo de compressão busca-se eliminar o que é denominado de redundância de dados, que seriam os dados repetidos ou não necessários, ou seja, irrelevantes para aquela determinada informação. Segundo (GONZALEZ, 2007) imagens duas dimensões (2D) podem conter três tipos principais de redundância, sendo elas: redundância de código, redundância espacial ou temporal e informações irrelevantes. O primeiro tipo está relacionado a símbolos utilizados para representar um corpo ou conjunto de eventos, no processamento de imagens pode-se utilizar mais *bits* que os necessários para representar intensidades, levando a esta redundância. O segundo tipo está relacionado a correlação ou dependência de um *pixel* em relação a sua vizinhança, gerando uma redundância espacial, normalmente essa correlação pode ser devidamente localizada e se possível removida. Vídeos também podem contar com redundância temporal, onde, similarmente a redundância espacial, há uma correlação temporal entre os *pixels* que pode ser eliminada para evitar duplicidade de dados. O terceiro tipo de redundância está em informações desnecessárias, onde uma imagem pode conter conteúdo não utilizado.

A compressão de dados pode ser classificada em compressão de dados com perda e compressão de dados sem perda, vamos discutir a seguir cada um destes casos mais profundamente.

2.3.1 Compressão de dados sem perda

De acordo com (TAUBMAN, 2001), o principal objetivo da compressão sem perda é minimizar o número de *bits* necessários para representar a imagem original sem perder nenhuma informação. Utilizando compressão de dados sem perda espera-se que para determinada imagem x de entrada e sua imagem resultante x' após o processo de compressão e descompressão respectivamente, x e x' devem conter as mesmas informações. Ou seja, x deve ser reconstruída perfeitamente durante a etapa de descompressão. Nota-se na figura 5 a exemplificação desta compressão, onde a quantidade de informações se mantém após a descompressão.

Esse tipo de compressão é demandado em aplicações que carecem de uma alta precisão e, portanto, não podem trabalhar com erros de imagens, como por exemplo em uma paleta de cores. Esta compressão também é empregada em casos de compressões e descompressões múltiplas, para evitar uma maior perda de qualidade.

2.3.2 Compressão de dados com perda

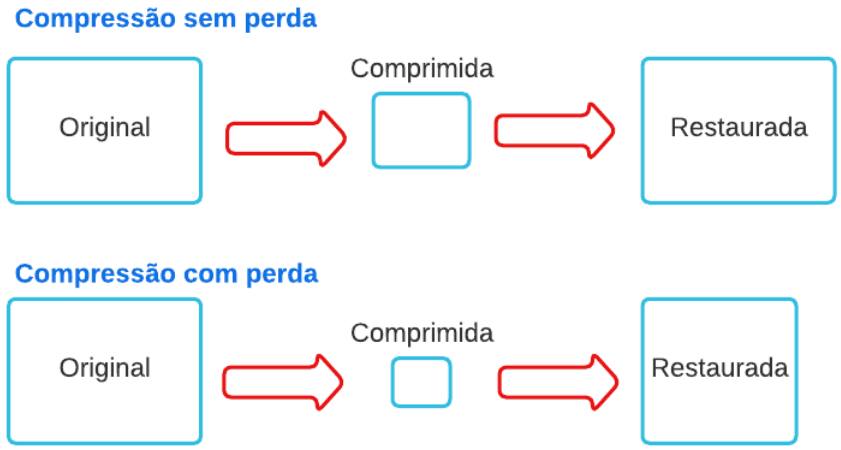
A compressão de dados com perda visa, como seu principal objetivo, representar com o menor número de dados uma determinada informação, aceitando entretanto uma perda desta no processo. Para isso, adota-se diversas abordagens não permitidas na compressão sem perda, como aproximações de dados, o que permite diversas representações

com menores números de *bits*. Podemos observar um exemplo visual desta compressão na figura 5.

De acordo com (TAUBMAN, 2001), há alguns pontos que podemos considerar quando pensamos em compressão com perda, sendo elas: 1) Perdas significativas podem ser toleradas pelo sistema visual humano, sem que isso interfira em nossas percepções. 2) Na maioria dos casos, as entradas digitais recebidas pelos algoritmos de compressão já são por elas mesmas, representações imperfeitas de cenas do mundo real. 3) Compressão sem perda não é usualmente capaz de adquirir grandes níveis de compressão, que são necessários para algumas aplicações.

Devido ao seu alto nível de compressão, técnicas de compressão com perda são utilizadas em diversas aplicações que contêm espaço de armazenamento limitado, que não exigem altos níveis de precisão de imagens ou também em transmissões de dados.

Figura 5 – Representação gráfica de compressão sem e com perda de dados respectivamente.



Fonte: Autor

2.4 Curvas paramétricas

Equações implícitas e funções paramétricas são os dois métodos mais comuns de representação de curvas. Enquanto equações implícitas representam curvas no plano cartesiano com eixos x e y na forma:

$$f(x, y) = 0 \tag{2.1}$$

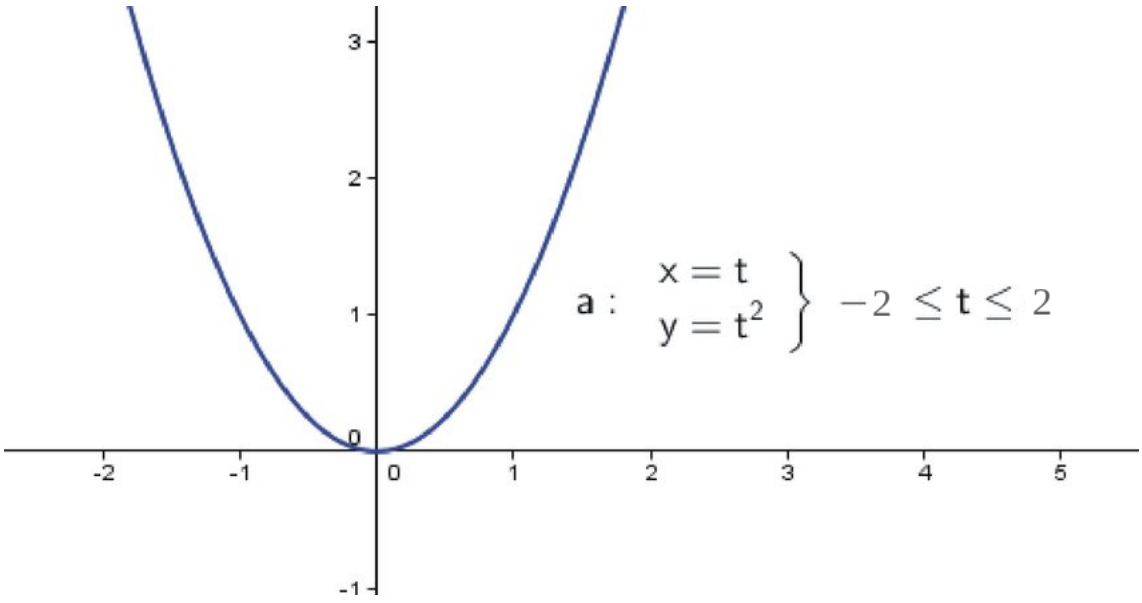
. Isto ocasiona em uma dependência implícita entre as coordenadas de x e y, funções paramétricas apresentaram pontos da curva separados para cada uma de suas coordenadas, como uma função explícita de parâmetros independentes (PIEGL, 1996).

Em curvas paramétricas, representa-se cada coordenada de um determinado ponto na curva como uma função de parâmetro único. Para exemplificar, dado uma curva 2D que se utiliza de um parâmetro t, as coordenadas cartesianas de x e y seriam respectivamente $x(t)$ e $y(t)$. Isto nos leva a seguinte representação de um ponto P em uma curva paramétrica (AZEVEDO, 2022):

$$P(t) = (x(t), y(t)) \tag{2.2}$$

. A figura 6 exemplifica uma curva paramétrica, onde se tem o intervalo do parâmetro t definido e x, y com suas respectivas funções em relação a t.

Figura 6 – Curva paramétrica de duas dimensões.



Fonte: O Autor

Curvas paramétricas proporcionam diversas vantagens, como por exemplo, a maior facilidade de representação de comprimentos constantes, facilidade no cálculo de derivadas em relação ao parâmetro único e também facilidade de cálculo de área. Isto se dá pela sua obtenção através do cálculo das integrais de x e y. Curvas paramétricas são especificadas por um único parâmetro e tem suas extremidades e tamanho definidas através de valores fixos, dados pela variação do parâmetro único, normalmente normalizados entre 0 e 1. Além disto, não há dependência de sistemas de coordenadas, permitindo a esta uma fácil manipulação através de transformações geométricas (AZEVEDO, 2022).

Curvas paramétricas podem facilmente representar curvas espaciais, não apresentando maiores dificuldades se comparado a representações curvas no plano. Para que consiga-se essa representação tridimensional basta inserirmos uma nova coordenada que pode ser dada pela função:

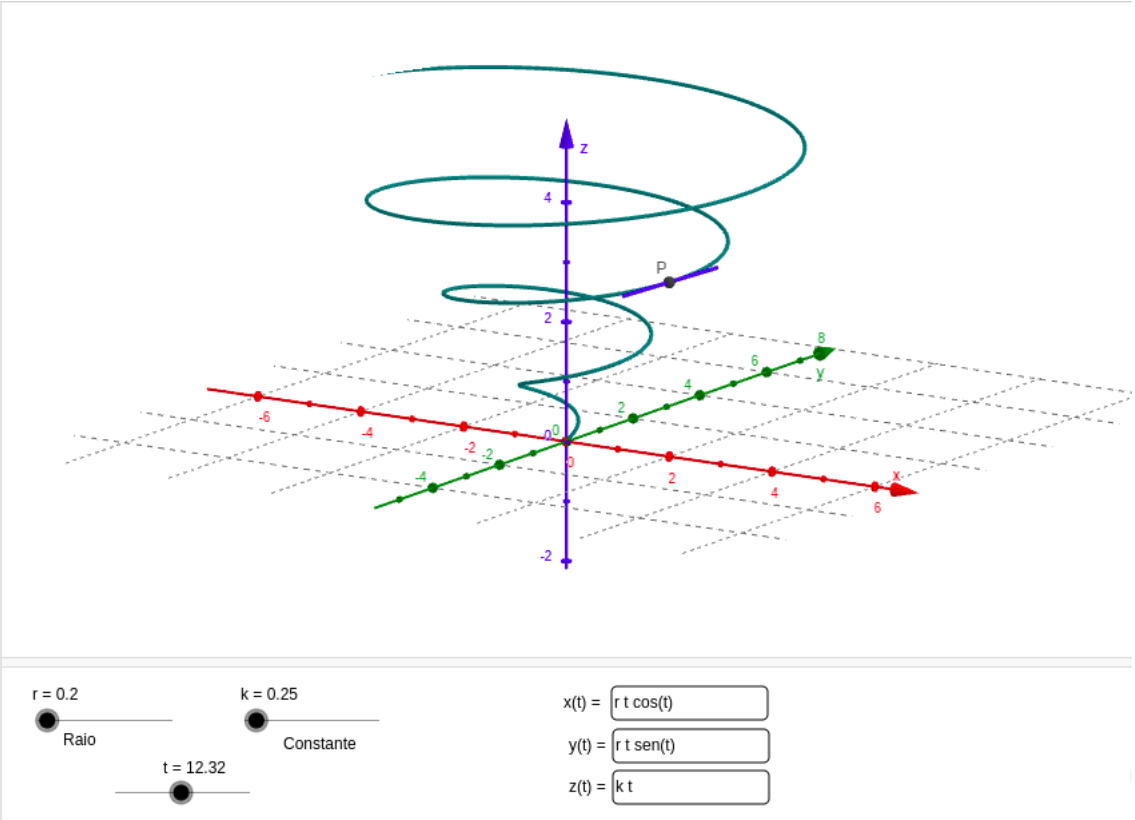
$$z = z(t) \tag{2.3}$$

. Como podemos observar, segue-se um mesmo padrão para essas representações e, portanto, um ponto da curva espacial seria dado por:

$$P(t) = (x(t), y(t), z(t)) \tag{2.4}$$

. Onde t representa o parâmetro único (AZEVEDO, 2022). A figura 7 exemplifica graficamente uma curva paramétrica espacial.

Figura 7 – Curva paramétrica de três dimensões.



Fonte: (IGM,)

2.5 Non-Uniform Rational Basis Splines Surfaces

O termo NURBS surge como uma abreviação de *Non-Uniform Rational Basis Splines Surfaces*. Este trata-se de um modelo matemático comumente utilizado em computação gráfica para a representação de curvas e superfícies (AZEVEDO, 2022). Para uma melhor compreensão de seu significado, podemos avaliar cada parte desse termo separadamente da seguinte forma.

- **Basis Splines:** *splines* são curvas geradas a partir de pontos de controle e uma expressão matemática. As *splines* provocam alterações em toda a curva a cada

modificação em qualquer um de seus pontos. Como esta representação não trata-se da mais adequada para curvas interativas, *basis spline* (B-spline) surge como um variação da spline, esta possui controle local, ou seja, alterações em pontos de controle serão propagados apenas para sua vizinhança próxima. Curvas B-spline não passam necessariamente pelos seus pontos de controle e podem ser geradas com uma quantidade qualquer de pontos de controle e grau de polinômio (AZEVEDO, 2022). A figura 8 exemplifica visualmente uma B-spline gerada a partir de sete pontos de controle.

- **Non-Uniform:** Significa que a influência da extensão de um controle de vértice não precisa ser a intervalos iguais do parâmetro único. Ou seja, uma determinada forma definida entre dois pontos pode ser maior ou menor se comparado a outras formas dessa mesma superfície. Isto trata-se de uma vantagem na representação de superfícies não regulares (AZEVEDO, 2022).

- **Rational:** Analogamente aos números racionais, pode ser dado como a razão entre dois polinômios. Sua importância se dá pela propriedade de invariância a transformações de projeção. A invariância a perspectiva tem como base coordenadas homogêneas, estes tratam-se de pontos no espaço três dimensões (3D) que são utilizados para a obtenção dos pontos de controle no espaço 3D, essas coordenadas homogêneas também são denominadas pesos (AZEVEDO, 2022).

Superfícies NURBS foram criadas exclusivamente para a modelagem computacional 3D, ou seja, não existem no mundo do desenho tradicional. Essas superfícies são geradas matematicamente e proveem flexibilidade para geração de uma variedade de formas livres, englobando todas as outras formas de representação. Isso faz com que as mesmas sejam casos particulares desta, bastando seguir restrições para representar o conjunto desejado (AZEVEDO, 2022). Pode-se observar um exemplo gráfico de NURBS na figura 9, onde utiliza-se de certos pontos de controle para a geração da superfície desejada no ambiente 3D simulado.

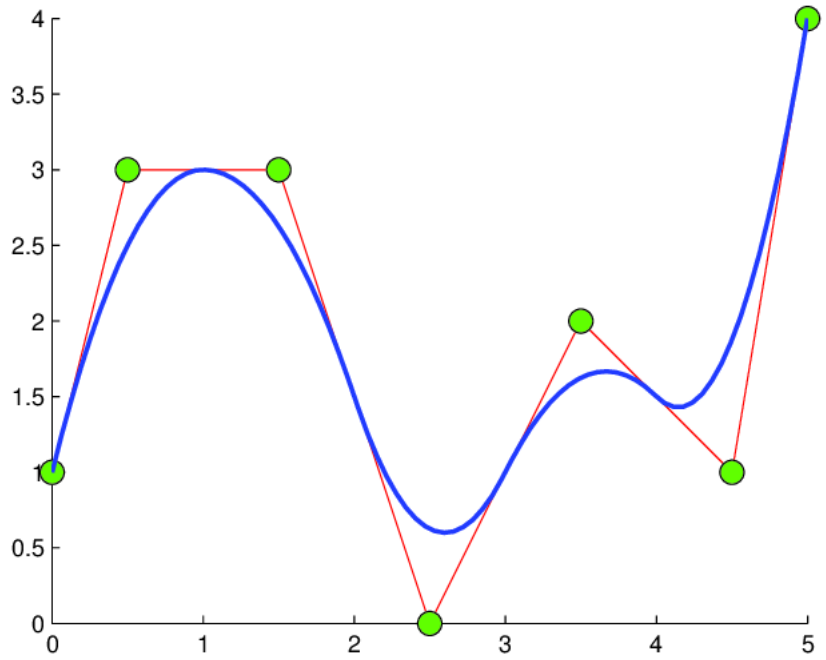
Uma determinada superfície NURBS de grau qualquer p pode ser definida através da seguinte equação:

$$S(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{ip}(u)N_{jq}(v)w_{ij}P_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{ip}(u)N_{jq}(v)w_{ij}}, 0 \leq u,v \leq 1 \tag{2.5}$$

Nesta equação, podemos observar diversos parâmetros que são necessários para a equação de NURBS. A seguir se encontra a definição de cada um desses elementos de acordo com (PIEGL, 1996).

- u e v representam o espaço paramétrico nas duas direções que tipicamente compõem uma superfície NURBS. Normalmente são normalizados para valores de 0 a 1.

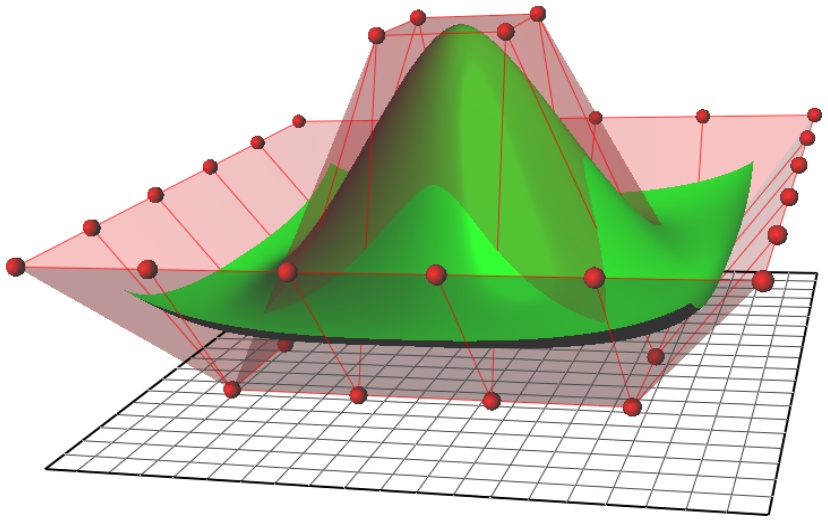
Figura 8 – Representação gráfica de uma curva B-spline.



Fonte: (HOANG, 2015)

- n e m representam o número de pontos de controle nas direções u e v respectivamente.
- i e j são os índices que serão utilizados para iterar nos pontos de controle da NURBS. i irá variar de 0 até n enquanto j de 0 a m.
- p e q representam os graus da superfície NURBS nas direções u e v, respectivamente. Esses graus determinam a ordem polinomial das funções bases utilizadas na interpolação dos pontos de controle.
- $N_{ip}(u)$ e $N_{jq}(v)$ são as funções base associadas com o i-ésimo ponto de controle na direção u e o j-ésimo ponto de controle na direção v, respectivamente.
- P_{ij} representa os pontos de controle da superfície em questão. É representada pelas suas coordenadas no espaço 3D.
- w_{ij} representa os pesos associados a cada um dos pontos de controle da superfície. Esses pesos determinam a influência de cada ponto de controle na superfície. Quanto maior este valor, mais influência o ponto de controle em questão tem na superfície.
- $S(u, v)$ representa o ponto na superfície NURBS para os valores dos parâmetros u e v. Trata-se do ponto que se deseja calcular na superfície.

Figura 9 – Representação gráfica de uma NURBS.



Fonte: (ERSTELLT, 2007)

3 TRABALHOS RELACIONADOS

Nesta seção será descrito o processo de pesquisa de trabalhos relacionados, bem como será apresentado um breve resumo de cada um destes para um melhor entendimento.

3.1 Protocolos de pesquisa

O processo de pesquisa destes trabalhos deu-se através do motor de busca Google Scholar, o qual é amplamente utilizado para estes fins. Este motor de busca foi utilizado para a seleção de diversos artigos para análise, os quais foram coletados ao buscar pelos seguintes fragmentos de frases:

- (Processamento OU vetorização OU compressão) E (imagens OU animações) E (*cartoon*)
- (Processing OR vectorization OR compression) AND (images OR animations) AND (*cartoon*)

Foram então filtrados os resultados obtidos através destas buscas aplicando um controle de inclusão ou exclusão baseado nas seguintes condições:

- Inclusão
  - Trabalhos que visam apresentar uma compressão específica ou direcionada para imagens *cartoon*.
  - Trabalhos ao quais desenvolvem algum tipo de processamento direcionado a imagens ou animações *cartoon*.
  - Trabalhos que desenvolvem sistemas de vetorização de imagens ou animações *cartoon*.
  - Trabalhos que visam avaliar o atual estado da arte na área de processamento de imagens *cartoon*.
- Exclusão
  - Trabalhos não disponíveis para download.
  - Trabalhos não disponíveis em português ou inglês.

Foram analisados brevemente todos os artigos que satisfizeram as restrições apresentadas anteriormente. Então selecionou-se os dez que melhor se enquadram no tema proposto por este trabalho.

3.2 Resultados da pesquisa

Selecionamos dez trabalhos após a aplicação dos critérios de inclusão e exclusão. Dentre estes trabalhos, dois abordam assuntos de processamento de animações *cartoon*, três de processamento de imagens *cartoon*, dois compressão de imagens *cartoon* e outros três sobre vetorização de imagens *cartoon*. A seguir, estes trabalhos serão brevemente descritos na subseção de seu assunto abordado.

3.2.1 Processamento de animações *cartoon*

No trabalho proposto por (KHAN, 2018), o autor apresenta um classificador de vídeos em mídias sociais, especialmente relacionadas a *cartoons* animados com comportamentos violentos e não violentos. Com isso, espera-se consiguir filtrar conteúdos não desejados de violência para crianças. Este trabalho tem como seu principal objetivo extrair informações gerais de uma imagem sem nenhuma consulta específica. O autor comenta que o principal desafio encontrado foi a ideia de fazer com que um filtro saiba que um vídeo em particular no estilo *cartoon* contém material violento. Utilizando processos de filtragem de quadros chaves para o classificador, e aplicando *scale-invariant feature transform* (SIFT), *Fisher Vector* e *gaussian mixture model* (GMM) faz-se a extração de features das imagens selecionadas e o seu agrupamento por proximidade. Após, o autor emprega o uso de *spectral regression-kernel discriminant analysis* (SRKDA) para a classificação destes vídeos. A validação deste classificador dá-se através da coleta de vídeos oriundos de diferentes origens. Foram escolhidos 100 *clips*, onde 52 continham conteúdo violento e o restante não contendo violência. Deve-se ressaltar que como o conceito de violência é bastante subjetivo, o autor os retirou de animes que são considerados muito violentos para uma criança de 8 anos assistir. Nos testes realizados pelo autor o classificador mostrou-se superior a diversos outros classificadores testados, conseguindo ter uma acurácia de até 97%.

No trabalho proposto por (ZHANG, 2009), o autor apresenta um sistema de vetorização 2D para animações *cartoon* no formato *raster*. O objetivo deste trabalho é gerar animações livres de *flicker*, de fácil edição e que necessitam de menos espaço de armazenamento. Para isto, o autor utiliza-se de algoritmos de detecção de linhas externas para cada um dos quadros, gerando uma máscara. Após, emprega-se o método de *trapped-ball* proposto pelo autor, o qual auxiliado pela máscara gerada faz a segmentação do objeto em questão. Então para garantir coerência temporal reconstrói-se a imagem de *background*. Por fim, faz-se a vetorização dos objetos chaves foreground e do *background*, e preenche seus interiores aplicando modelos de cores. O autor valida seu trabalho através de comparações a trabalhos relacionados com propostas similares. Este obtém bons resultados, onde há a produção de imagens com uma alta fidelidade a original, igualmente a trabalhos relacionados, porém sem diversas restrições impostas por estes. Além disto, mesmo para

casos fora do escopo principal do artigo, isto é, para imagens que fogem do estilo *cartoon*, visualizou-se uma alta precisão e fidelidade. Por fim, o autor comenta que em cenários específicos como pinturas aquarelas, *cartoon* com altos detalhamentos como efeitos de luz ofuscante ou texturas complexas o método desenvolvido não seja adequado.

3.2.2 Processamento de imagens *cartoon*

No trabalho proposto por (TAKAYAMA, 2012), o autor apresenta um método para detecção e reconhecimento de faces em personagens *cartoon*. Este método pode ser utilizado em diversas aplicações, como por exemplo, busca e classificação automática de personagens. Os principais objetivos deste trabalho são a detecção facial de personagens *cartoon* com boa acurácia e a extração de recursos de faces e busca de personagens através do reconhecimento facial. Para a detecção de faces inicia-se buscando a coloração da pele do personagem e a extração das linhas de contorno para que se consiga gerar a segmentação do personagem. Após, empregando os métodos para detecção do contorno da mandíbula do personagem e de simetria facial, chega-se a imagem de resultado. Para o reconhecimento facial extrai-se a cor da face do personagem, a coloração e a sua quantidade de cabelos. Após isso, determina-se um vetor de recursos do personagem utilizando-se destes três elementos para então buscar por imagens com vetores de recursos com alta similaridade.

Para a avaliação dos experimentos, foi utilizado um banco de dados (BD) com 300 imagens de personagens, 100 personagens continham apenas uma imagem e 80 personagens continham 2 ou mais imagens. O autor nos mostra que na detecção de faces há bons resultados, onde o método proposto se sai melhor que outros propostos anteriormente, atingindo um grande número de acertos no reconhecimento de faces. Entretanto para a validação do método de reconhecimento facial foram executados dois experimentos. O primeiro buscando por personagens com apenas uma imagem no BD. Enquanto o segundo selecionava os personagens que continham duas ou mais imagens. A precisão do primeiro teste foi de 71%, enquanto no segundo foi de aproximadamente 50%.

No trabalho proposto por (SýKORA, 2004), o autor apresenta uma nova técnica de coloração por exemplo na qual combina segmentação de imagem, *patch-based sampling* e relaxamento probabilístico. Seu principal objetivo com este trabalho é a automatização da coloração de *cartoons* preto e branco que já possuem um modelo de cor construído. A técnica tem início com a seleção determinista de pontos chaves do objeto em questão. Estes são usados para a identificação das regiões do objeto. Após isto, busca-se encontrar correspondências estruturais entre a imagem de exemplo e a imagem alvo, fazendo-se uso de *backward mapping*. Com essas correspondências e suas cores, verifica-se a coloração mais presente em cada uma das regiões e então colore-a desta cor. Para evitar erros de ambiguidade, utiliza-se também o relaxamento probabilístico para verificação de vizinhanças. O autor avaliou os resultados a partir da coloração de um *cartoon* antigo

nomeado "O loupežníku Rumcajsovi". Com isto verificou que há uma grande dependência do número de estruturas identificadas pela técnica. Porém foram alcançados resultados satisfatórios, conseguindo colorir a maior parte do *cartoon* a partir do exemplo inicial, eliminando esforços manuais.

No trabalho proposto por (ZHAO, 2022), o autor apresenta uma revisão dos métodos utilizados para processamento de imagens *cartoon* 2D, com foco em abordagens recentes baseadas em *deep-learning*. O objetivo deste trabalho é auxiliar futuras pesquisas na área de processamento de imagens *cartoon*. Para isso fez-se a revisão literária dos algoritmos de processamento de imagens *cartoon*, avaliando suas semelhanças, diferenças e diferentes aplicações. Além disto, coletou-se os BD relacionados a estes e aplicou-se testes para validação dos algoritmos propostos. O trabalho aborda o estado da arte em diversas ramificações do processamento de imagens *cartoon*. Dentre elas estão a avaliação de qualidade de imagem em *cartoon*, a colorização *cartoon*, a detecção em *cartoon*, a compressão de imagens *cartoon* entre outros. Através de testes desenvolvidos pelo autor, percebeu-se que muitas abordagens são focadas em *dataset* específicos. Além disto, a maioria não se tratava de aplicações *open source*, o que ocasiona em dificuldades comparativas. O autor observou também que em muitos ramos ainda há a tentativa de aplicação de abordagens genéricas em imagens *cartoon* ao invés de abordagens específicas que consigam se beneficiar de exclusividades destas imagens. Isto leva o autor a conclusão de que ainda há muito espaço para pesquisa nesta área, visto que ainda há muito espaço para criação de *cartoons* de alta qualidade, de processamento rápido e mais acessível.

3.2.3 Compressão de imagens *cartoon*

No trabalho proposto por (RAJ, 2019), o autor apresenta um esquema de compressão de imagens *cartoons* baseado em arestas. O trabalho tem como seu principal objetivo a eliminação de redundâncias sem afetar a integridade da imagem. Para isto, inicialmente há o mapeamento das arestas aplicando o detector de arestas baseado em zero-cruzamentos, então faz-se a compressão das arestas utilizando o método de *turtle edge* proposto pelo autor. Este utiliza uma representação simplificada de arestas proposta pelo autor. Após, faz-se a compressão das cores, onde se é armazenado a cor relacionada a cada uma das regiões da imagem, bem como um pixel aleatório dentro desta. O autor efetuou a validação de seu esquema proposto através da sua execução em 10 imagens oriundas de *world wide web* (WWW), selecionadas aleatoriamente. Então o autor analisa os resultados obtidos em comparação a diversas outras abordagens. Os resultados mostram que o esquema de compressão proposto obteve bons resultados em relação ao espaço de armazenamento necessário para representar a imagem e a *peak signal-to-noise ratio* (PSNR).

No trabalho proposto por (MAINBERGER, 2011), o autor apresenta um método de compressão com perdas baseada em arestas para imagens *cartoon*. O principal objetivo

deste trabalho é um *codec* simples com perdas, capaz de codificar e decodificar imagens *cartoon* em tempo real e com alta qualidade de reconstrução. Para isto, inicia-se detectando as arestas fazendo uso do método denominado *Marr-Hildreth edge detector*. Então é realizada a compressão das localizações e valores dos *pixels* de contorno, utilizando *joint bi-level image experts group* (JBIG) e um método proposto pelo autor respectivamente. Ao final deste processo termina-se com duas partes codificadas, as quais juntamente com um cabeçalho dão origem a um novo arquivo. Este poderá passar pela decodificação aplicando processos reversos aos de codificação juntamente com difusão homogênea para interpolação, aplicada na reconstrução dos pixels. Por fim, para a validação do trabalho, o autor comparou-o a sua antiga versão e também aos métodos de compressão de imagem com perda bem estabelecidos no mercado *joint photographic experts group* (JPEG) e *joint photographic experts group 2000* (JPEG2000). A comparação quantitativa deu-se por meio de PSNR e taxa de compressão. O autor avalia que o trabalho atual saiu-se tão bem quanto a sua antiga versão, porém, apresentando um tempo de compressão significativamente melhor. Enquanto se comparado a JPEG e JPEG2000, o método proposto saiu-se melhor em taxa de compressão e PSNR se comparado ao primeiro, e, se comparado ao segundo, apresentou uma mesma taxa de compressão porém resultados superiores se observarmos o PSNR. O autor observa que estes valores são atingidos quando aplicados a imagens no estilo *cartoon*, mas que o método proposto não trata-se do mais adequado para imagens com muitas texturas.

3.2.4 Vetorização de imagens *cartoon*

No trabalho proposto por (ZOU, 2001), o autor apresenta um método de esqueletização não baseada em *pixels* para a vetorização de imagens *cartoon* como ferramenta de auxílio para desenhistas. Seus principais objetivos com este trabalho são a geração de imagens *cartoons* com uma alta eficiência e qualidade de imagens, removendo os possíveis artefatos. Para isto, utiliza-se inicialmente a *Triangulação de Delaunay*, para que se consiga gerar triângulos não sobrepostos com os pontos que representam o objeto em questão. Então o esqueleto do objeto é gerado a partir do esqueleto desses triângulos. Após a esqueletização aplica-se o algoritmo de remoção de artefatos produzido pelo autor. Por fim, o autor nos mostra que há um ganho de performance considerável se comparado a outros métodos de esqueletização de objetos já existentes. Além disso, também nos é mostrado que há um resultado muito satisfatório na remoção de artefatos da imagem, conseguindo remover todos os artefatos gerados.

No trabalho proposto por (SýKORA, 2012), o autor apresenta um *framework* baseado em exemplos para a reutilização de desenhos *cartoon* tradicionais. O principal objetivo deste trabalho é a capacidade de geração de novos esboços a partir de uma imagem de *cartoon* exemplo, assim como a criação de uma interface intuitiva e de fácil utilização. Para isto, inicialmente faz-se uso de técnicas para fragmentação não supervisionada do

objeto *foreground* em um conjunto de regiões e ocorre a separação do *background*. Após, o usuário seleciona um subconjunto de regiões ao qual deseja modificar, que são vetorizadas. Então o usuário posiciona este subconjunto de regiões elaborando um novo esboço. O autor avalia os resultados obtidos aplicando seu *framework* desenvolvido em um *cartoon* real denominado "O loupežníku Rumcajsovi". Com isto, conseguindo realizar a criação de novos personagens em alguns segundos, feito este que demoraria minutos para ser realizado em ferramentas de manipulação de imagens padrão. Adquirindo assim, bons resultados nestes testes. Vale-se ressaltar também que para casos não ideias para o *framework*, isto é, *cartoons* que apresentem regiões não homogêneas ou linhas externas não contínuas e bem definidas, fez-se necessário o uso de algoritmos de junção de linhas externas ou segmentação como auxílio ao *framework*.

No trabalho proposto por (YAO, 2017), o autor apresenta um sistema de vetorização de mangás escaneados para manipulação interativa e renderização em tempo real independente de resolução. O objetivo do trabalho é a criação da representação vetorial do mangá escaneado sem artefatos e com a possibilidade de fácil deformação, para geração de novas imagens. Para isto, utiliza-se o método desenvolvido pelo autor para a identificação de regiões *screentone*. Então classifica-se estas em *screentone* complexas ou simples utilizando *local binary pattern* (LBP) auxiliado de um BD. Coleta-se as propriedades simples e considera-se as complexas como regiões de sombreamento sólido, fazendo assim a decomposição da imagem em sólida e *screentone*. Após, faz-se a vetorização da imagem sólida. Então ocorre a renderização da imagem final, composta pela parte sólida e *screentone*, podendo haver a manipulação ou não desta imagem através de *Bounded Biharmonic Weights*.

Para a validação deste trabalho o autor utilizou uma base de dados composta por 191 personagens, onde estes apresentavam diferentes propriedades relacionadas ao seu estilo *screentone*. Além disto, a fim de testar a robustez do método proposto, também escaneou-se as imagens resultantes para que as mesmas passassem novamente pela aplicação. Por fim, o autor nos mostra que conseguiu atingir grandes resultados, representando as imagens em sua versão vetorizada, removendo dependências de resoluções e exigindo menos espaço em memória que suas versões *raster*. Além de permitir a manipulação por parte do usuário.

4 DESENVOLVIMENTO

Nesta seção apresentaremos o desenvolvimento de uma cadeia de processos que visam gerar uma nova representação para animações *cartoon*. Esta representação, consegue, na maioria das vezes, devido a sua menor complexidade, representar a animação com uma menor quantidade de dados. Além disso, é independente de resolução e taxa de quadros, sendo possível interpolar novos quadros entre os originais. A seção é dividida da seguinte forma: 1) Ferramentas utilizadas. Buscará abordar as ferramentas que foram utilizadas durante o processo de criação. 2) Funcionalidades da aplicação. Nesta, será demonstrado qual foi a sequência lógica para realização da proposta. 3) Proposta de modelo para representação de animações *cartoon*. Por fim, será demonstrado utilizando recursos gráficos os procedimentos da nova representação.

4.1 Ferramentas utilizadas

Para o desenvolvimento deste trabalho, optamos pela utilização da linguagem Python. Esta trata-se de uma linguagem de fácil entendimento e desenvolvimento. Além disso, esta provém diversas funcionalidades através de módulos bem desenvolvidos que suprem as necessidades deste trabalho. Outro ponto importante para esta escolha está em não priorizarmos tempo de execução como um dos principais objetivos desse trabalho, portanto, este ponto fraco da linguagem não será um grande problema. Este trabalho visa apenas conseguir criar uma representação que será independente de taxas de quadros e resolução. Podendo também, devido a sua representação simplificada, necessitar de uma menor quantidade de espaço em disco para armazenamento.

Como módulos desta linguagem utilizados para o desenvolvimento deste trabalho, podemos citar. (BRADSKI, 2000), que será o responsável pelo redimensionamento dos quadros resultantes, para que então possamos criar os vídeos a partir dessas imagens redimensionadas. (BINGOL; KRISHNAMURTHY, 2019), que será o módulo responsável pela confecção das NURBS utilizadas para a criação da representação temporal. Além disso, também será utilizado para a confecção das B-spline que representam os novos quadros criados após o fatiamento da NURBS. Por último, também utilizamos o módulo (KLEIN et al., 2018), para a criação das animações com os quadros interpolados.

4.2 Processos para a criação da representação

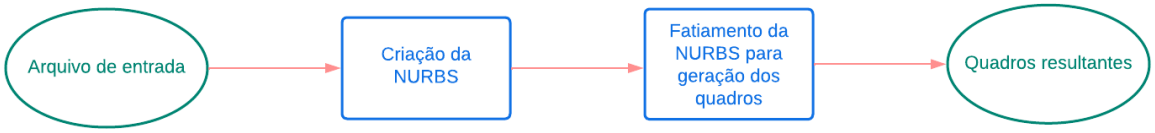
Este trabalho tem como sua proposta uma representação para animações *cartoon* independente de taxa de quadros. Para a criação desta representação temos que realizar duas cadeias de processos. A primeira dessas, efetua a criação da representação temporal da animação *cartoon* para a interpolação de novos quadros, que serão o resultado final destes procedimentos. Enquanto isto, a segunda efetua os tratamentos necessários nesses novos quadros gerados e faz a junção destes em uma nova animação resultante.

O processo de interpolação de novos quadros pode ser observado na figura 10, que representa visualmente o processo completo que será explicado a seguir.

Este processo terá inicio com o recebimento de um arquivo de entrada. Este arquivo contém os pontos de controle que representam temporalmente a forma que será representada por uma NURBS. Isto é, os pontos de controle que representam uma determinada forma de objeto em todos os quadros da animação original. Além disso, este arquivo também precisará conter duas *flags* utilizadas pelo algoritmo para separação dos dados. A primeira delas sendo a flag *fim_nurbs*, que será responsável por informar o fim de um determinado objeto naquele quadro. Visto que o modelo proposto suporta múltiplas NURBS. A fim de conseguir representar mais de um objeto por arquivo, ou mesmo objetos mais complexos que não podem ser representados por uma única. A segunda *flag* utilizada é *fim_imagem*. Esta será responsável por informar onde um determinado quadro encerra e se inicia outro.

Após o recebimento deste arquivo de entrada, efetuamos a construção temporal da NURBS com os dados fornecidos. Com essa NURBS criada, temos nossa representação temporal, pois a mesma pode ser fatiada em quantos quadros desejarmos. Isso se da pelo possível cálculo de novos pontos delimitadores destes quadros através da função de NURBS. Com esta representação tridimensional criada, efetuamos o fatiamento do mesmo para a interpolação de novos quadros. Aqui podemos ajustar a complexidade do objeto resultante como também a quantidade de quadros desejados.

Figura 10 – Processo de criação dos novos quadros.



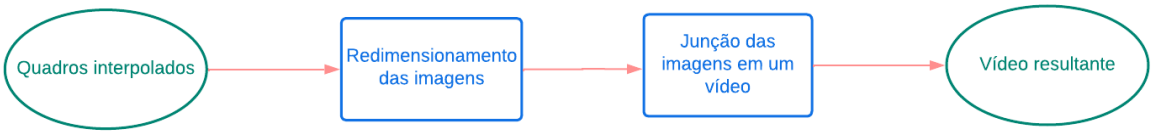
Fonte: Autor

Com o fatiamento desses quadros, o modelo proposto nos resulta em uma sequência de quadros gerados. Estes novos quadros então são armazenados localmente para que possamos prosseguir com os passos representados pela figura 11. Esta figura representa visualmente os passos para a reconstrução da animação a partir dos quadros interpolados, que será explicado a seguir.

O processo de criação da animação a partir dos quadros interpolados começa com o recebimento destes quadros, que serão a entrada deste processo. Após, fazemos um tratamento nessas imagens, que será o redimensionamento delas para criarmos imagens com valores de altura e largura iguais. Precisamos desta etapa pois os quadros interpolados

não necessariamente serão gerados com valores iguais de altura e largura. Além disso, se estes valores forem iguais, não conseguimos efetuar a criação da animação a partir deles. Após o redimensionamento dessas imagens, efetuamos a criação da animação através delas.

Figura 11 – Processo de criação da animação a partir dos quadros interpolados.



Fonte: Autor

Este processo nos resulta no vídeo que será o objeto final de nossos processos, e também nosso objetivo inicial. Com estes processos conseguimos gerar uma nova animação a partir da original com a quantidade de quadros desejada. Conseguindo com isso, por exemplo, criar uma animação mais suave e polida ao olho humano.

Essa representação possívelmente precisará de menos espaço para seu armazenamento, visto que pode ser armazenado na forma do arquivo de entrada do processo apresentado na figura 10. Dado que este arquivo precisa guardar poucas informações que representem os objetos se comparado a formas comumente utilizadas para armazenar vídeos.

4.3 Representação de animações *cartoon*

Para a criação da nova representação proposta, faz-se necessário o recebimento de um arquivo que contenha informações de pontos de controle de um objeto na animação. Na figura 12 pode-se observar um exemplo de dois quadros retirados de uma animação *cartoon*. Utilizaremos estes dois quadros para exemplificar de maneira gráfica a proposta de criação da nova representação. Demonstrando a utilização de NURBS para a construção de quadros a partir dos pontos de controles obtidos.

Inicialmente, a figura 13 busca representar um dos personagens em primeiro plano que compõem o quadro em questão, separado de seu plano de fundo em ambos os quadros. Com isto podemos simplificar o nosso objeto em questão utilizado para exemplificação. Após, a figura 14 demonstra a coleta dos pontos de controle necessários para a criação do arquivo de entrada. Para este exemplo, estamos realizando a construção de uma única NURBS, que representará a borda externa do escudo. Como podemos notar, esse processo é realizado para todos os quadros que compõem a sequência, nesse caso exemplo, os dois quadros utilizados são os que compõem a figura 12.

Figura 12 – Dois quadros que compõem a sequência de quadros esperada.

Fonte: Autor sobre imagens da internet

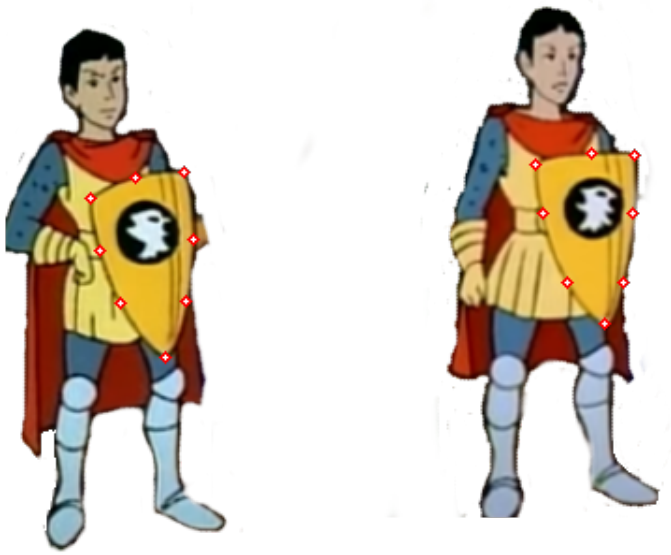
Figura 13 – Objetos selecionado após a remoção de fundo.

Fonte: Autor sobre imagens da internet

Com a identificação dos pontos de controle do componente do objeto em questão, pode-se aplicar a fórmula de NURBS para gerar uma representação do objeto em razão do tempo. Com isto, o objeto em um quadro será representado por uma B-spline. Enquanto a natureza tridimensional da fórmula de NURBS será utilizada para representar os próximos quadros que compõem a sequência. Podemos observar essa criação na figura 15.

A terceira dimensão da NURBS será a taxa de quadros. Portanto, na figura

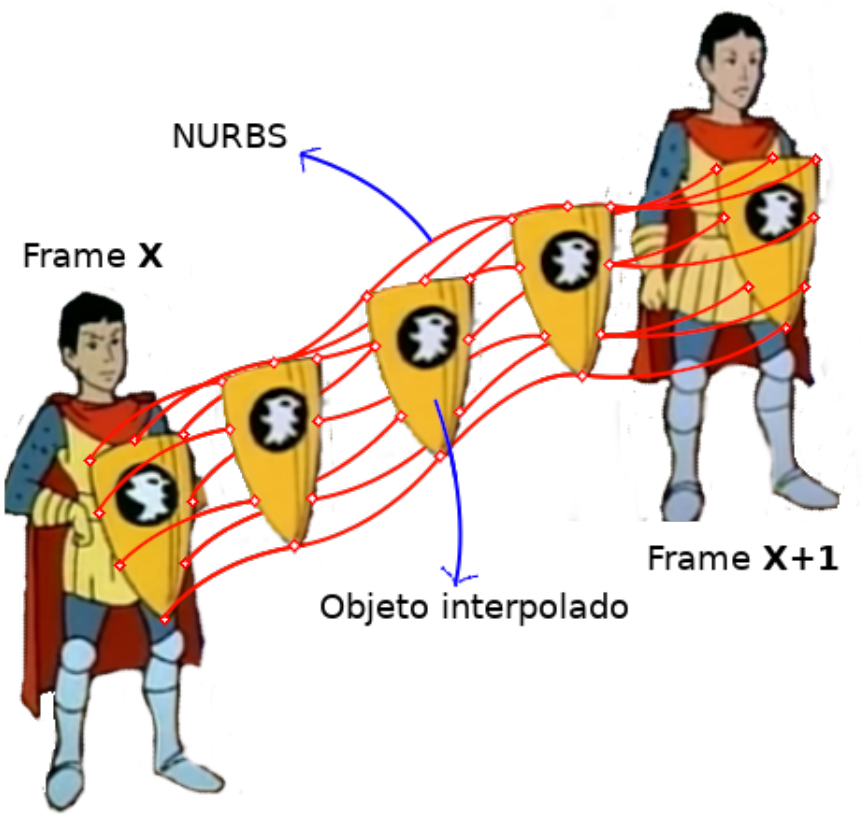
Figura 14 – Coleta dos pontos de controle nos quadros.



Fonte: Autor sobre imagens da internet

exemplo 15 a imagem a esquerda representa determinado quadro x enquanto a imagem a direita representa o seguinte *x+1*. Entretanto, essa representação gera a possibilidade de definirmos, por exemplo, esses quadros como sendo os quadros x e *x+8*, respectivamente. Para que com isto, consigamos criar outros seis quadros entre estes. Isto faz com que essa representação seja independente de taxa de quadros. Conseguimos então criar uma maior quantidade de quadros para gerar, por exemplo, uma suavização em suas transições. Além disso, a representação B-spline é naturalmente vetorial, como demonstrado na imagem 15, fazendo com que esta possa ser gerada como uma imagem vetorial. Isso faz com que haja a possibilidade de criar imagens independentes de resolução, dado que sua geração de imagens é efetuada através de seus pontos de controle.

Figura 15 – Utilização da função NURBS para modelagem do objeto.



Fonte: Autor sobre imagens da internet

5 RESULTADOS OBTIDOS

Nesta seção busca-se avaliar os resultados obtidos neste trabalho. Aqui mostraremos os resultados dos processos de interpolação dos novos quadros e construção da animação, apresentados nas figuras 10 e 11 respectivamente. Esta seção será dividida nas seguintes subseções: 1) Representação temporal de objetos utilizando NURBS. Essa subseção tem por objetivo mostrar os resultados na criação da superfície NURBS que representará temporalmente o objeto. 2) Interpolação de novos quadros. Aqui vamos analisar os novos quadros interpolados a partir dos originais da animação. Vamos avaliar a transição gerada a partir dos quadros originais da animação e a eventual suavização que isto poderá proporcionar a animação. Vamos também avaliar a similaridade que um quadro interpolado tem em relação ao quadro original, por exemplo, avaliando o quadro interpolado que representa exatamente o quadro x no eixo temporal em relação ao próprio quadro x original. Além disso, vamos abordar brevemente a necessidade de espaço em disco para armazenar a representação proposta em comparação a necessidade para armazenar os videos originais.

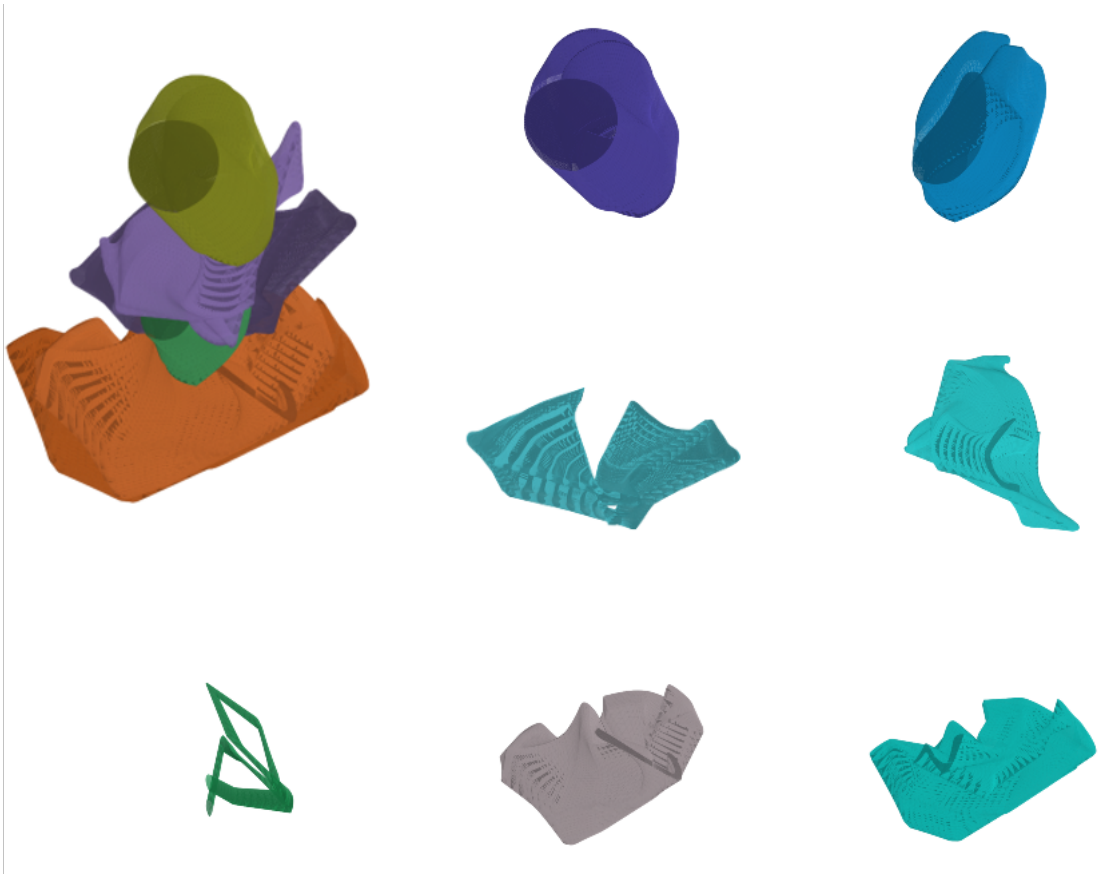
5.1 Representação temporal de objetos utilizando NURBS

A representação temporal de objetos trata-se de uma etapa importante do processo demonstrado na figura 10. Essa etapa é realizada logo após o recebimento do arquivo de entrada. Os resultados para esta etapa podem ser observados na figura 16. Essa figura representa as superfícies construídas para a representação temporal de um personagem realizando uma corrida. Como podemos notar, a figura é composta por uma imagem que representa todas as superfícies em conjunto, mas também por imagens de cada superfície separadamente. O objetivo das superfícies decompostas é proporcionar uma melhor visualização da transição de cada uma das partes que compõem o personagem ao longo da animação.

A construção dessas NURBS foi realizada utilizando a biblioteca (BINGOL; KRISHNAMURTHY, 2019). Essa biblioteca é capaz de criar essas superfícies por meio da especificação das variáveis que compõem a equação de NURBS mencionada anteriormente, bem como a complexidade desejada para a sua criação. A complexidade está relacionada a quantidade de pontos na superfície que devem ser calculados para a criação da representação computacional dessa superfície. Essa complexidade é muito importante, pois quanto maior ela for, mais pontos serão calculados para criar a representação, tornando-a mais fiel à imagem original.

Além disto, quanto maior a quantidade de pontos calculados nessa superfície, mais camadas interpoladas serão permitidas, pois serão necessários mais pontos calculados no eixo temporal para gerar mais camadas intermediárias. No entanto, é importante mencionar que quanto maior a complexidade incorporada, maior será o tempo necessário para a criação dessa representação, já que será preciso gastar mais tempo em cálculos de

Figura 16 – Personagem criado a partir de superfícies geradas pela equação de NURBS e cada uma das superfícies que o compõem



Fonte: Autor

pontos na superfície.

5.2 Interpolação de novos quadros

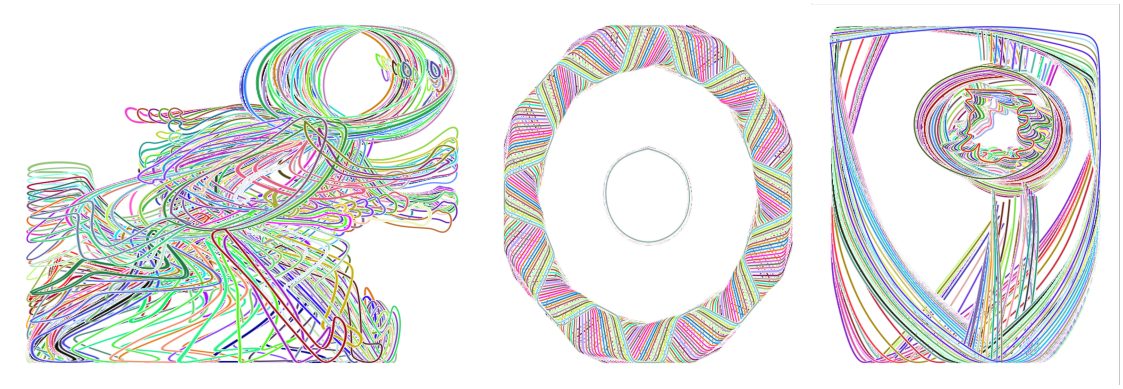
A interpolação de novos quadros é o que nos permite gerar os resultados apresentados no final do processo, conforme mostrado na figura 10, e que servem como entrada para a etapa demonstrada na figura 11.

A interpolação de novos quadros a partir dos pontos delimitadores que definem a animação de entrada pode ser observada na figura 17. Esta figura apresenta três exemplos de quadros interpolados utilizando nossa representação. Todos os quadros criados, a partir da representação temporal, foram colocados em uma mesma figura, sendo cada um deles pintado com uma cor diferente para melhor visualização. Neste exemplo, foram interpolados três novos quadros entre cada par de quadros originais das animações. Ou seja, entre cada quadro x e $x + 1$, são criados três novos quadros. Os trechos das animações interpoladas contavam com 11, 8 e 14 quadros respectivamente. A partir das

nossas representações, as animações resultantes foram compostas por 49, 37 e 53 quadros respectivamente.

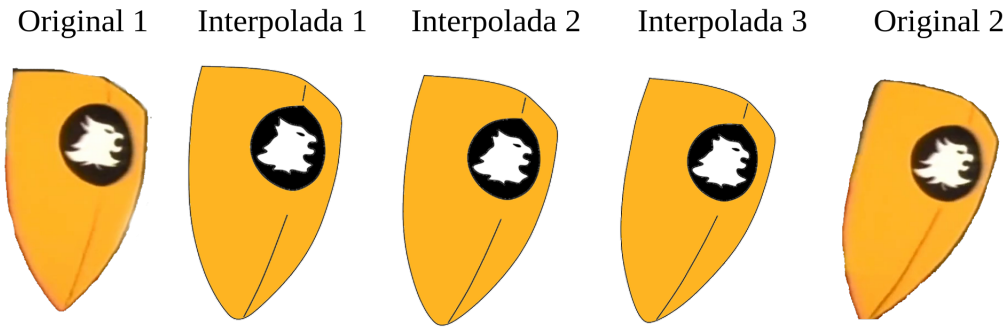
Como podemos observar pela figura 17, as transições entre as posições na animação foram suavizadas, visto que conseguimos perceber que há um espaço curto entre um objeto e outro. Além disto, as animações resultantes destes três casos apresentados podem ser visualizadas no repositório científico *figshare* (DULLIUS, 2023). O repositório conta com os três exemplos que interpolam três quadros novos entre cada par de quadros originais, mas também três exemplos com oito novos quadros criados entre cada par de quadros originais.

Figura 17 – Representação de quadros resultantes sobrepostos



Fonte: Autor

Figura 18 – Dois quadros originais do objeto na animação com três quadros interpolados entre estes



Fonte: Autor

Outro exemplo que podemos observar está na figura 18. Essa figura busca demonstrar visualmente a transição criada através dos quadros interpolados entre dois quadros da animação original. Foram selecionados dois quadros originais da animação, que podem

ser identificados na figura como *original 1* e *original 2*. Em seguida, foram interpolados três novos quadros a partir deles, chamados de *Interpolada 1*, *Interpolada 2* e *Interpolada 3*. Essas imagens interpoladas foram pintadas manualmente após a sua criação, a fim de proporcionar uma melhor representação de um cenário utilizando essas imagens em uma animação real.

Na figura 18, é possível notar que a transição inicial entre os dois quadros era brusca. No entanto, essa transição foi melhorada pela interpolação dos novos quadros, o que resultou em uma transição mais suave na animação entre esses quadros originais. Isso contribui significativamente para a qualidade da animação, pois causa menos estranheza aos espectadores.

Além disso, é possível observar que a animação fez a transição de forma correta entre esses dois quadros originais, seguindo o movimento esperado que o objeto teria ao sair do primeiro quadro e chegar ao segundo.

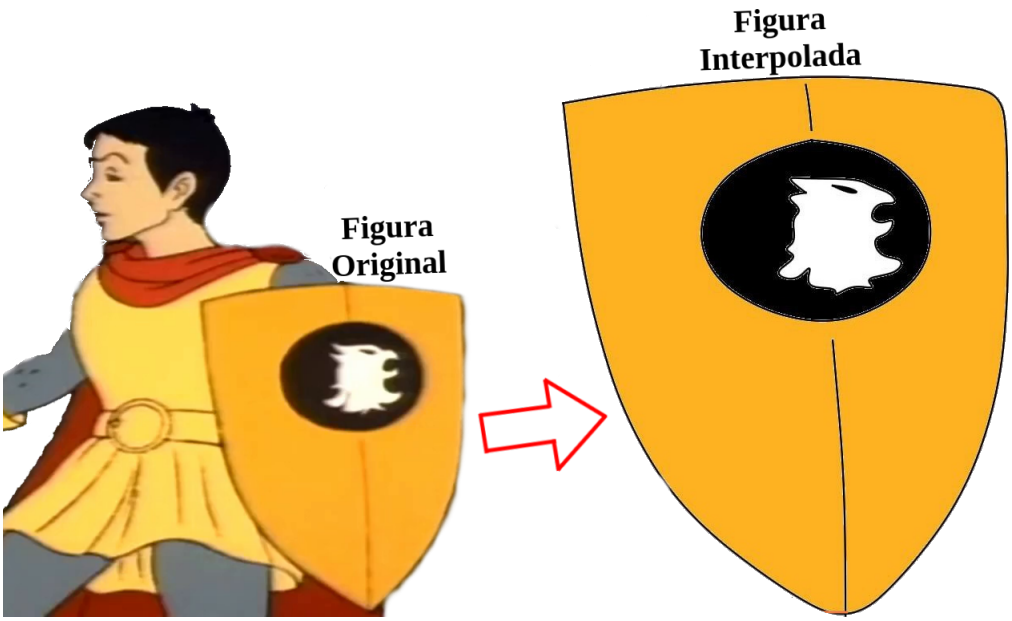
5.2.1 Similaridade entre o objeto interpolado e original

Nesta subseção, vamos avaliar a similaridade entre um objeto original na animação e seu respectivo quadro interpolado. Para isso, iremos comparar um objeto em um determinado quadro x com o objeto interpolado na mesma coordenada temporal da representação.

Como podemos observar na figura 19, a interpolação apresenta muita similaridade com o objeto original. Nesta figura, as formas externas e o círculo interno apresentam uma similaridade realmente agradável visualmente. Porém, a águia interna no escudo apresenta certos níveis de deformidades que podem ser notados quando observamos uma imagem. Esses níveis de deformidade estão principalmente relacionados à quantidade e qualidade dos pontos delimitadores coletados para a construção da representação temporal. Objetos mais complexos, que apresentam mais curvas e detalhes, naturalmente precisam de mais pontos de controle para efetuar sua representação, visto que apenas assim conseguiremos passar esses detalhes para a superfície interpolada. Além disso, a qualidade dos pontos deve ser considerada também, isto é, os pontos devem ser coletados de forma a mapear o objeto da melhor maneira. Esses pontos devem ser coletados principalmente em cantos e ângulos que mapeiem todos os detalhes do objeto.

Entretanto, muitas vezes, em uma animação, sutis deformidades não podem ser observadas e, inclusive, podem contribuir para a sensação de movimento do objeto. Podemos notar isso através das figuras 10 e 17, que representam os quadros em movimento na animação. Portanto, a similaridade apresentada nos objetos interpolados é satisfatória e resultados melhores podem ser alcançados através de uma maior quantidade e qualidade dos pontos de controle.

Figura 19 – Objeto original de um quadro da animação e o mesmo objeto interpolado através da NURBS



Fonte: Autor

5.2.2 Necessidade de armazenamento para representação proposta

A necessidade de armazenamento para nossa representação proposta está ligada, assim como a similaridade do objeto, à quantidade de pontos delimitadores coletados. Além disso, também há o impacto menor gerado pelas flags. Entretanto, se olharmos para o contexto como um todo, as flags normalmente representam uma proporção muito menor da quantidade de dados que compõem o arquivo de entrada, sendo quase insignificantes para um contexto maior.

Como podemos observar na tabela 1, para ambos os casos utilizados como exemplo de comparação, a representação proposta exigiu menos espaço de armazenamento do que a animação original. No caso da animação de uma engrenagem girando, tivemos uma compressão de 84,23% dos dados em relação à animação original, o que pode ser considerado um grande feito.

Entretanto, para objetos mais complexos, como por exemplo, o personagem correndo, que necessitam de uma maior quantidade de pontos delimitadores para sua representação, a quantidade de compressão é menor. Isso ocorre, como já citado anteriormente, devido à exigência de uma maior quantidade de pontos delimitadores para a representação do objeto, a fim de manter uma maior similaridade.

Portanto, o espaço necessário para o armazenamento da representação proposta está bastante ligado à qualidade desejada da imagem interpolada e também à complexidade do objeto a ser representado. Em casos mais críticos, isso pode resultar em uma

representação que demande uma maior quantidade de dados em relação à animação original. No entanto, esse não é o foco principal deste trabalho e pode ser melhor abordado em trabalhos futuros que tenham como objetivo específico aplicar processos que diminuam esse espaço requisitado.

Nome animação	**Quantidade de quadros**	**Espaço original**	**Espaço proposta**
Personagem correndo	12	40,0kB	11,6kB
Engrenagem girando	9	22,2kB	3,5kB

Tabela 1 – Comparação de necessidade de espaço para armazenar animação original e representação proposta

6 CONSIDERAÇÕES FINAIS

Neste trabalho foram investigados diversos métodos utilizados no processamento de imagens estilo *cartoon*, assim como também tratamentos de animações e compressão de imagens neste estilo. Com isso, aplicamos técnicas de processamento de imagens e construção de NURBS para efetuar a criação de uma representação paramétrica de animações *cartoon*. Essa representação se provou promissora, visto que com ela conseguimos efetuar a interpolação de novos quadros a partir de um conjunto de pontos delimitadores coletados a partir dos quadros originais da animação. Além de também possibilitar a construção de novos objetos a partir de pontos delimitadores quaisquer estabelecidos pelo animador.

A representação proposta também se demonstrou útil em diversos casos, em relação a economia de espaço necessário para armazenamento. Visto que trata-se de uma representação menos complexa que as comumente utilizadas para a construção de vídeos, necessitando apenas de um arquivo que descreva os pontos delimitadores e as *flags* de fim de quadro e fim de NURBS.

No processo atual de construção desta representação, os pontos delimitadores utilizados ainda são coletados manualmente, dado que este não foi o foco do escopo atual deste trabalho. Como esta pode ser uma tarefa trabalhosa, um possível caminho a ser tomado em trabalhos futuros é a coleta automática destes pontos. O qual deve delimitar e relacionar os objetos no espaço temporal que descreve os quadros. Outro caminho que pode ser abordado em trabalhos futuros é a identificação e separação de objetos em primeiro plano. Para que com isso possa também automatizar o processo da escolha dos objetos a serem interpolados.

Por fim, este trabalho apresentou uma proposta de representação de animações *cartoon* que busca ser independente de taxas de quadros. Sendo uma alternativa as redes neurais profundas, que são as mais comumente utilizadas na interpolação de quadros. Além disso, buscamos desenvolver uma representação específica para animações no estilo *cartoon*, buscando explorar a sua natureza.

REFERÊNCIAS

AZEVEDO, E. **Computação Gráfica - Teoria e Prática**. 2022. Citado 5 vezes nas páginas 14, 15, 21, 22 e 23.

BAO, W. **Depth-Aware Video Frame Interpolation**. 2019. Citado na página 12.

BINGOL, O. R.; KRISHNAMURTHY, A. NURBS-Python: An open-source object-oriented NURBS modeling framework in Python. **SoftwareX**, Elsevier, v. 9, p. 85–94, 2019. Citado 2 vezes nas páginas 32 e 38.

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000. Citado na página 32.

DULLIUS, D. **Animações originais e interpoladas**. figshare, 2023. Disponível em: <https://figshare.com/collections/Anima_es_originais_e_interpoladas/6741606>. Citado na página 40.

ERSTELLT, S. **Non-uniform rational B-spline**. 2007. Disponível em: <https://www.wikiwand.com/en/Non-uniform_rational_B-spline#Media/File:NURBS_surface.png>. Citado na página 25.

GIOVANINI, A. **Processamento digital de imagens: o que é?** 2019. Disponível em: <https://adenilsongiovanini.com.br/blog/processamento-digital-de-imagens/>. Citado na página 17.

GONZALEZ, R. C. **Digital Image Processing**. 2007. Citado 6 vezes nas páginas 14, 15, 16, 17, 18 e 19.

HOANG, C. thai. **DEVELOPMENT OF ISOGEOMETRIC FINITE ELEMENT METHODS**. 2015. Citado na página 24.

HOLINEY, V. **Qual a diferença entre o gráfico vetor o raster?** 2019. Disponível em: <https://www.logaster.com.br/blog/vector-and-raster/>. Citado 2 vezes nas páginas 15 e 16.

IGM, I. G. de M. **Curvas paramétricas**. Disponível em: <https://www.geogebra.org/m/f5q9scd9>. Citado na página 22.

KHAN, M. **Detection of Violent Content in Cartoon Videos Using Multimedia Content Detection Techniques**. 2018. Citado na página 27.

KLEIN, A. et al. **imageio/imageio: V2.4.1**. Zenodo, 2018. Disponível em: <https://doi.org/10.5281/zenodo.1488562>. Citado na página 32.

KUNDU, R. **BLOG COMPUTER VISION Image Processing: Techniques, Types, Applications [2023]**. 2023. Disponível em: <https://www.v7labs.com/blog/image-processing-guide>. Citado na página 18.

MAINBERGER, M. **Edge-based compression of cartoon-like images with homogeneous diffusion**. 2011. Citado 2 vezes nas páginas 12 e 29.

PIEGL, L. **The NURBS Book**. 1996. Citado 2 vezes nas páginas 20 e 23.

RAJ, Y. A. **Turtle edge encoding and flood fill based image compression scheme**. 2019. Citado 2 vezes nas páginas 12 e 29.

SýKORA, D. **Unsupervised colorization of black-and-white cartoons**. 2004. Citado na página 28.

SýKORA, D. **Sketching Cartoons by Example**. 2012. Citado na página 30.

TAKAYAMA, K. **FACE DETECTION AND FACE RECOGNITION OF CARTOON CHARACTERS USING FEATURE EXTRACTION**. 2012. Citado na página 28.

TAUBMAN, D. **JPEG200 Image compression fundamentals, standards and practice**. 2001. Citado 2 vezes nas páginas 19 e 20.

YAO, C.-Y. **Manga Vectorization and Manipulation with Procedural Simple Screentone**. 2017. Citado na página 31.

ZHANG, S.-H. **Vectorizing Cartoon Animations**. 2009. Citado na página 27.

ZHAO, Y. **Cartoon Image Processing: A Survey**. 2022. Disponível em: <https://link.springer.com/article/10.1007/s11263-022-01645-1>. Citado na página 29.

ZOU, J. J. **Cartoon image vectorization based on shape subdivision**. 2001. Citado na página 30.