UNIVERSIDADE FEDERAL DE PELOTAS
Centro de Desenvolvimento Tecnológico
Curso de Bacharelado em Engenharia de Computação

Trabalho de Conclusão de Curso

Um novo método para detecção de colisão em tempo real para simulação de colonoscopia

Lucas Zanusso Morais

Pelotas, 2023

Lucas Zanusso Morais

Um novo método para detecção de colisão em tempo real para simulação de colonoscopia

Trabalho de Conclusão de Curso apresentado ao Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas, como requisito parcial à obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Rafael Piccin Torchelsen
Coorientador: Prof. Dr. Anderson Maciel

Pelotas, 2023

Universidade Federal de Pelotas / Sistema de Bibliotecas
Catalogação na Publicação

M827n Morais, Lucas Zanusso

Um novo método para detecção de colisão em tempo real para simulação de colonoscopia / Lucas Zanusso Morais ; Rafael Piccin Torchelsen, orientador ; Anderson Maciel, coorientador. — Pelotas, 2023.

55 f. : il.

Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) — Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, 2023.

1. Computação gráfica. 2. Detecção de colisão. 3. Simuladores cirúrgicos. 4. Métodos numéricos. I. Torchelsen, Rafael Piccin, orient. II. Maciel, Anderson, coorient. III. Título.

CDD : 005

Elaborada por Maria Inez Figueiredo Figas Machado CRB: 10/1612

Lucas Zanusso Morais

Um novo método para detecção de colisão em tempo real para simulação de colonoscopia

Trabalho de Conclusão de Curso aprovado, como requisito parcial, para obtenção do grau de Bacharel em Engenharia de Computação, Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas.

Data da Defesa: 8 de Maio de 2023

Banca Examinadora:

Prof. Dr. Rafael Piccin Torchelsen (orientador)
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Anderson Maciel (co-orientador)
Doutor em Computação pela Ecole Polytechnique Fédérale de Lausanne.

Prof. Dr. Marilton Sanchotene de Aguiar
Doutor em Computação pela Universidade Federal do Rio Grande do Sul.

Prof. Dr. Luciana Porcher Nedel
Doutora em Computação pela Ecole Polytechnique Fédérale de Lausanne.

Dedico esse trabalho a minha universidade, professores e colegas, que me proveram com conhecimento, propósito e inspiração. Em especial ao meu grupo de pesquisa coordenado pelo professor Rafael Piccin Torchelsen a quem devo por suas orientações e encaminhamentos. Também em especial aos meus colegas durante o curso, que tanto me inspiraram e compartilharam esses anos comigo. E principalmente a minha família, que se hoje consigo dar um passo a frente, é por estar em seus ombros.

AGRADECIMENTOS

Agradeço a todos que tanto me apoiaram e estiveram ao meu lado por esses anos. A minha mãe e vó, Rosane e Enilda, as quais tornaram tudo possível. Aos meus colegas e amigos que me direcionam e inspiram a alçar voos maiores. Meus colegas que entraram na graduação comigo e passaram esses anos juntos. Meus amigos da 337 a quem devo por tantos conselhos, acolhimento e carinho. Meus amigos do projeto de pesquisa que trabalharam comigo lado a lado Victor, Marcelo e João. Meus amigos Matheus, Guilherme, João, Victor, Lucas, Beatriz, Gabriel e Milene, os quais devo todo suporte e apoio do mundo. A minha namorada Raiza, que tanto me inspira e impulsiona. E muitas outras pessoas incríveis que me ajudaram, inspiraram, direcionaram e de muitas formas fazem parte desse marco.

Standing on the shoulders of giants.
— ISAAC NEWTON

RESUMO

MORAIS, Lucas Zanusso. **Um novo método para detecção de colisão em tempo real para simulação de colonoscopia**. Orientador: Rafael Piccin Torchelsen. 2023. 54 f. Trabalho de Conclusão de Curso (Engenharia de Computação) – Centro de Desenvolvimento Tecnológico, Universidade Federal de Pelotas, Pelotas, 2023.

O câncer colorretal se mostra como uma das principais causas de mortes relacionadas à câncer no mundo. Nesse cenário, a prevenção e tratamento em estágios iniciais se mostra fundamental, onde a colonoscopia é o principal procedimento na detecção e examinação. Uma importante parte do treinamento de novos médicos é feito através de simuladores virtuais, onde é possível ter um treinamento intensivo e eficaz. Simuladores virtuais possuem seus próprios desafios, pois precisam se adequar a realidade e prover um ambiente realístico. Neste cenário, a detecção de colisões entre objetos da simulação é uma das partes centrais da aplicação, sendo necessário uma resposta acurada e interativa. Apesar de muita pesquisa sobre detecção de colisões entre malhas estáticas, ainda há muitos desafios quando se trata de malhas deformáveis, as quais são amplamente implementadas em simulações médicas. Além de não haverem técnicas específicas que tirem vantagem das características únicas da simulação de colonoscopia. Apresentamos assim uma simulação específica para detecção de colisões em simulações de colonoscopia, podendo ser generalizada para atender outros casos. Nos apropriamos da paridade endoscópio e cólon serem a únicas malhas presentes na simulação e seu formato cilíndrico, montando estruturas de dados específicas para esse cenário. Conseguimos prover uma detecção de colisão interativa e de crescimento linear, apresentando uma alternativa a outros métodos mais genéricos.

Palavras-chave: Computação Gráfica. Detecção de Colisão. Simuladores Cirúrgicos. Métodos Numéricos.

ABSTRACT

MORAIS, Lucas Zanusso. **A new collision detection method between deformable meshes, using splines**. Advisor: Rafael Piccin Torchelsen. 2023. 54 f. Undergraduate Thesis (Computer Engineering) – Technology Development Center, Federal University of Pelotas, Pelotas, 2023.

Colorectal cancer is one of the main causes of cancer-related deaths in the world. In this scenario, prevention and treatment in the early stages is essential, where colonoscopy is the main procedure for detection and examination. An important part of training new doctors is done through virtual simulators, where it is possible to have an intensive and effective training. Virtual simulators have their own challenges, as they need to adapt to reality and provide a realistic environment. In this scenario, collision detection between simulation objects is one of the central parts of the application, requiring an accurate and interactive response. Despite much research on collision detection between static meshes, there are still many challenges when it comes to deformable meshes, which are widely implemented in medical simulations. In addition, there are no specific techniques that take advantage of the unique characteristics of simulated colonoscopy. Thus, we present a specific simulation for collision detection in colonoscopy simulations, which can be generalized to other cases. We appropriated the endoscope and colon parity as the only meshes present in the simulation and its cylindrical format, assembling specific data structures for this scenario. We were able to provide interactive collision detection and linear growth, presenting an alternative to other more generic methods.

Keywords: Computer Graphics. Collision Detection. Surgical simulation. Numerical Methods.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

CRC	Câncer Colorretal
VR	Realidade Virtual
GPU	Placa Gráfica
BVH	Hierarquia de Volume Delimitadora
AABB	Axis-Align Bounding Box
OBB	Oriented Bounding Box

SUMÁRIO

1 INTRODUÇÃO

O câncer colorretal (CRC) é a terceira maior causa de mortes relacionadas ao câncer nos Estados Unidos, especula-se que chegue a 2.2 milhões de casos e 1.1 milhão de mortes até 2030 (SIEGEL; MILLER; JEMAL, 2019) (FENG et al., 2021). Neste cenário, a prevenção e o tratamento do câncer colorretal está fortemente relacionado à detecção de ademas, pólipos e CRC em estágios iniciais, sendo a colonoscopia o padrão ouro para detecção e remoção dos mesmos (DIK; MOONS; SIERSEMA, 2014) (REX et al., 2015).

A colonoscopia é um procedimento invasivo, com um potencial risco de complicações, onde é inserido um endoscópio, uma haste flexível com uma câmera presa em sua ponta, pelo ânus do paciente até o fim do cólon, para investigação e remoção de ademas, pólipos e CRC em estágios iniciais. A colonoscopia é um procedimento de difícil maestria, adquirida através de um treinamento intensivo, em que a principal forma de aprendizagem é observando um médico mais experiente na sala de cirurgia, seguindo o fluxo "veja uma vez, faça uma vez e ensine uma vez". Dessa forma, treinamentos com simuladores de realidade virtual (VR) levam a uma melhor performance durante um procedimento real (KOCH et al., 2012), sendo uma ferramenta essencial para novos médicos, melhorando o conforto e segurança de pacientes. Além de prover um ambiente seguro para novos médicos se aperfeiçoarem, simuladores proveem a capacidade de treinamento ininterrupto, quantas vezes for necessário, até a masterização do procedimento, acelerando a formação de novos médicos (WIEL et al., 2016) e servindo como forma de avaliação das habilidades de profissionais.

Assim, um dos aspectos que contribuem para a fidelidade dos simuladores é a capacidade de deformação dos tecidos e órgãos. Mas apenas mimetizar o comportamento mecânico não se faz suficiente, é necessário que a resposta háptica e a plausabilidade visual sejam respondidas em um curto período de tempo para usarmos em uma simulação interativa. Dessa forma, o desafio recai em uma solução rápida para a interação tecido-ferramenta (ZHANG; ZHONG; GU, 2017), onde a detecção de colisão se apresenta como um dos principais aspectos na efetiva simulação de malhas deformáveis (BERNDT; TORCHELSEN; MACIEL, 2017).

Como observado por (ROSA, 2019), a qualidade e fidelidade dos simuladores está em constante melhora pelo contínuo avanço do poder computacional das Placas Gráficas (GPU). Dessa forma, simulações complexas com visuais plausíveis e um comportamento físico aproximado da realidade conseguem ser construídas, retratando até o corte de tecidos, como visto em (BERNDT; TORCHELSEN; MACIEL, 2017).

Apesar da pesquisa sobre detecção de colisão entre malhas fixas já ter sido muito investigada, a detecção de colisão para malhas deformáveis apresenta novos desafios (TESCHNER et al., 2005), já que os atuais métodos aplicados para simular deformações em simuladores cirúrgicos não possuem o mesmo nível de escalabilidade que seus gráficos. Assim, não podemos apenas contar com o avanço do poder computacional, necessitando dessa forma de novas técnicas robustas e escaláveis para detecção de colisão entre malhas deformáveis em simuladores de colonoscopia.

1.1 Objetivos

O objetivo do estudo é desenvolver e validar uma nova técnica para detecção de colisão entre malhas deformáveis em simuladores de colonoscopia. O escopo é reduzido a simuladores de colonoscopia pelo caso específico onde a técnica é aplicada se apresentar de forma evidente. Assim é proposto um algoritmo de tempo real e consistente.

Os objetivos específicos buscarão: definir os principais aspectos no desenvolvimento de um simulador de colonoscopia, provendo uma visão geral do problema enquanto foca especificamente na detecção de colisão entre o cólon e o endoscópio, desenvolvendo e mostrando um método alternativo para detecção de colisão que possa ser utilizado em simulações de colonoscopia.

1.2 Fundamentos

1.2.1 Colonoscopia

A colonoscopia é um procedimento endoscópico, que permite a visualização da mucosa do intestino grosso e possibilita a retirada de amostras para histopatologia e a oportunidade de procedimentos terapêuticos como polipectomia ou dilatação de uma estenose. Com isso, a colonoscopia é considerada o padrão ouro na detecção precoce do câncer colorretal. No entanto, a colonoscopia é um procedimento invasivo, com potencial risco de complicações, como perfuração e sangramento (KO et al., 2010).

A inspeção do cólon requer um cólon limpo, sem resíduos que possam mascarar uma área suspeita. Por isso, são necessárias instruções claras do médico ao paciente, com seu consentimento nas preparações do procedimento (FROEHLICH et al., 2005). Assim, há varias opções de sedações, sendo a que resulta em um melhor conforto e

recuperação a sedação profunda (SINGH et al., 2008), conforme observado na Figura 1:

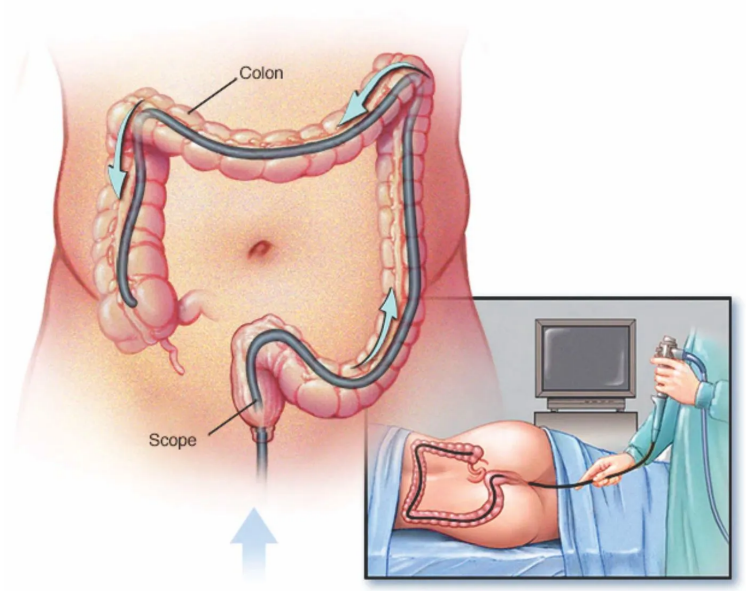

Figura 1 – A figura mostra um procedimento de colonoscopia bem-sucedido, onde no monitor ao fundo é mostrado a visão da ponta do endoscópio. Imagem retirada de Pace Hospital site https://www.pacehospital.com/colonoscopy

Complicações tendem a ocorrer aproximadamente em 2 a cada 1.000 colonoscopias feitas, com o risco aumentando se algum procedimento terapêutico foi performado (KO et al., 2010), podendo ocorrer imediatamente ou até vários dias após o procedimento. Uma das principais complicações que ocorre durante o procedimento é a formação de laços, a qual é ocasionada quando o colonoscópio se estende e distende no cólon em resposta aos esforços do médico para avançar o escopo para a frente (BRUCE; CHOI, 2018), ilustrado na Figura 2.

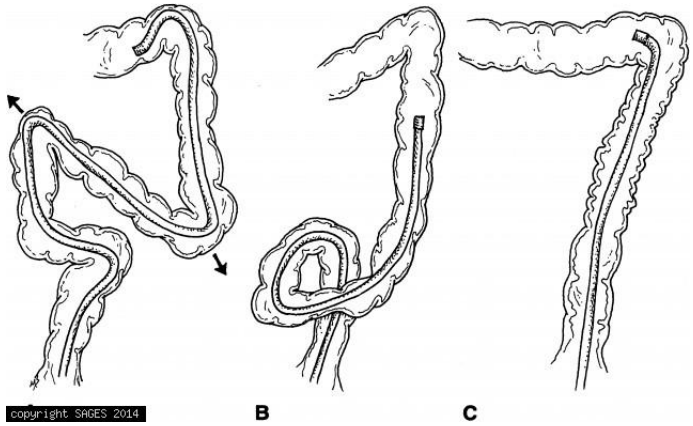

Figura 2 – A figura ilustra um tipo de laço sendo formado na região sigmoide do cólon. Imagem retirada de Sages 2014 https://www.sages.org/image-library/formation-of-loops-in-the-colon/

Durante a imagiologia endoscópica magnética (MEI), uma configuração tridimensi-

onal da haste do endoscópio é exibida em um monitor, provendo informação da posição anatômica do endoscópio e a ocorrência da formação de laços no endoscópio em tempo real. Além disso, o MEI pode mostrar a real efetividade de torcer o endoscópio, mudar o paciente de posição, exercer pressão abdominal e resolver laços formados no endoscópio, conseguindo assim performar o procedimento. (HAZEWINKEL; DEKKER, 2011)(SHAH et al., 2000).

Mesmo a colonoscopia sendo o padrão ouro na detecção de pólipos, é percebida uma taxa de erro de 2.1% para pólipos > 10mm, 13% para pólipos entre 5-10mm e 26% para pólipos < 5mm (VAN RIJN et al., 2006). Portanto, para alcançar níveis de detecção de ademas adequados, a inspeção minuciosa da mucosa do cólon é essencial. Nesse sentido, a ASGE-ACG e a BCSP Endoscopy Quality Assurance Group recomendam um tempo mínimo para a retirada do endoscópio de 6 minutos (HAZEWINKEL; DEKKER, 2011) e dados do National Popyp Trial mostram que a colonoscopia e a polipectomia previnem até 90% dos incidentes de câncer (WINAWER et al., 1993).

Desse forma, visualizamos alguns dos parâmetros e métricas centrais da qualidade de uma colonoscopia. A preparação do intestino é essencial, junto com uma colonoscopia completa, onde é visualizada a válvula ileocecal e uma cuidadosa inspeção com um maior tempo de observação, os quais favorecem a detecção de ademas. (HAZEWINKEL; DEKKER, 2011).

1.2.2 Simuladores

Tal importância da colonoscopia traz a necessidade da formação de um maior número de profissionais capacitados. Como vimos, o treinamento de novos profissionais é lento e custoso, forçando a procura de alternativas, que com o avanço de novas tecnologias e hardware capacitado, consegue simular o comportamento e aparência de um procedimento real. Como o procedimento real as simulações possuem seus próprios desafios e complexidades.

1.2.2.1 Detecção de Colisão

A tarefa de determinar se dois ou mais objetos estão em contato em algum ponto é chamada de detecção de colisão. Assim, quando possuímos dois objetos no espaço, a estratégia mais simples para verificarmos se estão colidindo é testar cada segmento do primeiro objeto contra todos os segmentos do segundo objeto para ver se eles se sobrepõem. Cada objeto é definido como uma malha de pontos, conectados para formarem triângulos (MOORE, 1988). Porém, dessa forma, não conseguimos testar todos os pares necessários de objetos, os quais possuem milhares de triângulos, pois a detecção de colisão precisa ser calculada a cada passo da simulação. Cenas reais possuem milhares de objetos, onde cada um pode possuir uma geometria complexa de milhares de polígonos. Assim se torna computacionalmente custoso a detecção de

colisão (KOCKARA et al., 2007).

A principal divisão da detecção de colisão é a *Broad-Phase* e *Narrow-Phase* (HUBBARD, 1993). Esses conceitos reduzem significativamente o custo da detecção de colisão ao remover testes desnecessários entre geometrias.

1.2.2.1.1 *Broad-Phase*

A *Broad-Phase* identifica geometrias que não podem colidir, as eliminando de testes futuros, agindo em uma granularidade mais grossa. Onde a *Narrow-Phase* age em uma granularidade mais fina, determinando se pares de geometrias estão se colidindo realmente (ERICSON, 2004).

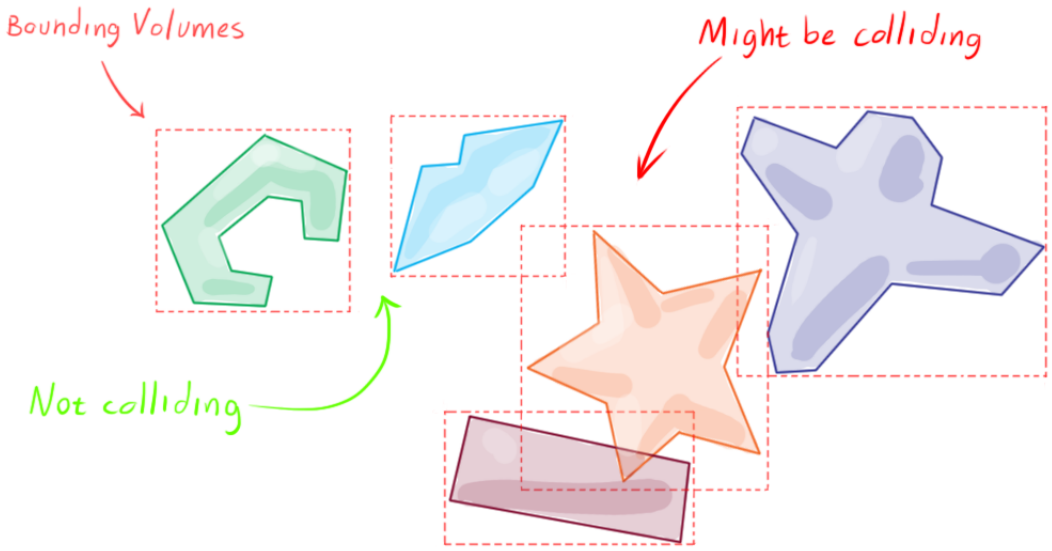

Figura 3 – A figura ilustra um dos algoritmos utilizados para a broad-phase. Imagem retirada de Ewa Szurlej Cov site `https://ewaszurlejgames.wordpress.com/project-2/`

Essa abordagem acaba funcionando como um pipeline, onde a *Broad-Phase* determina os testes a serem feitos na *Narrow-Phase*, como visto na Figura 3. Com isso, um algoritmo de *Broad-Phase* deve rapidamente eliminar geometrias onde não há chances de colisão. Uma das formas mais conhecidas para se fazer isso é encapsular geometrias mais complexas, dentro de mais simples como cubos com seus eixos alinhados. Chamamos esse encapsulamento de volume encapsulado e há alguns algoritmos que podemos aplicar para fazer os testes agora. O método mais simples é testar todos os volumes contra todos os outros. Mas outra abordagem é dividir o espaço em uma grade de células de mesmo tamanho, onde todas geometrias na mesma célula são prováveis colisões. Mas a abordagem mais comum entre malhas deformáveis é organizar esses volumes delimitadores em uma estrutura de dados em árvore (MIRTICH, 1996).

1.2.2.1.2 *Narrow-Phase*

Após a *Broad-Phase*, a *Narrow-Phase* possui todos pares de possíveis colisões, onde é responsável por determinar se uma colisão realmente ocorreu, e onde ocorreu. A *Narrow-Phase* retorna informações mais detalhadas sobre as colisões, as quais serão futuramente utilizadas na resposta de colisão e forças. Focaremos aqui nas duas principais estratégias para algoritmos na *Broad-Phase* que se adéquam a simulações dinâmicas, como a simulação de colonoscopia. Esses algoritmos são estruturas para o particionamento do espaço e hierarquia baseada em volume (KOCKARA et al., 2007).

Há duas principais estruturas de dados para o particionamento do espaço, divisão espacial e hierarquia de volume delimitadora (BVH). Na divisão espacial o espaço é dividido em células, onde na BVH os objetos em si são segmentados e encapsulados por geometrias mais simples, comumente um cubo alinhado aos eixos, caracterizando uma *Axis-Align Bounding Box* (AABB) (HELD; KLOSOWSKI; MITCHELL, 1995) (BAREQUET et al., 1996) (NAYLOR; AMANATIDES; THIBAULT, 1990).

Na divisão espacial o particionamento das geometrias é inevitável, aumentando a profundidade da árvore e perdendo performance, além de não cobrir as primitivas de forma justa, dificultando os testes de colisão. Ao contrário, BVHs promovem limites mais estreitos além de se adaptarem melhor a geometrias complexas (KOCKARA et al., 2007). A raiz de uma BVH apresenta o nível mais grosseiro da representação do objeto, onde os próximos níveis representam mais detalhes do objeto, encapsulando de forma recursiva até que algum requerimento seja atendido nas folhas, normalmente contendo as primitivas (linhas e triângulos), que serão testados par a par (TESCHNER et al., 2005). Mesmo que os objetos não estejam colidindo, é possível que suas BHVs estejam, dessa forma, precisamos investigar mais a fundo para responder se os objetos estão ou não colidindo. Dessa forma, vamos descendo na árvore, onde ramos que não colidem são removidos de futuras considerações, até chegarmos às folhas da árvore (GOTTSCHALK, 2000).

1.3 Trabalhos Relacionados

A rápida e eficiente detecção de colisão entre malhas deformáveis possui um importante papel na computação gráfica, sendo especialmente útil em simulações cirúrgicas. Desse modo, esse é um tópico de pesquisa corrente, onde a evolução de placas gráficas junto de algoritmos robustos e escaláveis vem possibilitando avanços na área. Ao introduzirmos malhas deformáveis, adicionamos um desafio maior, recorremos a uma constante atualização do modelo de colisão, para corretamente representar a geometria no estado corrente do objeto, impossibilitando pré-calcularmos estruturas fixas e adicionando a necessidade de tratarmos auto colisões, normalmente negligenciadas em colisões com malhas fixas (LARSSON; AKENINE-MÖLLER, 2001).

1.3.1 Colisões

Uma das revisões feitas sobre a detecção de colisão as dividiu, como já visto, em particionamento do espaço e hierarquias volumétricas limitadas (FIGUEIREDO; MARCELINO; FERNANDO, 2002). Outras revisões sobre a detecção de colisão entre malhas deformáveis foram feitas por (TESCHNER et al., 2005) o qual mostra que um dos principais e primeiros métodos empregados foram BVHs, tanto utilizando AABBs quanto caixas delimitadoras orientadas (OBB) (BERGEN, 1997), evidenciando que, para uma BVH ser considerada eficiente, é necessário sua construção seja bem realizada, existindo três métodos principais para a construção de uma BVH, sendo eles: *Top-Down*, *Bottom-Up* e inserção (GOLDSMITH; SALMON, 1987). O método *Top-Down* divide de forma recursiva o objeto com base em alguma heurística até que algum limite seja atingido. Em contraponto, na estratégia *Bottom-Up*, partimos de primitivas e vamos construindo níveis superiores através da aglutinação dos nós desse nível até chegarmos à raiz. Para percorrer a árvore possuímos duas principais estratégias: *Depth-First Search* (DFS), que faz uma busca em profundidade, e *Breadth-First Search* (BFS), que realiza uma busca no atual nível da árvore primeiro (HABER; STAMMINGER; SEIDEL, 2000).

Mas tal modelo é dito como uma Detecção de Colisão Discreta (CCD), onde leva apenas em consideração se houve colisão nos quadros gerados, perdendo desse modo as colisões ocorridas no período de tempo entre quadros. Assim, surge outra classe para detecção de colisão, a Detecção de Colisão Contínua (DCC) (TANG et al., 2011).

O primeiro fator determinante é a geração da árvore, sendo que atualizar apenas parte da árvore a cada quadro pode ser até dez vezes mais eficiente que reconstruí-la (BERGEN, 1997). Como relatado por (LARSSON; AKENINE-MÖLLER, 2001), dependendo do nível que descemos na árvore, a estratégia textitTop-Down apresenta melhores resultados quando a árvore apresenta poucos níveis, da mesma forma, *Bottom-Up* oferece maior performance ao longo de uma busca em profundidade, dessa forma, propuseram uma técnica de construção e atualização da árvore de maneira discreta, em que apenas a metade superior da árvore é atualizada utilizando *Bottom-Up*, sendo os demais nós atualizados utilizando *Top-Down* apenas quando são alcançados por uma busca.

Outras revisões foram feitas ao longo dos anos. Assim como (WANG; CAO, 2021) observou, (CHOI; KIM; SUNG, 2017) propôs utilizar uma BVH *Bottom-up* com esferas como volumes delimitadores, se preocupando com os níveis de detalhe do contato, construindo uma BVH mais compacta, que aglutina as esferas na *Broad-Phase*. Já (LIANG et al., 2018) propôs um método utilizando BVHs para detecção de colisão em uma simulação de folhas de trigo, que possue melhor tempo de inicialização e reconstrução após uma intersecção. Já (KIM et al., 2019) propôs um novo método para

detecção de colisão entre objetos e tecidos, utilizando uma BVH com esferas como volumes delimitadores. E (ELOE et al., 2014) propôs um novo método para criação da BVH, utilizando um grafo dual, o que agrupa as primitivas utilizando o grafo dual, fazendo-as estarem próximas no espaço e, assim, resultando em um agrupamento mais eficiente e uma árvore mais otimizada.

1.3.2 Implementação em Simuladores

Nessa segmento, focaremos em como simuladores de colonoscopia utilizaram essas técnicas. (YI et al., 2006) utiliza um algoritmo de particionamento do espaço realizando uma pesquisa por BFS, com adaptações para uma detecção de colisão contínua. Em (FRANCE et al., 2005), é utiliza uma cadeia de esferas na *broad-phase* tanto do colonoscópio quanto do cólon para lidar com as colisões. (KORZENIOWSKI et al., 2016) utiliza uma BVH com AABBs como volumes delimitadores para colisão entre o cólon e o endoscópio.

Dessa forma, vemos que não há técnicas para detecção de colisão específicas que atendam o cenário ocorrido em uma simulação de colonoscopia.

2 METODOLOGIA

Visto que possuímos completo controle sobre nossa cena, a nova técnica proposta nesse trabalho se caracteriza em se aproveitar da situação encontrada em uma simulação de colonoscopia, para aplicar otimizações nesse cenário. Nos apropriamos de um cenário específico em que apenas lidamos com malhas cilíndricas, criando assim uma malha de colisão otimizada para os nossos modelos.

Como vimos anteriormente nas técnicas utilizadas na *Broad-Phase*, a melhor técnica para simulações dinâmicas é uma árvore BVH, a qual fornece dinamicidade na atualização da mesma. Assim, utilizamos AABBs estritas montadas a partir de duas das primitivas da técnica apresentada. Isso foi feito em uma BVH construída de forma *Bottom-Up* a cada quadro. Já na *Narrow-Phase* utilizamos nosso algoritmo para testarmos o par de primitivas, gerando por fim os vetores de colisão desses pares. Um dos principais pontos do algoritmo está na interpolação entre as primitivas, o que nos dá a habilidade de aumentarmos de forma dinâmica o grau de precisão do nosso algoritmo para regiões específicas.

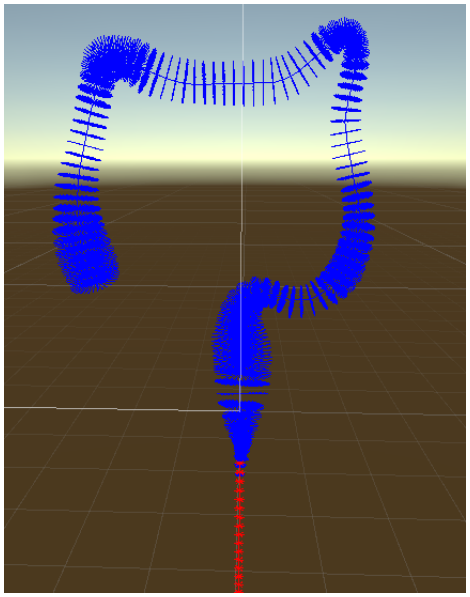

Figura 4 – A figura mostra uma visão geral da técnica, mostrando os pontos a serem abordados

2.1 Generalização

Como vimos, a técnica apresentada utiliza uma malha adicional para detecção de colisão, suportando N objetos. Nossa malha de colisão é constituída pela discretização de A pontos de controle, nossas primitivas, que são geradas a partir da linha média de nosso objeto. Os quais possuem B vetores por ponto de controle, que ligam o ponto de controle até as paredes da malha como é mostrado na Figura 5.

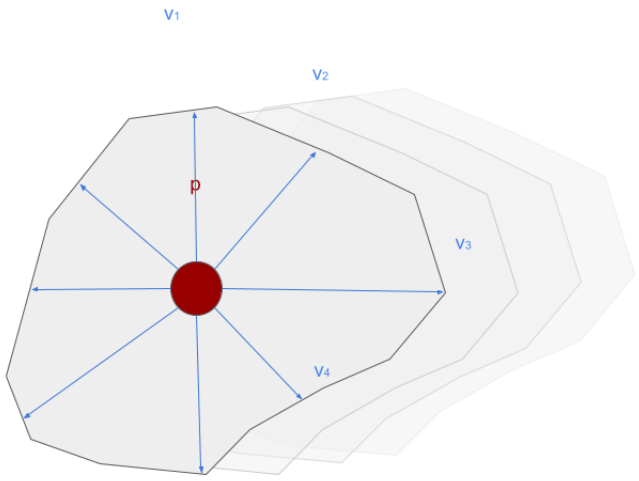

Figura 5 – A figura ilustra uma visão transversal da malha de colisão de seus componentes

Dessa forma, compomos o modelo de colisão pelas seguintes três estruturas de dados: uma lista contendo as posições dos pontos de controle em coordenadas globais, sendo cada entrada da lista um ponto de controle. Para cada ponto de controle, há duas listas de coordenadas, sendo a primeira uma lista de vetores que representam as direções radiais, e a segunda a posição absoluta dos vetores obtidos a partir da soma vetorial do ponto de controle e do vetor radial. Essas estruturas podem ser observadas na Figura 6

2.1.1 *Broad-Phase* Check

Assim passamos para primeira etapa do algoritmo. A *Broad-Phase* testa par a par nossos N objetos. Começamos recebendo duas malhas de colisão, onde o primeiro passo é montar as folhas da nossa árvore de ambos modelos. Fazemos isso gerando uma AABB para cada par de pontos de controle com seus vetores radias, como vemos na Figura 7:

Dando continuidade, construímos nossa árvore BVH de forma *Bottom-Up*, a partir de dois nós, criamos um nó superior que encapsula os inferiores. Fazemos isso até possuirmos apenas um nó que contém todos os outros, o qual chamamos de raiz da árvore.

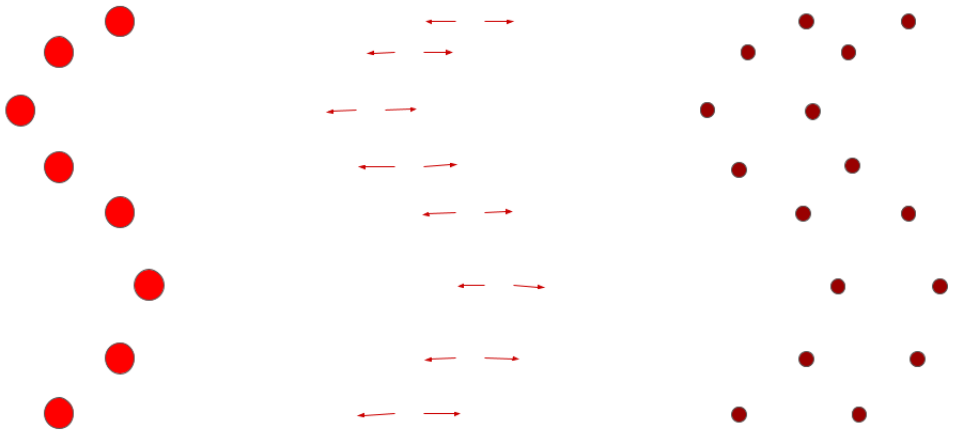

Figura 6 – A figura ilustra a decomposição das 3 estruturas utilizadas pelo algoritmo, lado a lado

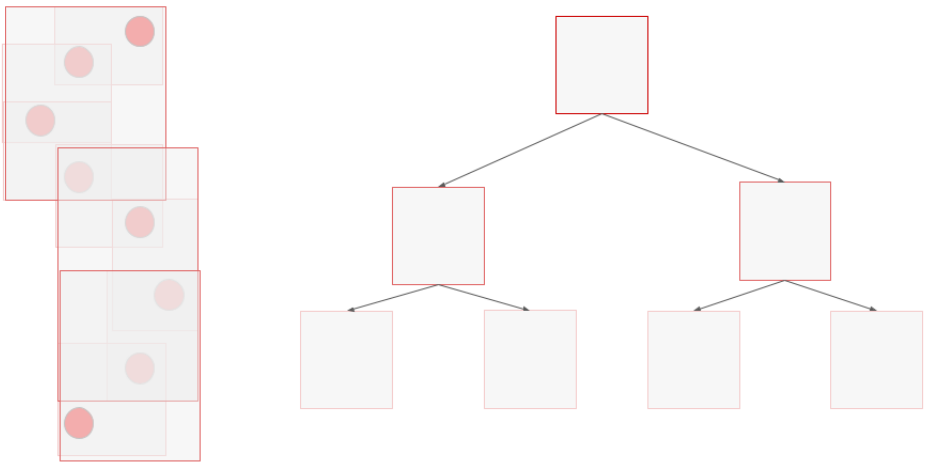

Figura 7 – A figura ilustra a composição das AABBs e sua estrutura em árvore binária

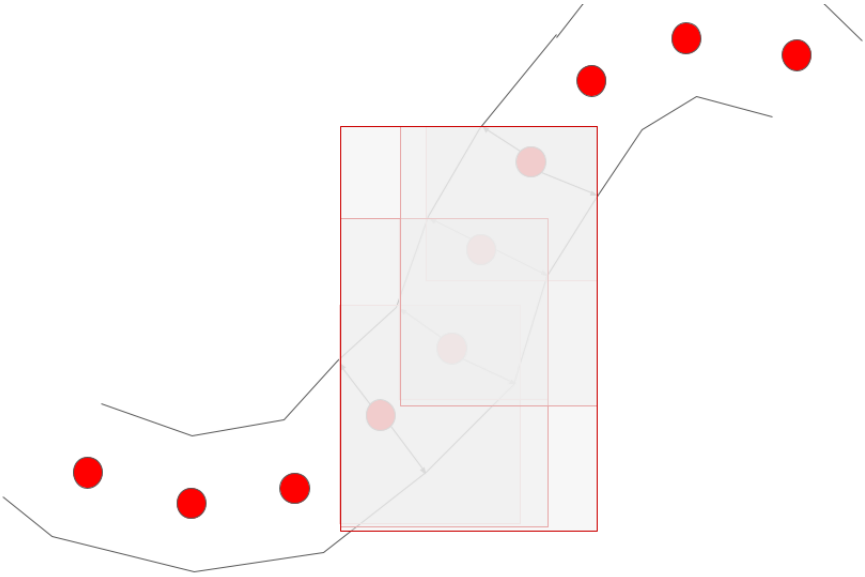

Figura 8 – A figura ilustra a criação da árvore BVH no contexto de uma simulação, mostrando seus limites

Dessa forma, passamos para checagem dos candidatos. Fazemos a checagem com uma *Depth First Search* entre as duas malhas, começando da raiz das árvores. Testamos se há uma intersecção entre as AABBs dos nós e descemos a árvore enquanto houver intersecções. Quando não há intersecção entre as AABB, esses candidatos são descartados.

2.1.1.1 Splines

Quando chegamos nas folhas da árvore, possuímos um par de nós candidatos, nos quais é feita a interpolação dos pontos de controle e uma interpolação linear dos vetores radiais, como é possível ver na Figura 9. A interpolação é uma estimativa da posição de outros pontos entre os conhecidos, com as informações que possuímos no momento. Podemos gerar x pontos entre os candidatos, visto que esse é um parâmetro parametrizado, nos dando uma ferramenta de controle da precisão e granularidade do algoritmo.

Vimos que utilizar a interpolação entre os pontos de controle nos dá uma maior granularidade nas próximas etapas da detecção de colisão. Possuímos agora muito mais informações sem o custo de ter que carregar essa informação e atualizá-la. Mas há inúmeras formas de fazermos a interpolações de pontos, cada uma com suas características intrínseca e objetivos próprios, e é necessário fazer uma análise sobre como fazer essa interpolação para atingir os objetivos específicos da técnica.

A forma mais fácil de interpolarmos dois pontos no espaço é através de uma interpolação linear (KUFFEL; KENT; IRWIN, 1997).

$$P(t) = (1 - t)P_0 + tP_1 \tag{1}$$

Figura 9 – A figura ilustra a interpolação entre os pontos de controle juntamente com a criação da AABB

Podemos ver P_0 e P_1 como pontos tridimensionais e P_t um ponto interpolado entre P_0 e P_1. Com isso, obtemos um resultado como visto na Figura 10 que ilustra a interpolação linear com cores.

Figura 10 – Figura mostrando uma interpolação linear no espectro de cor de vermelho à amarelo. Imagem retirada do blog Harmonic Code `https://harmoniccode.blogspot.com/2011/04/bilinear-color-interpolation.html`

Queremos que nossos pontos de controle sigam a curvatura do modelo. Uma forma fácil de conseguirmos isso seria levarmos em conta um terceiro ponto P_2. E fazermos uma interpolação linear também entre P_1 e P_2 e, com isso, fazermos outra interpolação linear entre os resultados, obtendo dessa forma o que chamamos de uma curva de Bézier Quadrática, como vimos na Figura 11.

Temos um resultado agora que segue uma curva. Da mesma forma, podemos seguir levando em conta mais pontos e fazendo interpolações lineares neles para obtermos curvas mais precisas ao aumentarmos o grau (FARIN, 2006). Entretanto, temos alguns problemas nessa abordagem. Dessa forma não possuímos um controle preciso de por onde a curva irá passar, ou um controle local sobre os pontos de controle sob os quais queremos interpolar, como podemos ver na Figura 12. A curva não passa pelos nossos pontos de controle P_1 e P_2, dizemos assim que os pontos não estão interpolados. Com uma curva de Bézier de maior grau, isso continua sendo verdade, a curva apenas interpola o primeiro e o último ponto de controle.

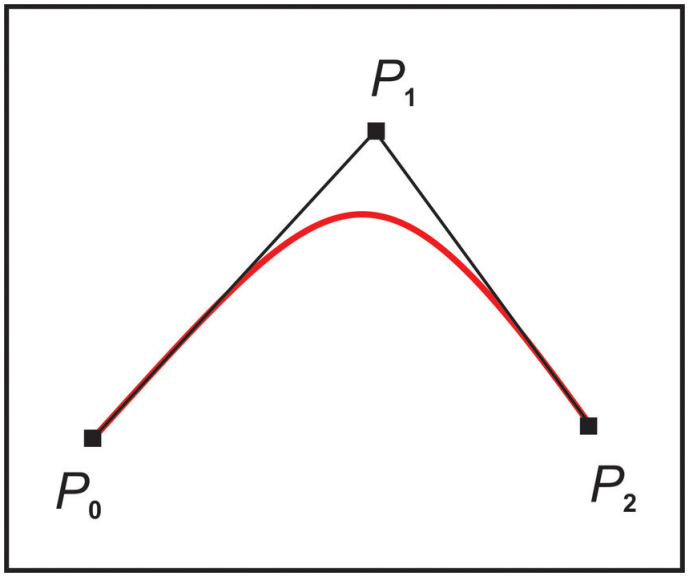

Figura 11 – A figura ilustra uma curva de Bézier Quadrática com 1 ponto de controle. Imagem retirada de (STARIBRATOV; MANOLOVA, 2022)

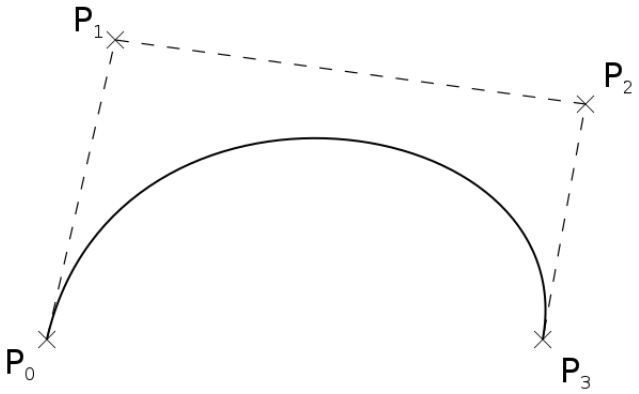

Figura 12 – A figura ilustra uma curva de Bézier Cubica, com 2 pontos de controle. Imagem retirada da página oficial da Wikipedia sobre curvas de Bézier `https://en.wikipedia.org/wiki/B%C3%A9zier_curve`

Isso leva a uma curva degenerada e deslocada de nosso objeto inicial, com um alto custo computacional para se fazer a cada iteração (ECK, 1993). Do ponto de vista computacional, para solucionar essa equação precisamos resolver uma multiplicação de matrizes $A_{n,n} \times B_{n,n}, n \in \mathbb{N}$. O que precisamos é de uma curva que passe por todos pontos de controle e siga a forma geral do nosso objeto, sua curvatura, de uma forma que esteja preenchendo os espaços onde não temos informação, sendo estável computacionalmente e com baixo custo para se calcular.

Podemos nos livrar de alguns desses problema com uma outra abordagem. Em vez de termos apenas um polinômio de alto grau, conseguimos fazer a junção de polinômios de menor grau em cada sessão da nossa curva, chamamos essa abordagem de *Splines*. Nessa abordagem não precisamos abandonar as curvas de Bézier, mas sim utilizarmos uma junção de curvas de Bézier cúbicas (ordem 3) com o ponto de controle final compartilhado com o inicial do próximo segmento. Chamamos essa abordagem de *Bézier Spline*. Dessa forma, obtemos o controle local da nossa curva, além de ser numericamente estável e fácil de se calcular como vemos na Figura 13.

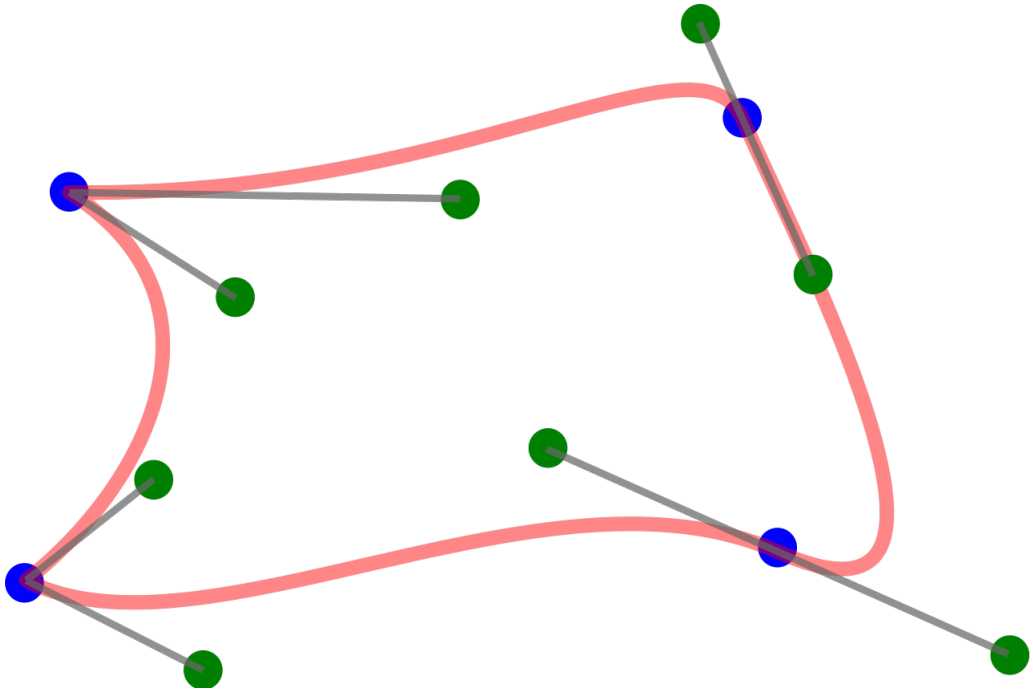

Figura 13 – A figura ilustra uma Bézier Spline, com seus pontos interpolados em azul e os pontos de controle em verde. Imagem retirada da página oficial de Bézier Splines da Wikipedia `https://en.wikipedia.org/wiki/Composite_B%C3%A9zier_curve`

Mas ainda possuímos alguns problemas nessa abordagem. Nossa curva apenas interpola metade dos nossos pontos de controle, além de precisarmos testar a continuidade dessa curva.

Quando falamos de *Splines*, possuímos graus C_n de continuidade por não se tratar do mesmo segmento de curva. C_0 é o grau mais básico de continuidade, e a única restrição é os segmentos compartilharem um mesmo ponto de controle no fim e inicio de

segmentos distintos, que chamamos de ligação, para ter esse grau de continuidade. Dessa forma, dizendo que as posições dessa curva são contínuas. Possuímos continuidade C_1 quando a 1ª derivada no final e início do ponto posterior dos segmentos são iguais. A 1ª derivada da posição tem seu significado explicado como a velocidade, logo a velocidade no final de um segmento precisa ser igual a velocidade no início do próximo (FADHEL; OMAR, 2021). Por conta disso, a curva acaba por não ter pontos de inflexão, parecendo de uma forma mais suave. Para graus de continuidade maiores, a derivada i no final de um segmento precisa ser igual à derivada i no início do próximo. Para ser C_{n+1} contínuo é necessário atender aos critério de C_n também.

$$A^i(t_{end}) = B^i(t_{start}), i \in 1...n \tag{2}$$

Dessa forma, conseguimos definir que nossa curva não pode ter pontos de inflexão para ser suave, logo ela precisa ser no mínimo C_1 contínua. Mas temos outro tipo de continuidade que precisamos nos atentar, a continuidade geométrica. A continuidade geométrica segue princípios parecidos, onde C_0/G_0 possuem o mesmo significado, são continuidades de posição. Já a continuidade G_1, é quando a tangente é contínua no final do segmento antecessor e início do próximo (LI et al., 2020). Essa continuidade tem um impacto direto na forma como a nossa curva se parece, tornando-a mais suave e sem curvas abruptas. Assim, conseguimos observar que, para obtermos uma curva suave que segue o formato do nosso objeto, precisamos uma curva C_1/G_1 contínua. Para isso, podemos utilizar uma propriedade para nos ajudar, onde uma continuidade $C_n \implies G_n$. Dessa forma, precisamos apenas garantir uma continuidade C_1 para nossa curva.

Uma forma de alcançar esse objetivo é utilizar um modelo de *Spline* conhecido como *Catmull-Rom Spline*, o qual é largamente utilizado em video games e possui características próximas das desejadas. O modelo da *Catmull-Rom Spline* permite um controle local dos pontos de controle, a interpolação de todos os pontos de controle e oferece uma curva suave e numericamente estável (CATMULL; ROM, 1974). Para utilizar esse modelo, precisamos de quatro pontos de controle, onde a interpolação ocorre entre os pontos centrais. Como estamos falando de uma *Spline*, aplicamos esse algoritmo em cada sessão da curva. Uma desvantagem de utilizarmos esse modelo de *Spline* é a impossibilidade de interpolarmos o primeiro e o último ponto de controle, de modo que esses pontos passam a ser ignorados na simulação de colisão. Uma forma de contornar esse problema é adicionando pontos de controle ao modelo, de modo a impedir que partes importantes da malha não sejam representadas na colisão.

Em definição deixando $\mathbf{P}_i = [x_i \quad y_i]^T$ definir um ponto. Para um segmento de curva $\mathbf{C}$ definido pelos pontos $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2, \mathbf{P}_3$ e a sequência de ligamentos t_0, t_1, t_2, t_3,

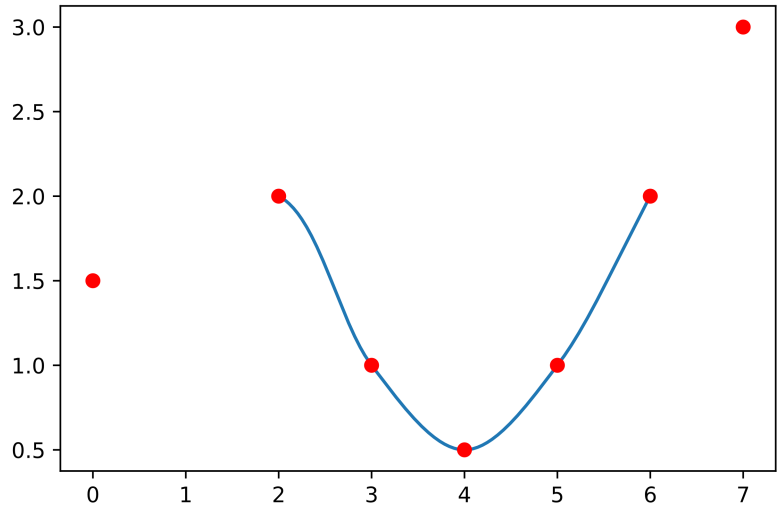

Figura 14 – A figura ilustra o uso do algoritmo da *Catmull-Rom* com sete pontos de controle e a curva resultante. Imagem retirada da página oficial da *Catmull-Rom* na Wikipedia `https://en.wikipedia.org/wiki/Centripetal_Catmull%E2%80%93Rom_spline`

a centripetal Catmull–Rom spline pode ser adquirida através de

$$\mathbf{C} = \frac{t_2 - t}{t_2 - t_1}\mathbf{B}_1 + \frac{t - t_1}{t_2 - t_1}\mathbf{B}_2, \tag{3}$$

$$\mathbf{B}_1 = \frac{t_2 - t}{t_2 - t_0}\mathbf{A}_1 + \frac{t - t_0}{t_2 - t_0}\mathbf{A}_2, \tag{4}$$

$$\mathbf{B}_2 = \frac{t_3 - t}{t_3 - t_1}\mathbf{A}_2 + \frac{t - t_1}{t_3 - t_1}\mathbf{A}_3, \tag{5}$$

$$\mathbf{A}_1 = \frac{t_1 - t}{t_1 - t_0}\mathbf{P}_0 + \frac{t - t_0}{t_1 - t_0}\mathbf{P}_1, \tag{6}$$

$$\mathbf{A}_2 = \frac{t_2 - t}{t_2 - t_1}\mathbf{P}_1 + \frac{t - t_1}{t_2 - t_1}\mathbf{P}_2, \tag{7}$$

$$\mathbf{A}_3 = \frac{t_3 - t}{t_3 - t_2}\mathbf{P}_2 + \frac{t - t_2}{t_3 - t_2}\mathbf{P}_3, \tag{8}$$

$$t_{i+1} = \left[\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}\right]^{\alpha} + t_i. \tag{9}$$

O sistema de equações descrito pode ser representado em forma matricial. Tal transformação traz as vantagens de otimizações na álgebra de matrizes, reduzindo o custo computacional. A forma matricial desse mesmo sistema de equações é representada como:

$$P(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \times \frac{1}{2} \times \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}. \tag{10}$$

Com a *Catmull-Rom*, temos mais um grau de liberdade com o α, onde α varia de 0 a 1 para a parametrização das ligações, e $i = 0, 1, 2, 3$ com $t_0 = 0$. O α controla a extrapolação da curva, sendo valores maiores se aproximando do comportamento de uma curva de Bézier de alto grau e valores menores de uma interpolação linear (YUKSEL; SCHAEFER; KEYSER, 2011b). É dado nomes a alguns valores de α. Para o valor de α =0.5 é dado o nome de *Centripetal Catmull–Rom Spline*. Já com α =0, a curva resultante é a padrão *Uniform Catmull–Rom Spline*. E com α =1, o resultado é a *Chordal Catmull–Rom Spline*. A Centripetal apresenta o melhor balanço nas interpolações, exatamente por possuir o valor central, assim conseguimos resultados que seguem bem a forma do objeto sem extrapolar (YUKSEL; SCHAEFER; KEYSER, 2011b).

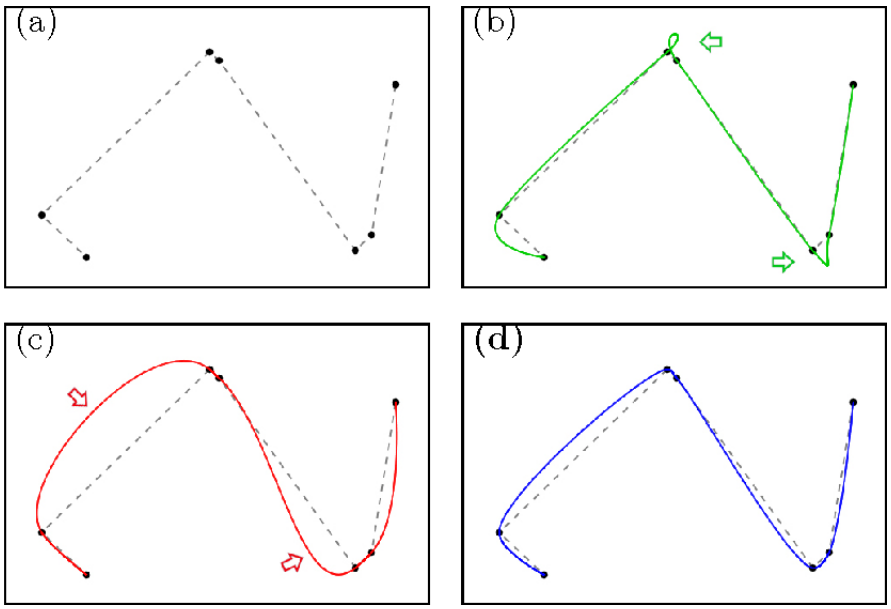

Figura 15 – A figura ilustra os três tipos de parametrização de α. $a)$ Linear. $b)$ Uniform. $c)$ Chordal. $d)$ Centripetal. Imagem retirada de (YUKSEL; SCHAEFER; KEYSER, 2011a)

Há outras opções de *Splines* que seguem a curvatura do nosso objeto, as quais são estável e produzam curvas suaves, possuindo ainda maior grau de continuidade como C_2. Apesar de permitirem maior grau de continuidade que a *Catmull-Rom*, os outros modelos de *Splines* impedem a interpolação dos pontos de controle, a qual é característica essencial em nosso trabalho. Podemos ver esse efeito na Figura 16.

Já para os vetores radiais, não é necessário que os vetores sigam a curvatura do objeto. Visto que o ajuste na posição dos pontos de controle já irá dar a correta posição dos vetores, podemos fazer uma interpolação deles com uma interpolação

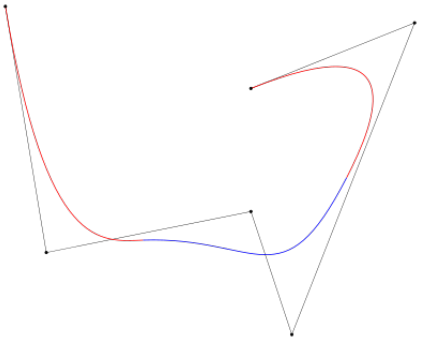

Figura 16 – A figura ilustra a curva resultando de uma B-Spline. Imagem retirada da página oficial da B-Spline na Wikipedia `https://en.wikipedia.org/wiki/B-spline`

linear. Assim precisamos apenas calcular a posição desses vetores como vemos na Figura **??**. Organizamos os vetores em ordem para os cálculos futuros. Como será visto, a interpolação desses vetores será necessária para apenas uma das malhas.

2.1.1.2 Saída da Broad-Phase

Conseguimos fazer a interpolação de i candidatos, onde cada um é composto por j pontos interpolados. No final, é gerada uma lista de proximidade contendo o par com cada ponto interpolado da malha a associado ao ponto interpolado mais próximo entre os interpolados da malha b e a distância entre eles. Conseguimos ter uma visão do resultado final da *Broad Phase* na Figura 17.

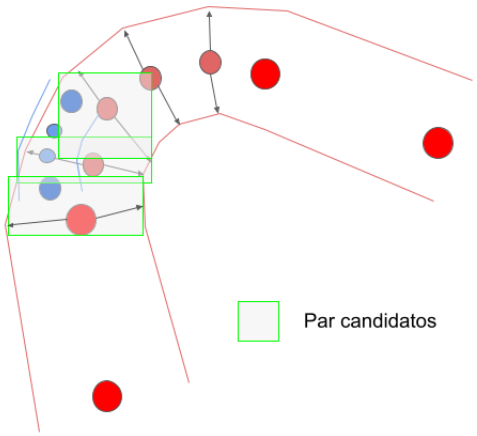

Figura 17 – A figura ilustra os pares candidato gerados ao final de *Broad-Phase*

```
1 std::vector<CollisionStructure* a, std::pair<float distance,
     CollisionStructure* b>>
```

Listing 2.1 – Possível implementação da estrutura resultante da Broad Phase

2.1.2 *Narrow-Phase Check*

Já na *Narrow-Phase*, como vimos, possuímos uma lista que relaciona cada ponto interpolado da malha a com outro ponto interpolado da malha b e a distância entre eles. Essa é nossa lista de pares candidatos. Precisamos agora realizar uma detecção de colisão entre as geometrias que obtemos, afim de checar uma intersecção entre a malha a e b. Para tal utilizaremos os pontos de controle e vetores radiais interpolados.

Já que possuímos dois pontos com seus respectivos raios, uma solução simples para o problema é fazer uma detecção de colisão entre esferas como vemos na Figura 18.

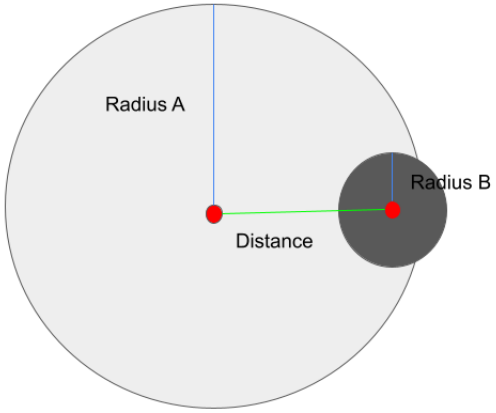

Figura 18 – A figura ilustra uma detecção de colisão entre duas esferas

Dessa forma estamos desperdiçando muitas das informações que possuímos, mas conseguimos analisar as informações que perdemos. Conseguimos efetivamente detectar a intersecção entre duas esferas com a seguinte equação.

$$RadiusA + RadiusB > DistanceBetweenSpheres \tag{11}$$

Com uma equação simples e rápida de calcular, resolvemos o problema. Mas perdemos nosso objetivo principal. Queremos que a malha se deforme e, com raios fixos não possuímos essa capacidade, estaríamos presos a valores fixos.

Com isso, precisamos de uma abordagem mais dinâmica, que leve em conta a deformação das malhas. Conseguimos isso testando o ponto de controle da malha b contra os vetores associados ao ponto de controle da malha a. Como o par proximidade nos garante os pontos interpolados mais próximos um do outro, temos a segurança que caso esteja acontecendo uma colisão com o ponto da malha b, é na seção do ponto da malha a, onde a malha a teve seus vetores radiais interpolados. Dessa forma, testamos o ponto de controle da malha b contra o conjunto de vetores

do ponto de controle da malha a. Podemos analisar de uma perspectiva transversal o problema, como vemos na Figura 19. Assim podemos testar nosso ponto contra cada par sequencial de vetores, descobrindo em qual seção de pares de vetores nosso ponto P_b está.

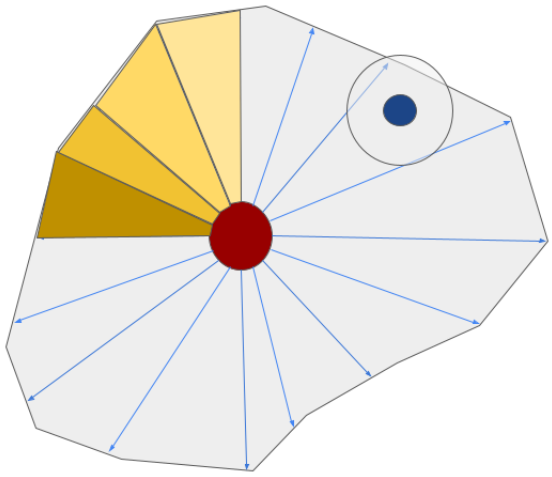

Figura 19 – A figura ilustra uma visão transversal de ambas malhas, e as seções dos vetores interpolados

Para melhor visualizar o algoritmo, vamos recortar a seção em que P_b está inserido e suas seções adjacentes. Agora iteramos por cada seção onde chamamos o par de vetores da seção que estamos testando de V_1 e V_2. Conseguimos gerar um vetor $\vec{w}$ que vai de V_1 a V_2 através de $\vec{w} = V_2 - V_1$, e outro vetor $\vec{z}$ que conseguimos através de $\vec{z} = V_1 - P_b$. Checamos primeiro se P_b está nas regiões adjacentes da que estamos testando. Conseguimos verificar se o ponto P_b está à esquerda da seção testada com a equação 12, que chamamos de c_1.

$$\vec{w} \cdot \vec{z} <= 0 \tag{12}$$

Verificando o ângulo entre os vetores $\vec{w}$ e $\vec{z}$ com o uso de uma propriedade do produto escalar. O produto escalar entre vetores pode ser visto como uma projeção de um vetor em outro, resultando em um número real. Caso o ângulo entre os vetores seja $\theta > 90$, ou seja, estejam apontando para direções opostas, o valor do produto escalar será negativo. Assim, conseguimos ver se o ponto de controle P_b está a esquerda da seção sendo testada. Caso a primeira verificação falhe, é feita a checagem se ele está à direita da seção em teste com a equação 13. Utilizamos outra propriedade do produto escalar, onde caso o produto escalar seja aplicado ao mesmo vetor, ele terá como resultado o tamanho desse vetor elevado ao quadrado. Como o produto escalar entre dois vetores demonstra a proporção de um vetor projetado em outro, conseguimos verificar a direção e distância do nosso ponto de controle em relação a

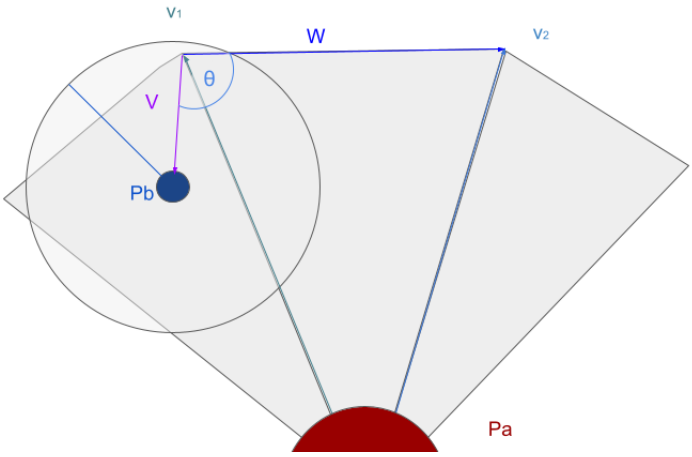

Figura 20 – A figura ilustra o cenário onde P_b está à esquerda da região de teste

região testada.

$$\vec{w} \cdot \vec{w} \leq \vec{w} \cdot \vec{z} \tag{13}$$

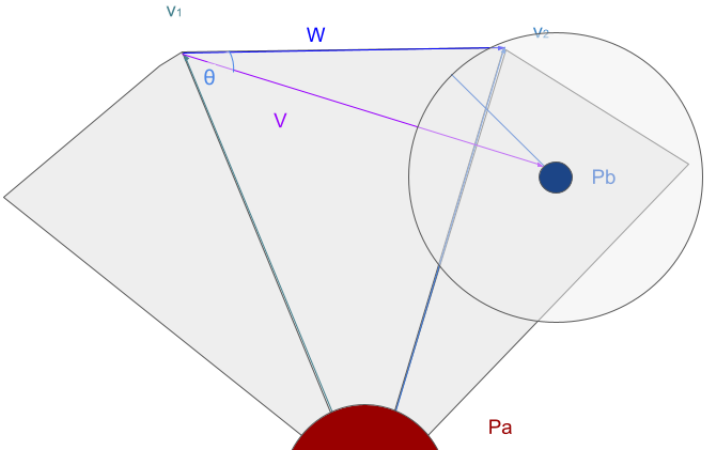

Figura 21 – A figura ilustra o cenário onde P_b está à direita da região de teste

Se nenhum caso anterior se enquadre, conseguimos deduzir que o ponto de controle P_b se encontra na região que estamos testando. Dessa forma, conseguimos saber a região em que o ponto de controle está. É necessário gora testar se há colisões e gerar os vetores de colisão. Com as regiões adjacentes, temos um método fácil, para região adjacente à esquerda, a distância entre o ponto de controle P_b e a região em teste é exatamente o tamanho do vetor $\vec{w}$. Já a distância na região adjacente à direita, é o tamanho de um novo vetor $\vec{x} = P_b - V_2$, que é a distância entre o ponto

P_b e o segundo vetor da região. Comparamos essa distância com o raio da malha b, onde:

$$colisao = distancia < raio \tag{14}$$

E o vetor colisão caso esteja na região esquerda ou direita respectivamente será $\vec{w}$ ou $\vec{x}$.

Quando o ponto P_b está na região em teste, é necessário calcular o vetor colisão, o que é feito com interpolação linear. Conseguimos calcular a proporção entre V_1 e V_2, em que o ponto P_b está com a equação 15.

$$p = \frac{\vec{w} \cdot \vec{z}}{\vec{w} \cdot \vec{w}} \tag{15}$$

É feita uma interpolação linear entre V_1 e V_2 com a equação 16.

$$P = V_1 + p \cdot \vec{w} \tag{16}$$

Obtemos o vetor colisão agora com $\vec{c} = P - P_b$. Dessa forma, a menor distância do ponto de controle até a região em teste com o tamanho de $\vec{c}$. Conseguimos verificar se há colisão com a mesma equação 14, onde checamos se o tamanho de $\vec{c}$ é menor que o raio de P_b.

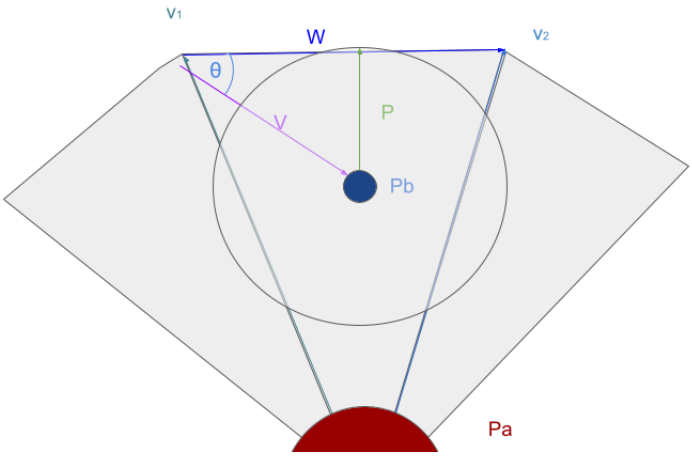

Figura 22 – A figura ilustra o cenário onde P_b está na região de teste

2.1.3 Algoritmo Final

Temos uma visão em alto nível de todos algoritmos com o pseudo código apresentado abaixo. Esse código é executado a cada iteração da simulação.

Algorithm 1 Collision Detection - Input Collision Models (a, b)

```
1: Construct Leaves for a
2: Construct Leaves for b
3: RootA = Build Bottom-Up Tree for a
4: RootB = Build Bottom-Up Tree for b
5: Candidates = GenerateCandidates(RootA, RootB)
6: for All Candidates do
7:     Check Collision between candidates
8:     collisions = Generate collision vector
9: end for
10: return collisions
```

Algorithm 2 GenerateCandidates ($RootA$, $RootB$)

```
1: procedure GENERATECANDIDATES(RootA, RootB)
2:     if not RootA intersects RootB then
3:         return
4:     else if RootA = LEAF and RootB = LEAF then
5:         InterA = Interpolate RootA
6:         InterB = Interpolate RootB
7:         Interpolate Vectors RootB
8:         Candidates.Add(InterA, InterB)
9:     else
10:        if RootA = LEAF then
11:            GENERATECANDIDATES(RootA.Left, RootB)
12:            GENERATECANDIDATES(RootA.Left, RootB)
13:        else
14:            GENERATECANDIDATES(RootA, RootB.Left)
15:            GENERATECANDIDATES(RootA, RootB.Right)
```

3 IMPLEMENTAÇÃO

A técnica apresentada se mostra como parte de um sistema maior, invés de um produto final. A técnica foi implementada na linguagem C++20, tendo como única dependência a biblioteca Eigen-3, utilizada para álgebra vetorial e matricial das equações utilizadas. O objetivo da técnica é de funcionar como uma biblioteca utilizada em um simulador, especificamente um simulador para colonoscopia, oferecendo uma interface para criação das malhas, gerenciamento dos pontos de controle e vetores, controle da precisão através do número de pontos interpolados e a checagem de colisão entre as malhas instanciadas. A implementação da técnica teve por objetivo central a integração e utilização em um simulador já existente.

3.1 Simulador de Colonoscopia

No contexto do projeto de pesquisa, junto com outros pesquisadores, foi desenvolvido um simulador de colonoscopia. Devido à grande complexidade de motores gráficos atuais, os quais possuem inúmeros componentes a dar suporte (ANDRADE, 2015) a estratégia abordada foi criar o simulador como um *Framework* a ser integrado em um motor gráfico já consolidado. O motor gráfico escolhido foi a *Unity*, um dos maiores e mais consolidados motores gráficos, com anos de histórico e ferramentas para o desenvolvimento de aplicações (HAAS, 2014), além de um grande suporte a integração de código externo escrito em C++ e a facilidade de integração ao motor com uma vasta documentação. A técnica apresentada foi implementada seguindo os mesmo princípios, com o objetivo de ser integrada ao simulador criado dentro da *Unity*, exportando uma interface manipulável. Podemos ver a relação dos componentes na Figura 23.

3.1.1 Simulador de Física

Devido a performance, simplicidade e robustez o simulador utiliza como motor físico uma implementação do *Position Based Dynamics* (PBD), mais especificamente sua versão extendida XPBD. No PBD os objetos são modelados como partículas liga-

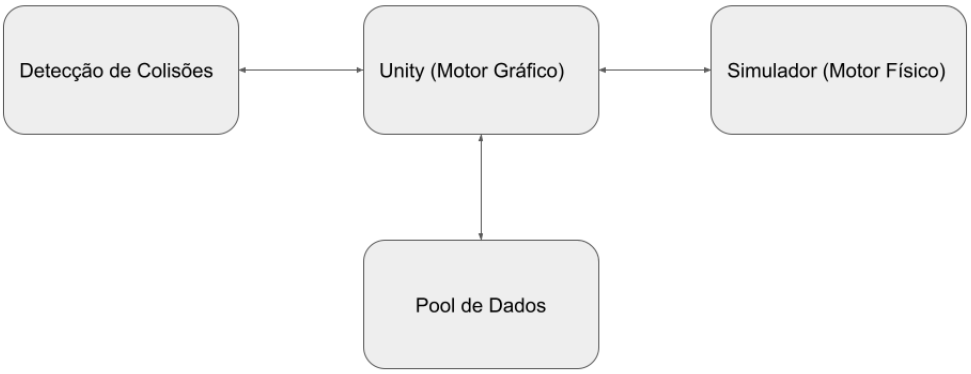

Figura 23 – A figura apresenta um diagrama de blocos da implementação e integração da técnica de detecção de colisão

das por restrições. Essas restrições são resolvidas de forma interativa a cada quadro da simulação. Apesar de ser uma solução aproximada diferente da física simulada de alguns outros *Frameworks*, o XPBD é muito utilizado em simulações físicas para uso médico, especialmente para treinamento (FANG et al., 2023).

3.1.2 Integração

Com um simulador de colonoscopia possuímos duas principais estruturas, o cólon e o endoscópio. Dessa forma, temos duas malhas cilíndricas, que podem ser generalizadas igualmente, apresentando um caso cenário ideal para técnica apresentada. Permitindo dessa forma, uma generalização ainda maior entre a técnica de detecção de colisão e o motor físico. Para construção do comportamento do endoscópio é utilizada uma técnica para simulação de hastes flexíveis, a *Cosserat Rod*, a qual foi implementada utilizando o XPBD com sucesso por **??**. A teoria da *Cosserat Rod* é um modelo matemático que descreve o comportamento de hastes flexíveis ao dobrar, esticar e torcer. Onde sua implementação no XPBD utiliza uma linha central de cilindros conectados com restrições em dobrar, esticar e torcer. A integração da *Cosserat Rod* ao sistema de colisões também se dá de forma intuitiva, visto que é controlada por uma linha central com pontos de controle com raios, os quais são representados por cilindros sólidos. Dessa forma, é mantida a homogeneidade dos dados através dos sistemas. Mantemos o mesmo *Framework* para ambos modelos, conseguimos manter a robustez numérica, aumentando a convergência do sistema.

Conseguimos ver na Figura 24 a malha de colisão de um cólon no simulador. A malha é atualizada a cada quadro com as atualizações feitas pela resposta de colisão geradas no XPBD.

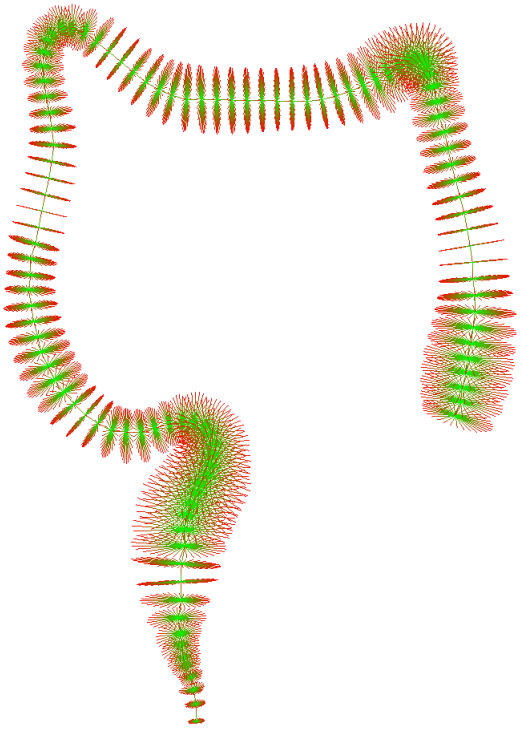

Figura 24 – A figura ilustra uma malha de colisão de um cólon com as estruturas do detecção de colisão

Podemos ver a malha do objeto na Figura 25 e observar como a malha de colisão é ligada estritamente a cada vértices da malha. Dessa forma conseguimos manter uma relação $1 \longleftrightarrow 1$ entre os vértices e os vetores ligados aos pontos de controle.

Algumas simplificações e otimizações advém de manter essa relação. Conseguimos dessa forma fazer um pool de dados entre a detecção de colisão, *Unity* e o simulador. Como é necessário atualizar as informações a cada quadro, muitas informações precisam ser copiadas entre os sistemas. Ao manter essa relação, conseguimos fazer a cópia de apenas um bloco de memória com as posições de todos vetores. Junto a isso, o motor de física do simulador também utiliza os pontos de controle, possibilitando que todas informações sejam feitas em apenas uma cópia de memória. Assim, conseguimos reduzir um dos gargalos da integração, visto que cópias em memória tendem a ser a parte mais custosa de um algoritmo (NITZBERG; LO, 1991).

Na construção das malhas de colisão, precisamos da informação dos pontos de controle e vetores vetores radiais. Apesar de parecer mais simples gerar as informações da colisão com base na malha do objeto, optamos por fazer o inverso. Modelamos a malha de um cólon em um software externo, onde utilizamos um algoritmo para calcular a linha média e vetores radiais ligados a todos vértices, gerando um arquivo final com todas informações. Dessa forma, carregamos esse arquivo na inicialização do algoritmo, onde instanciamos a malha de colisão que posteriormente utilizamos para gerar a malha no motor gráfico. Temos dessa forma a garantia da relação entre

Figura 25 – A figura mostra a malha do objeto associada a malha de colisão da Figura 24

ambas, mantendo a consistência entre os dados.

4 RESULTADOS

Possuímos uma técnica especializada em um grupo de cenários presentes em simulações médicas, e aplicada especificamente em uma simulação de colonoscopia. Conseguimos gerar uma simulação estável e performática com suporte à deformações na malha. Conseguindo gerar a formações de laços no endoscópio, que é um dos principais desafios no procedimento de colonoscopia.

Assim obtemos uma aplicação funcional e integrada, a qual consegue facilmente ser estendida e aplicada em diversos contextos. Medimos sua aplicação específica em um simulador de colonoscopia, integrado a um motor de física. Obtendo dessa forma uma simulação coesa e em tempo real, com crescimento linear em relação a precisão da técnica. Podemos ver uma simulação completa na Figura 26.

Figura 26 – A figura mostra a visão final do procedimento no simulador

Com a visão da ponta do endoscópio sendo mostrada na Figura 27.

4.1 Performance

A implementação e integração da técnica conseguiu reproduzir a simulação com sucesso. Podemos avaliar a técnica sobre alguns parâmetros como facilidade de integração, generalização, robustez numérica mas o mais importante é a performance.

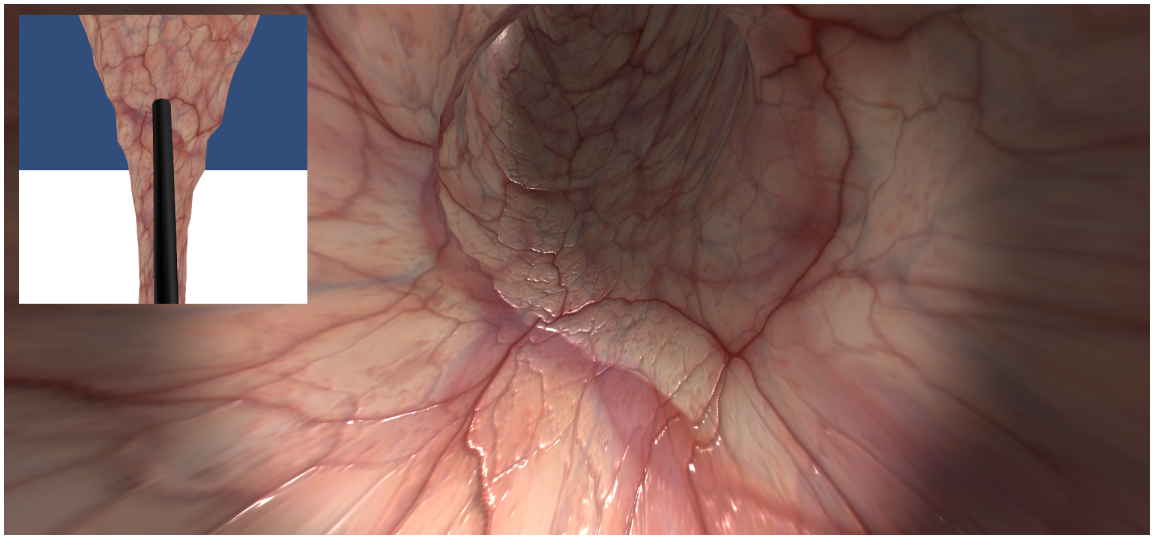

Figura 27 – A figura demonstra uma visão limpa do endoscópio

Avaliamos a técnica ao decorrer de uma simulação completa, coletando dados sobre o número de colisões detectadas e o tempo levado em cada quadro. Conseguimos ver na Figura 28 o gráfico do tempo tomado pelo algoritmo ao curso de uma simulação completa de colonoscopia.

Junto a isso, conseguimos ver na Figura 29 como o número de colisões se comporta no decorrer da simulação.

Podemos ver uma clara relação entre ambos gráficos, onde há uma relação proporcional entre o tempo levado para o cálculo em um quadro e o número de colisões detectadas nesse quadro. Também observamos um aumento em ambos gráfico ao decorrer da simulação. Visto que a área de contato inserido dentro do cólon pelo endoscópio aumenta ao passar da simulação como visto na Figura 30 é esperado que o número de colisões aumente de forma linear também.

Observamos uma certa estabilidade no número de colisões detectadas ao longo da simulação, junto com um comportamento oscilatório, onde há uma variação linear com constantes picos e vales. Explicamos esse comportamento ao olharmos no gráfico do número de colisões detectadas, e vermos algo muito similar, com menor variação total mas um comportamento mais oscilatório. Vemos a causa disso na própria natureza da detecção de colisão. Quando o endoscópio bate na parede do cólon e a colisão é detectada, é criada uma restrição de distância entre as malhas, afim de penalizar tanto o endoscópio quanto o cólon. Isso tem o efeito de separá-los, e logo em seguida colidirem novamente. Assim, não é surpresa vermos esse comportamento.

Ao final dos gráficos, observamos um aumento repentino no número de colisões detectadas, mas não um aumento tão significativo no tempo levado. Isso demonstra robustez na técnica, mantendo a performance em alta demanda.

Conseguimos observar também o tempo tomado para cada ocorrência de um certo número de colisões como vemos na Figura 31. Apesar desse gráfico ser um pouco

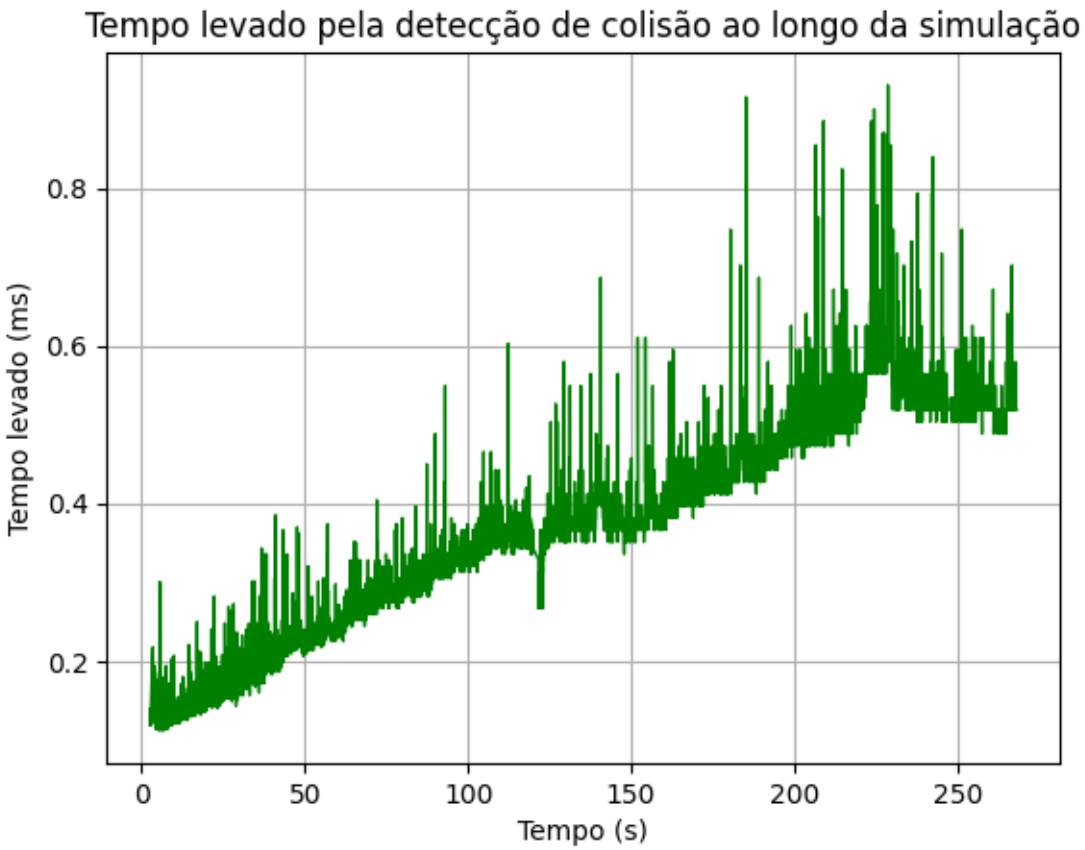

Figura 28 – A figura mostra o gráfico do tempo levada pela técnica ao decorrer de uma simulação

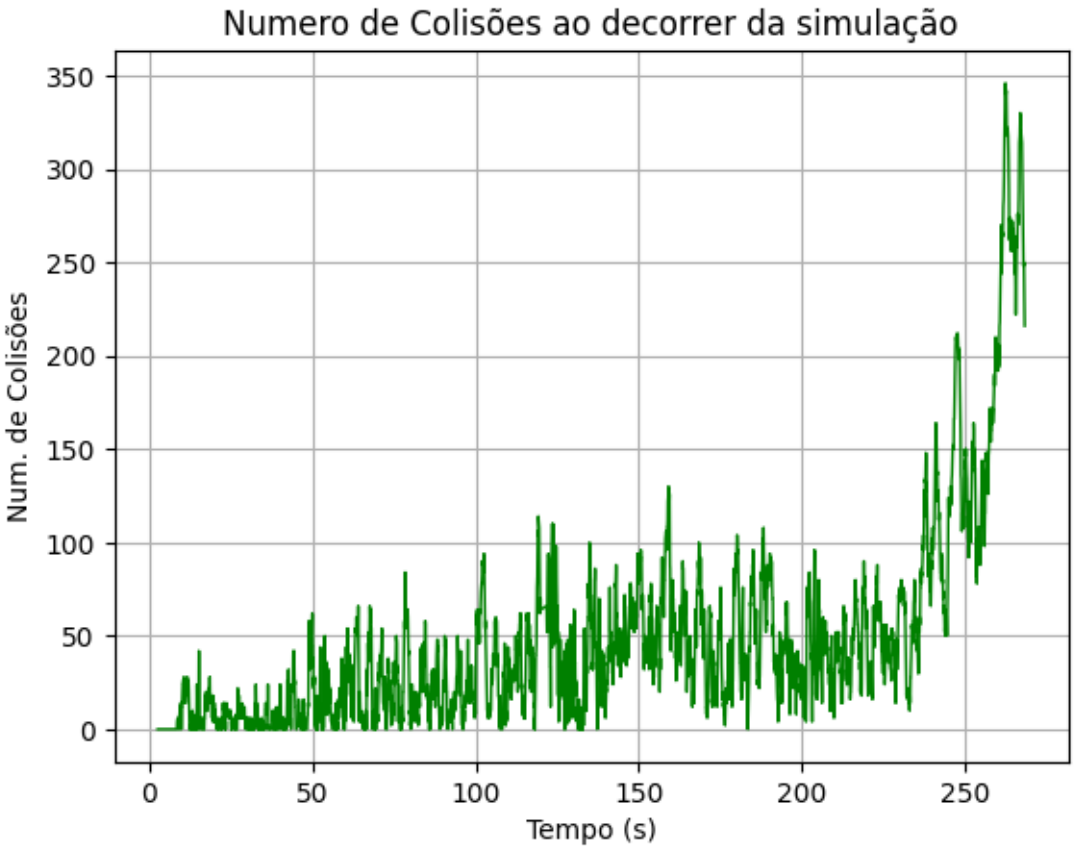

Figura 29 – A figura mostra o gráfico do número de colisões detectadas ao decorrer de uma simulação

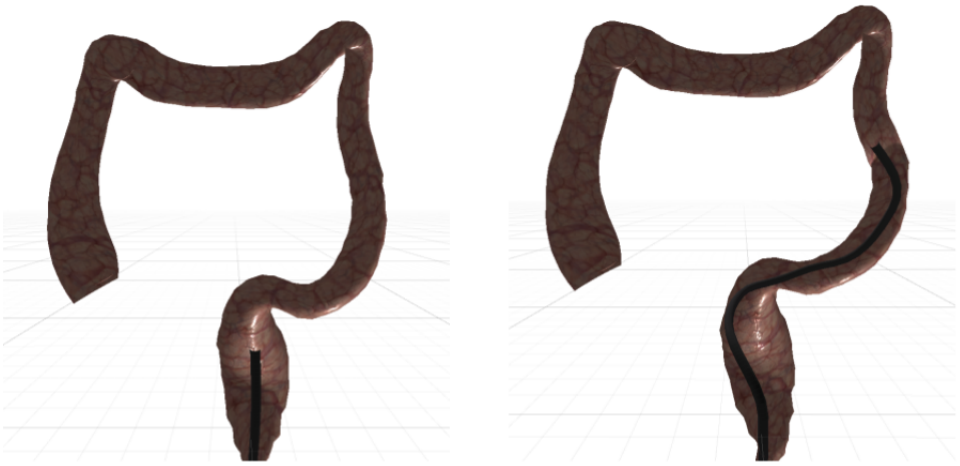

Figura 30 – A figura mostra como a área de contato entre o endoscópio e o cólon aumenta conforme a simulação avança

mais difícil de ser interpretado, conseguimos ver que o crescimento de tempo é no máximo linear de acordo com o número de colisões apontadas.

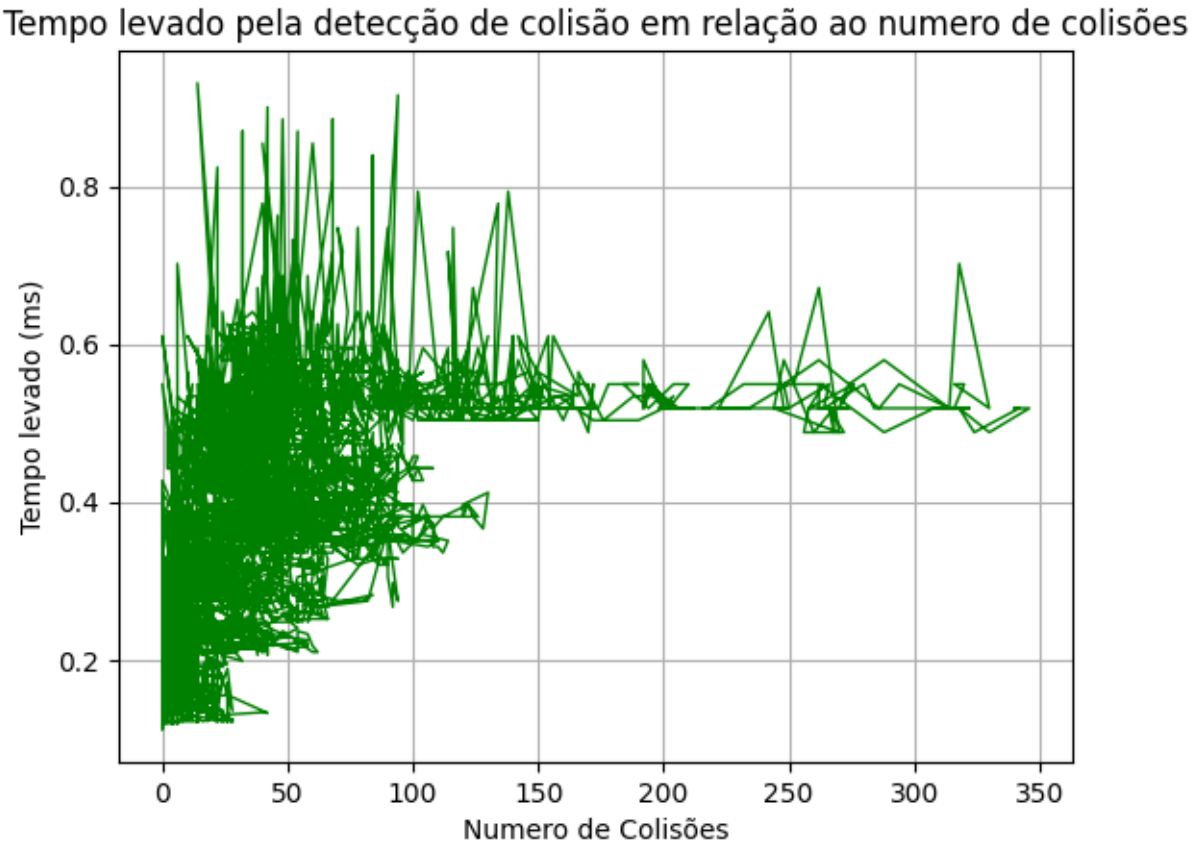

Figura 31 – A figura mostra um gráfico da relação do número de colisões detectadas e o tempo levada pelo algoritmo

4.2 Falhas

Apesar de bons resultados em relação á performance e robustez da simulação, durante os testes com o simulador, foi apontado um erro na *Broad-Phase* e *Narrow-Phase* do algoritmo. A ideia do algoritmo se baseia em malhas cilíndricas contidas entre si, mas age de forma mais geral. De forma que caso as malhas não estejam contidas uma na outra, não há forma do algoritmo saber e penalizá-las, logo o motor físico não consegue gerar as restrições. Dessa forma, a técnica consegue ser utilizada em outros contextos, mas perde eficácia em sua principal função. Como vemos na Figura 32, apesar do endoscópio se encontrar fora do cólon, não há detecções de colisão com a parte que está fora.

Apesar da descrição da técnica não aparentar ter falhas, na integração com um simulador, um cenário mais complexo expõe as fraquezas da integração. Como um simulador físico necessita resolver sistemas de equações diferenciais em uma ordem não específica, ao tentar acomodar as próprias restrições uma das soluções possíveis

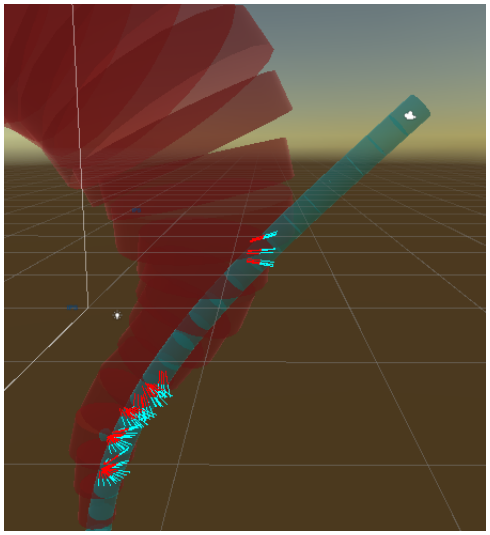

Figura 32 – Na figura conseguimos ver dentro do simulador o endoscópio penetrando o cólon e não sendo punido

será fazer o endoscópio penetrar no cólon muito mais do que devia. Esse cenário gera uma simulação degenerada no estado atual, sem volta possível.

4.2.1 Falha na *Broad-Phase*

Como vimos, fazemos a divisão espacial com AABBs estritas, o que significa que elas delimitam o objeto sem folga. Assim, no momento que uma parte do endoscópio está completamente fora do cólon, não há mais formas de corrigir, pois essa parte não passará no teste de candidatos, visto que sua AABB não intersecta nenhuma AABB do cólon. Conseguimos observar esse problema na Figura 33, onde no último segmento da malha azul a AABB não intersecta a AABB da malha vermelha. Dessa forma esse segmento sequer será um candidato.

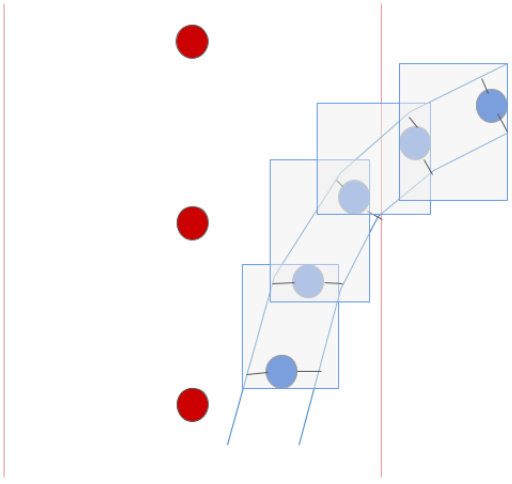

Figura 33 – A figura ilustra a falha na *Broad-Phase*

4.2.2 Falha na *Narrow-Phase*

Apesar do erro na *Broad-Phase* ser relevante e precisar ser resolvido o real motivo do endoscópio conseguir sair do cólon se dá ao fato do algoritmo não penalizar as estruturas que estão fora do cólon. Mesmo que a parte que está fora do cólon passasse pela *Broad-Phase*, o algoritmo não conseguiria penalizar. Como vimos na *Narrow-Phase*, associamos cada ponto interpolado do candidato de uma malha a outro de outra malha. Continuamos por testar cada par de vetores radiais da segunda estrutura, e testamos isso através do raio da primeira estrutura. Essa abordagem enfrenta o mesmo problema que na *Broad-Phase*, uma vez que o segmento está fora, não é possível penalizá-lo. Observamos esse efeito na Figura 34.

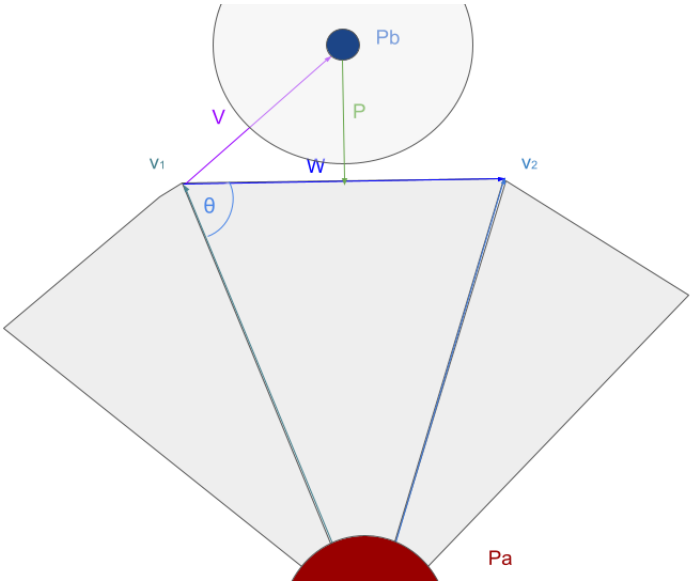

Figura 34 – A figura ilustra o problema na *Narrow-Phase*

Nesse cenário o algoritmo irá comparar o raio de P_b com o tamanho de $\vec{p}$, o que nunca irá resultar em uma colisão.

4.3 Possíveis Soluções

Como temos um entendimento completo do algoritmo e conhecimento sobre outras técnicas, possuímos algumas possibilidades de soluções. Vimos que o principal problema recai na *Narrow-Phase*. Conseguimos algumas soluções geométricas que penalizam corretamento quando o endoscópio está fora do cólon, sem um custo adicional muito alto, levando as outras estruturas nos cálculos.

4.4 Outras Aplicações

Outra aplicação do simulador desenvolvida por pesquisadores dentro do projeto foi sua integração com realidade virtual, sendo outro campo com seus inúmeros desafios

e complexidades. Obtendo dessa forma uma maior imersão e capacidade de simular outros pontos necessários em relação ao procedimento de colonoscopia. Podemos ver parte do resultado desse estudo de caso na Figura 35.

Figura 35 – A figura mostra o simulador de colonoscopia em VR, utilizando a técnica desenvolvida na simulação

5 CONCLUSÃO

Foi apresentada uma nova técnica para detecção de colisão entre malhas deformáveis, focada em simulações de colonoscopia. Com isso, foi feita a implementação de um *Framework* da técnica, expondo os métodos necessários para sua integração à aplicações que se encaixam em seu uso.

Utilizamos uma BVH com AABBs para *Broad-Phase* e um algoritmo específico na *Narrow-Phase*, o qual se aproveita do formato cilíndrico das malhas para trabalhar com a geometria dos objetos de uma forma simples e objetiva.

Sua integração com um simulador mostrou resultados em relação a sua robustez numérica e performance, demonstrando sua funcionalidade ao ponto que a simulação ocorreu como esperado.

5.1 Trabalhos Futuros

Como trabalhos futuros iremos solucionar os problemas levantados na seção de Resultados. Em cenários que o simulador físico fica sob muito estresse é gerado um cenário degenerado onde não conseguimos mais resultados plausíveis. Dessa forma, o primeiro passo na continuação da técnica é resolver esse problema. Para isso temos alguma opções, incluir mais uma opção nas checagens da *Narrow-Phase*. Levando em conta outros pontos de controle, conseguimos determinar se a malha se encontra dentro ou fora da seção transversal. Com isso, precisamos de novos testes e experimentos para analisar o comportamento na integração.

Também iremos produzir testes contra outros métodos implementados nos atuais simuladores utilizados no estados da arte. Gerar comparativos entre o método apresentado com relação a performance, estabilidade e integração. Dessa forma, planejamos montar um comparativo entre todos principais métodos para detecção de colisão entre malhas deformáveis no âmbito de simulações cirúrgicas.

REFERÊNCIAS

ANDRADE, A. Game engines: a survey. **EAI Endorsed Transactions on Serious Games**, [S.l.], v.2, n.6, 2015.

BAREQUET, G. et al. BOXTREE: A hierarchical representation for surfaces in 3D. In: COMPUTER GRAPHICS FORUM, 1996. **Anais...** [S.l.: s.n.], 1996. v.15, n.3, p.387–396.

BERGEN, G. v. d. Efficient collision detection of complex deformable models using AABB trees. **Journal of graphics tools**, [S.l.], v.2, n.4, p.1–13, 1997.

BERNDT, I.; TORCHELSEN, R.; MACIEL, A. Efficient surgical cutting with position-based dynamics. **IEEE computer graphics and applications**, [S.l.], v.37, n.3, p.24–31, 2017.

BRUCE, M.; CHOI, J. Detection of endoscopic looping during colonoscopy procedure by using embedded bending sensors. **Medical Devices: Evidence and Research**, [S.l.], p.171–191, 2018.

CATMULL, E.; ROM, R. A class of local interpolating splines. In: **Computer aided geometric design**. [S.l.]: Elsevier, 1974. p.317–326.

CHOI, A. R.; KIM, S. M.; SUNG, M. Y. Controlling the contact levels of details for fast and precise haptic collision detection. **Frontiers of Information Technology & Electronic Engineering**, [S.l.], v.18, n.8, p.1117–1130, 2017.

DIK, V. K.; MOONS, L. M.; SIERSEMA, P. D. Endoscopic innovations to increase the adenoma detection rate during colonoscopy. **World journal of gastroenterology: WJG**, [S.l.], v.20, n.9, p.2200, 2014.

ECK, M. Degree reduction of Bézier curves. **Computer Aided Geometric Design**, [S.l.], v.10, n.3-4, p.237–251, 1993.

ELOE, N. W.; STEURER, J. A.; LEOPOLD, J. L.; SABHARWAL, C. L. Dual graph partitioning for Bottom-Up BVH construction. **Journal of Visual Languages & Computing**, [S.l.], v.25, n.6, p.764–771, 2014.

ERICSON, C. **Real-time collision detection**. [S.l.]: Crc Press, 2004.

FADHEL, M. A.; OMAR, Z. B. Geometric piecewise cubic bézier interpolating polynomial with C2 continuity. , [S.l.], v.20, n.1, p.133–159, 2021.

FANG, J.; YOU, L.; CHAUDHRY, E.; ZHANG, J. State-of-the-art improvements and applications of position based dynamics. **Computer Animation and Virtual Worlds**, [S.l.], p.e2143, 2023.

FARIN, G. Class a Bézier curves. **Computer aided geometric design**, [S.l.], v.23, n.7, p.573–581, 2006.

FENG, Y. et al. Causes of death after colorectal cancer diagnosis: A population-based study. **Frontiers in Oncology**, [S.l.], v.11, p.647179, 2021.

FIGUEIREDO, M.; MARCELINO, L.; FERNANDO, T. A survey on collision detection techniques for virtual environments. In: V SYMPOSIUM IN VIRTUAL REALITY, 2002. **Proceedings...** [S.l.: s.n.], 2002. p.285–307.

FRANCE, L. et al. A layered model of a virtual human intestine for surgery simulation. **Medical image analysis**, [S.l.], v.9, n.2, p.123–132, 2005.

FROEHLICH, F. et al. Impact of colonic cleansing on quality and diagnostic yield of colonoscopy: the European Panel of Appropriateness of Gastrointestinal Endoscopy European multicenter study. **Gastrointestinal endoscopy**, [S.l.], v.61, n.3, p.378–384, 2005.

GOLDSMITH, J.; SALMON, J. Automatic creation of object hierarchies for ray tracing. **IEEE Computer Graphics and Applications**, [S.l.], v.7, n.5, p.14–20, 1987.

GOTTSCHALK, S. A. **Collision queries using oriented bounding boxes**. [S.l.]: The University of North Carolina at Chapel Hill, 2000.

HAAS, J. K. A history of the unity game engine. **Diss. Worcester Polytechnic Institute**, [S.l.], v.483, n.2014, p.484, 2014.

HABER, J.; STAMMINGER, M.; SEIDEL, H.-P. Enhanced automatic creation of multipurpose object hierarchies. In: EIGHTH PACIFIC CONFERENCE ON COMPUTER GRAPHICS AND APPLICATIONS, 2000. **Proceedings...** [S.l.: s.n.], 2000. p.52–437.

HAZEWINKEL, Y.; DEKKER, E. Colonoscopy: basic principles and novel techniques. **Nature reviews Gastroenterology & hepatology**, [S.l.], v.8, n.10, p.554–564, 2011.

HELD, M.; KLOSOWSKI, J. T.; MITCHELL, J. S. Evaluation of collision detection methods for virtual reality fly-throughs. In: CANADIAN CONFERENCE ON COMPUTATIONAL GEOMETRY, 1995. **Anais...** [S.l.: s.n.], 1995. p.205–210.

HUBBARD, P. M. Interactive collision detection. In: IEEE RESEARCH PROPERTIES IN VIRTUAL REALITY SYMPOSIUM, 1993., 1993. **Proceedings...** [S.l.: s.n.], 1993. p.24–31.

KIM, M. et al. Parallel cloth simulation with effective collision detection for interactive AR application. **Multimedia Tools and Applications**, [S.l.], v.78, n.4, p.4851–4868, 2019.

KO, C. W. et al. Serious complications within 30 days of screening and surveillance colonoscopy are uncommon. **Clinical Gastroenterology and Hepatology**, [S.l.], v.8, n.2, p.166–173, 2010.

KOCH, A. D. et al. Competence measurement during colonoscopy training: the use of self-assessment of performance measures. **Official journal of the American College of Gastroenterology| ACG**, [S.l.], v.107, n.7, p.971–975, 2012.

KOCKARA, S. et al. Collision detection: A survey. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2007., 2007. **Anais...** [S.l.: s.n.], 2007. p.4046–4051.

KORZENIOWSKI, P. et al. NOViSE: a virtual natural orifice transluminal endoscopic surgery simulator. **International journal of computer assisted radiology and surgery**, [S.l.], v.11, n.12, p.2303–2315, 2016.

KUFFEL, P.; KENT, K.; IRWIN, G. The implementation and effectiveness of linear interpolation within digital simulation. **International Journal of Electrical Power & Energy Systems**, [S.l.], v.19, n.4, p.221–227, 1997.

LARSSON, T.; AKENINE-MÖLLER, T. Collision detection for continuously deforming bodies. In: EUROGRAPHICS (SHORT PRESENTATIONS), 2001. **Anais...** [S.l.: s.n.], 2001.

LI, F.; HU, G.; ABBAS, M.; MIURA, K. T. The generalized H-Bézier model: geometric continuity conditions and applications to curve and surface modeling. **Mathematics**, [S.l.], v.8, n.6, p.924, 2020.

LIANG, T. et al. Collision detection of virtual plant based on bounding volume hierarchy: A case study on virtual wheat. **Journal of integrative agriculture**, [S.l.], v.17, n.2, p.306–314, 2018.

MIRTICH, B. V. **Impulse-based dynamic simulation of rigid body systems**. [S.l.]: University of California, Berkeley, 1996.

MOORE, M. P. **A flexible object animation system**. [S.l.]: University of California, Santa Cruz, 1988.

NAYLOR, B.; AMANATIDES, J.; THIBAULT, W. Merging BSP trees yields polyhedral set operations. **ACM Siggraph Computer Graphics**, [S.l.], v.24, n.4, p.115–124, 1990.

NITZBERG, B.; LO, V. Distributed shared memory: A survey of issues and algorithms. **Computer**, [S.l.], v.24, n.8, p.52–60, 1991.

REX, D. K. et al. Quality indicators for colonoscopy. **Gastrointestinal endoscopy**, [S.l.], v.81, n.1, p.31–53, 2015.

ROSA, G. Real-time knot evaluation for suturing simulation. **Federal University of Pelotas**, [S.l.], 2019.

SHAH, S. G. et al. Effect of magnetic endoscope imaging on colonoscopy performance: a randomised controlled trial. **The Lancet**, [S.l.], v.356, n.9243, p.1718–1722, 2000.

SIEGEL, R. L.; MILLER, K. D.; JEMAL, A. Cancer statistics, 2019. **CA: a cancer journal for clinicians**, [S.l.], v.69, n.1, p.7–34, 2019.

SINGH, H. et al. Propofol for sedation during colonoscopy. **Cochrane database of systematic reviews**, [S.l.], n.4, 2008.

STARIBRATOV, I.; MANOLOVA, N. Application of Mathematical Models in Graphic Design. , [S.l.], v.65, p.72–81, 03 2022.

TANG, M. et al. VolCCD: Fast continuous collision culling between deforming volume meshes. **ACM Transactions on Graphics (TOG)**, [S.l.], v.30, n.5, p.1–15, 2011.

TESCHNER, M. et al. Collision detection for deformable objects. In: COMPUTER GRAPHICS FORUM, 2005. **Anais...** [S.l.: s.n.], 2005. v.24, n.1, p.61–81.

VAN RIJN, J. C. et al. Polyp miss rate determined by tandem colonoscopy: a systematic review. **Official journal of the American College of Gastroenterology| ACG**, [S.l.], v.101, n.2, p.343–350, 2006.

WANG, M.; CAO, J. A review of collision detection for deformable objects. **Computer Animation and Virtual Worlds**, [S.l.], v.32, n.5, p.e1987, 2021.

WIEL, S. Van der et al. Simulator training in gastrointestinal endoscopy–From basic training to advanced endoscopic procedures. **Best Practice & Research Clinical Gastroenterology**, [S.l.], v.30, n.3, p.375–387, 2016.

WINAWER, S. J. et al. Prevention of colorectal cancer by colonoscopic polypectomy. **New England Journal of Medicine**, [S.l.], v.329, n.27, p.1977–1981, 1993.

YI, S. et al. New colonoscopy simulator with improved haptic fidelity. **Advanced Robotics**, [S.l.], v.20, n.3, p.349–365, 2006.

YUKSEL, C.; SCHAEFER, S.; KEYSER, J. Parameterization and applications of Catmull-Rom curves. **Comput. Aided Des.**, [S.l.], v.43, p.747–755, 2011.

YUKSEL, C.; SCHAEFER, S.; KEYSER, J. Parameterization and applications of Catmull–Rom curves. **Computer-Aided Design**, [S.l.], v.43, n.7, p.747–755, 2011.

ZHANG, J.; ZHONG, Y.; GU, C. Deformable models for surgical simulation: a survey. **IEEE reviews in biomedical engineering**, [S.l.], v.11, p.143–164, 2017.