4
Resultados

O método proposto foi implementado em *C++* usando *OpenGL* e a linguagem de *Shader GLSL*. A partir da implementação corrente foram realizados diversos testes visando quantificar o desempenho e analisar a qualidade dos resultados obtidos.

Os testes foram efetuados em um processador *Intel i7* de *2.8 GHz* com *12Gb* de memória *RAM* utilizando uma *Nvidia GeForce GTX 480*. Todas as medições de tempo das próximas seções são tempos médios da execução dos algoritmos.

As seções a seguir apresentam os resultados alcançados. Uma análise da qualidade dos resultados para diversos modelos é feita na Seção 4.1. A Seção 4.2 compara o efeito produzido pela oclusão ambiente juntamente com a utilização de iluminação difusa. O desempenho de cada etapa do algoritmo é analisado na Seção 4.3. A utilização de *jitter* é discutida brevemente na Seção 4.4. A Seção 4.5 compara a técnica proposta com outros algoritmos de oclusão ambiente implementados durante o desenvolvimento desta dissertação. Uma comparação com uma aplicação de oclusão ambiente em espaço de tela desenvolvida pela *Nvidia* é feita na Seção 4.6. Finalmente, o comportamento do algoritmo com a variação da resolução da tela e da complexidade da geometria é discutido na Seção 4.7.

4.1
Qualidade

Visando verificar a qualidade dos resultados obtidos usando a técnica proposta de traçado de cones em volumes voxelizados foram executados testes usando diferentes configurações de qualidade. Foram definidas três configurações: Qualidade Baixa, Qualidade Média e Qualidade Alta, as quais podem ser alternadas de acordo com a capacidade do *hardware* sendo utilizado. A Tabela 4.1 exibe os parâmetros utilizados em cada configuração de qualidade.

Tabela 4.1: Parâmetros utilizados para cada configuração de qualidade.

Qualidade	Número de Cones	Número de Esferas por Cone	Número de Amostras por Esfera
Baixa	6	3	3
Média	6	5	6
Alta	6	7	12

Como critério de comparação de qualidade utiliza-se o aplicativo de traçado de raios Optix (12), desenvolvido pela *Nvidia*. Este *software* é executado na GPU utilizando *CUDA* e produz resultados de grande qualidade mas não em tempo real. Nenhum parâmetro ou informação sobre detalhes da execução do algoritmo (tal como número de raios, utilização de *multi-sampling*, raio do hemisfério, etc.) é disponibilizado. Contudo, pode-se obter a posição corrente da câmera, o que facilita a reprodução de poses de cada cena para comparação. A resolução padrão das imagens utilizadas nos testes foi escolhida para facilitar a comparação com esta referência, a qual utiliza a resolução 640x480.

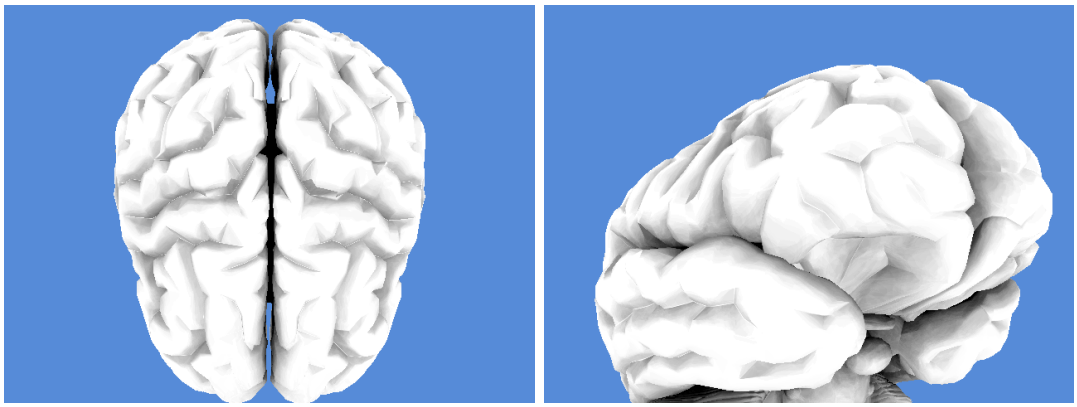
As subseções a seguir comparam a execução do algoritmo proposto utilizando as diferentes configurações de qualidades. São apresentadas uma tabela contendo o tempo de execução do algoritmo para cada qualidade e imagens para uma comparação visual. Ao final, é feita uma analise sobre os resultados obtidos pelas as diferentes configurações de qualidade.

4.1.1 Modelo brain

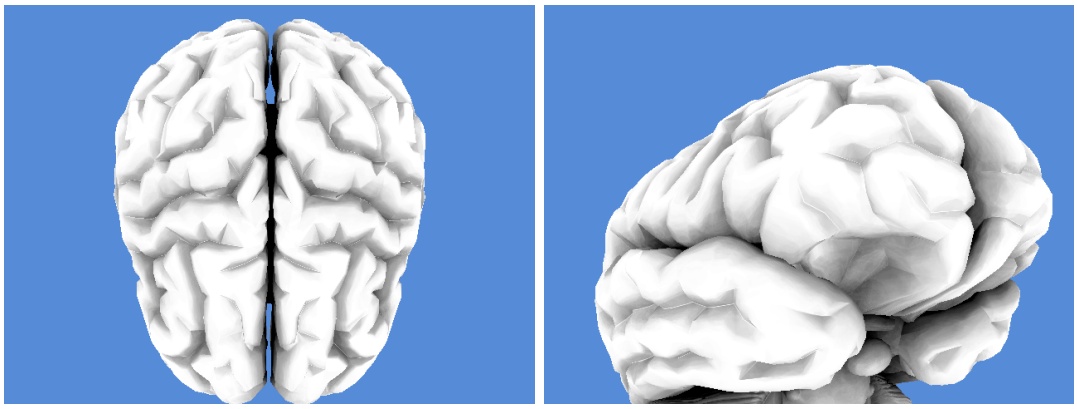
A cena utilizada neste teste é composta pelo modelo *brain* e possui 110 mil vértices e 36 mil triângulos. A Tabela 4.2 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.1 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

Tabela 4.2: Comparação entre execuções de diferentes qualidades do modelo *brain*.

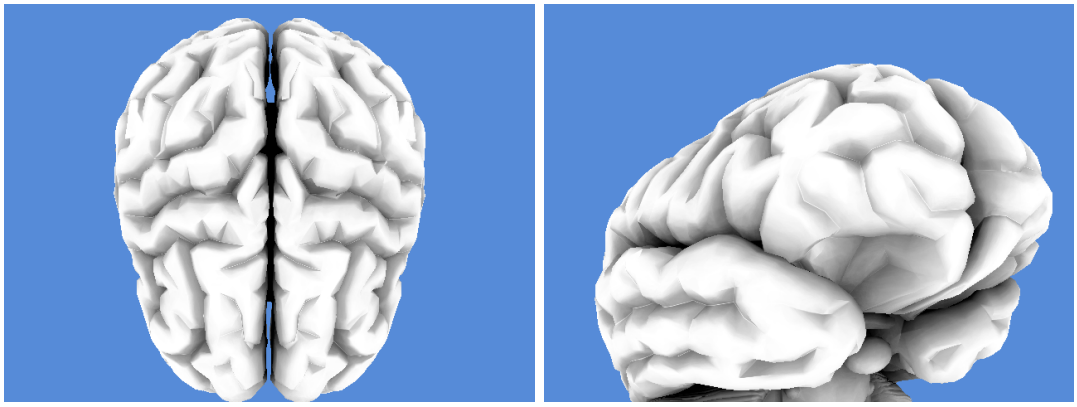
Qualidade	Pose 1	Pose 2
Baixa	5.75 ms 173.91 fps	6.38 ms 156.74 fps
Média	17.06 ms 58.62 fps	18.51 ms 54.02 fps
Alta	45.6 ms 21.93 fps	49.18 ms 20.33 fps



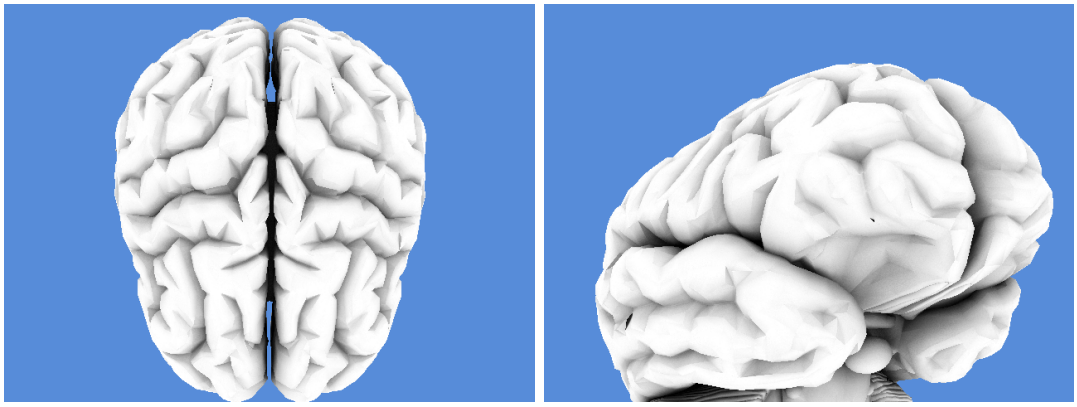
Baixa



Média



Alta



Optix

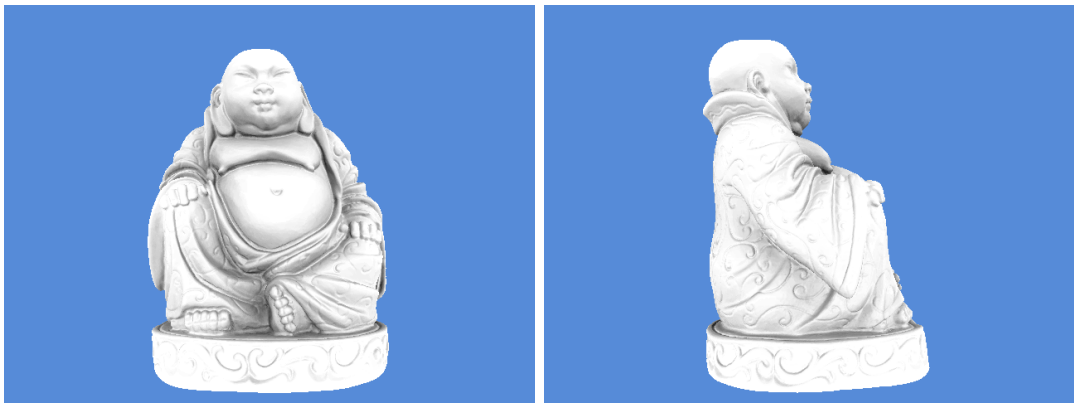
Figura 4.1: Comparação entre diferentes configurações de qualidade do modelo *brain*.

4.1.2 Modelo buddha15M

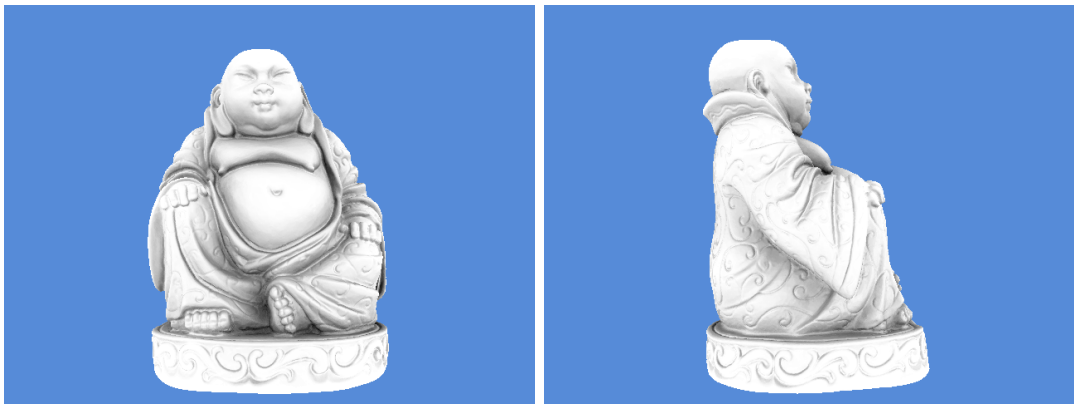
A cena utilizada neste teste é composta pelo modelo *buddha15M* e possui 757 mil vértices e 1.514 mil triângulos. A Tabela 4.3 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.2 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

Tabela 4.3: Comparação entre execuções de diferentes qualidades do modelo *buddha15M*.

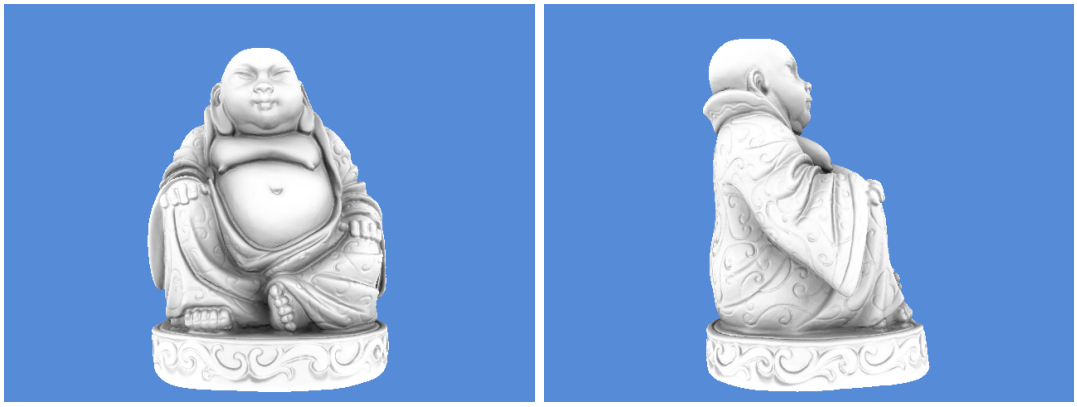
Qualidade	Pose 1	Pose 2
Baixa	7.98 ms 125.31 fps	7.19 ms 139.08 fps
Média	17.10 ms 58.48 fps	14.64 ms 68.31 fps
Alta	40.39 ms 24.76 fps	33.57 ms 29.79 fps



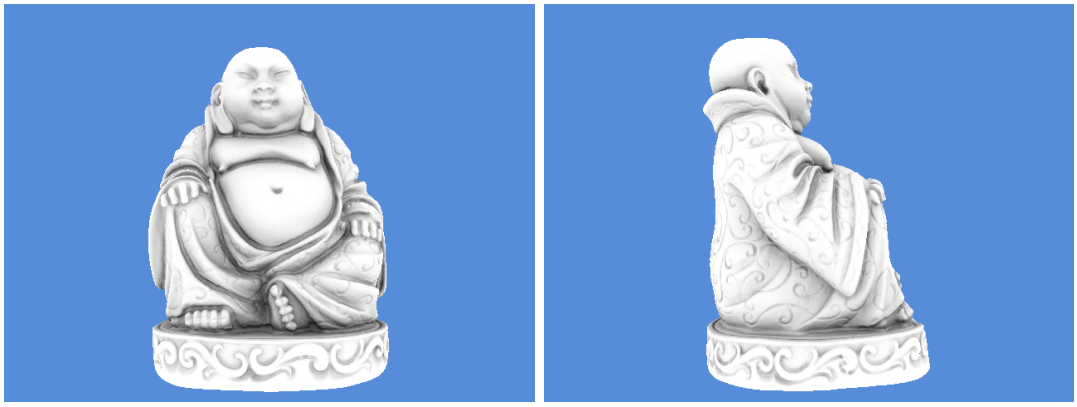
Baixa



Média



Alta



Optix

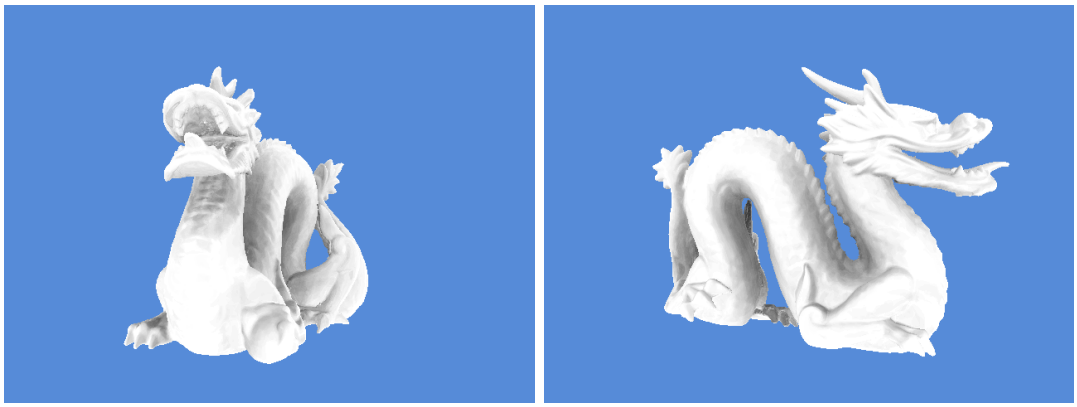
Figura 4.2: Comparação entre diferentes configurações de qualidade do modelo *buddha15M*.

4.1.3 Modelo dragon

A cena utilizada neste teste é composta pelo modelo *dragon* e possui 300 mil vértices e 100 mil triângulos. A Tabela 4.4 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.3 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

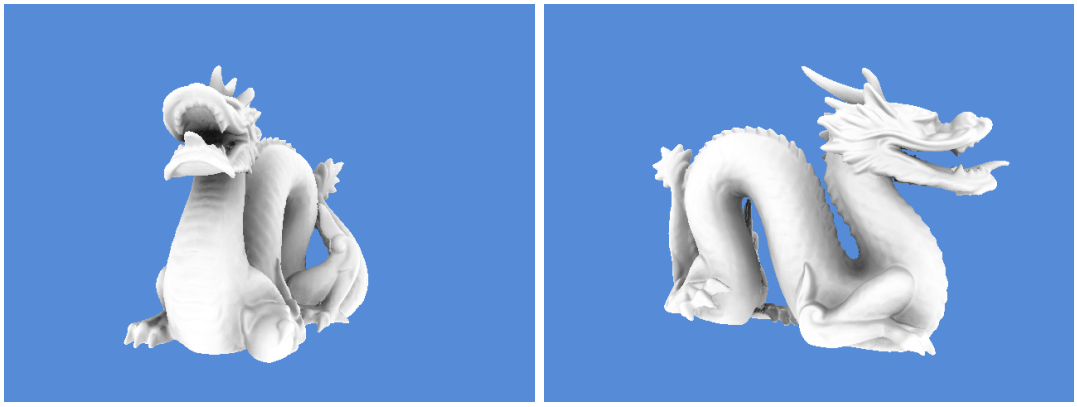
Tabela 4.4: Comparação entre execuções de diferentes qualidades do modelo *dragon*.

Qualidade	Pose 1	Pose 2
Baixa	3.47 ms 288.18 fps	4.42 ms 226.24 fps
Média	9.79 ms 102.15 fps	12.40 ms 80.65 fps
Alta	25.42 ms 39.34 fps	32.44 ms 30.83 fps

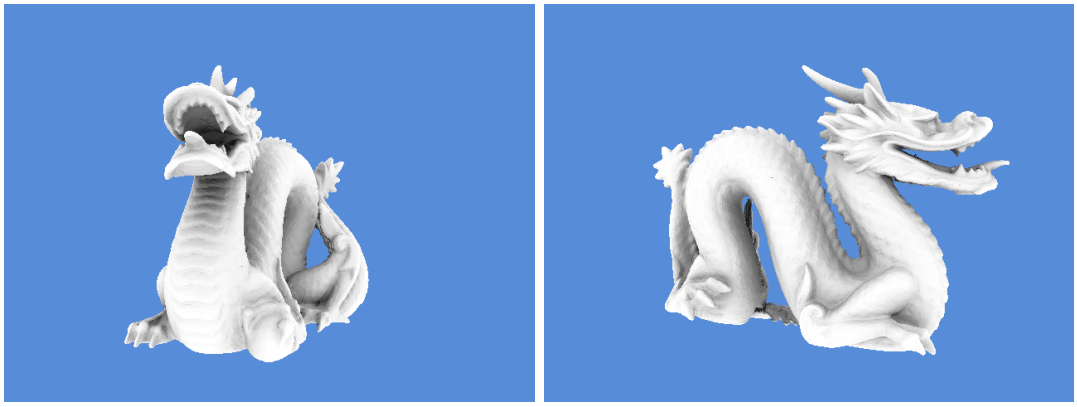


Baixa

Média



Alta



Optix

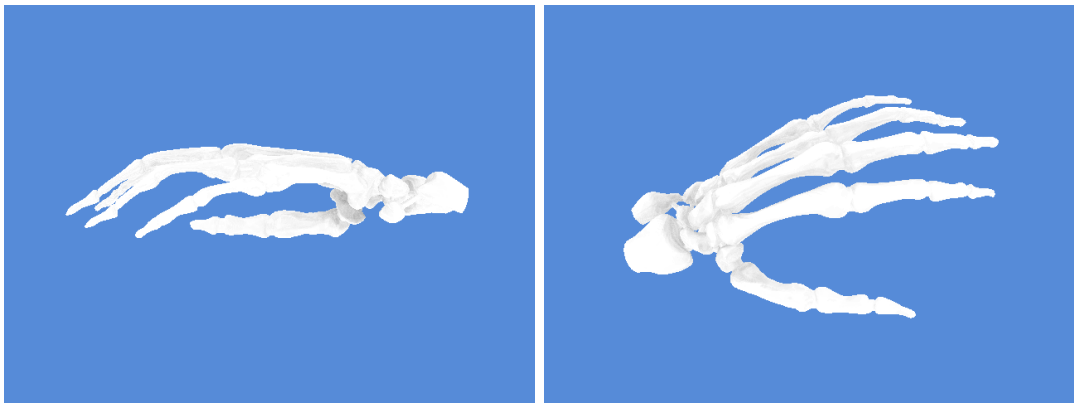
Figura 4.3: Comparação entre diferentes configurações de qualidade do modelo *dragon*.

4.1.4 Modelo hand

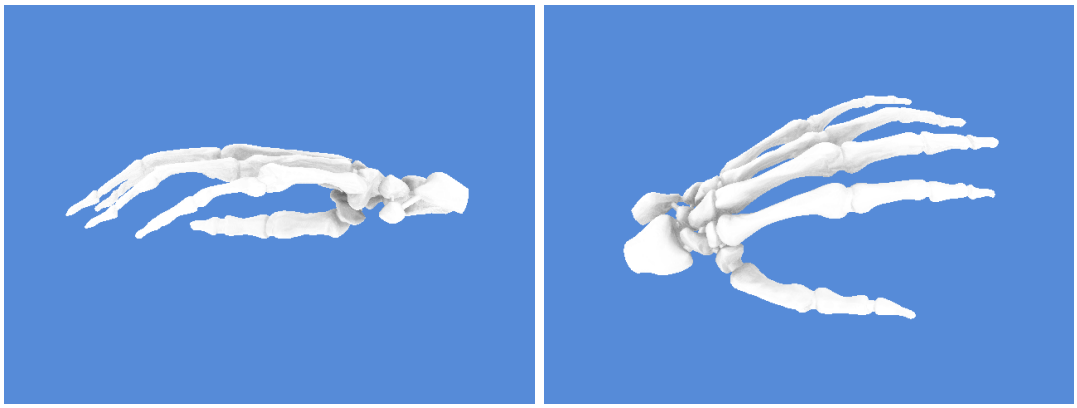
A cena utilizada neste teste é composta pelo modelo *hand* e possui 327 mil vértices e 654 mil triângulos. A Tabela 4.5 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.4 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

Tabela 4.5: Comparação entre execuções de diferentes qualidades do modelo *hand*.

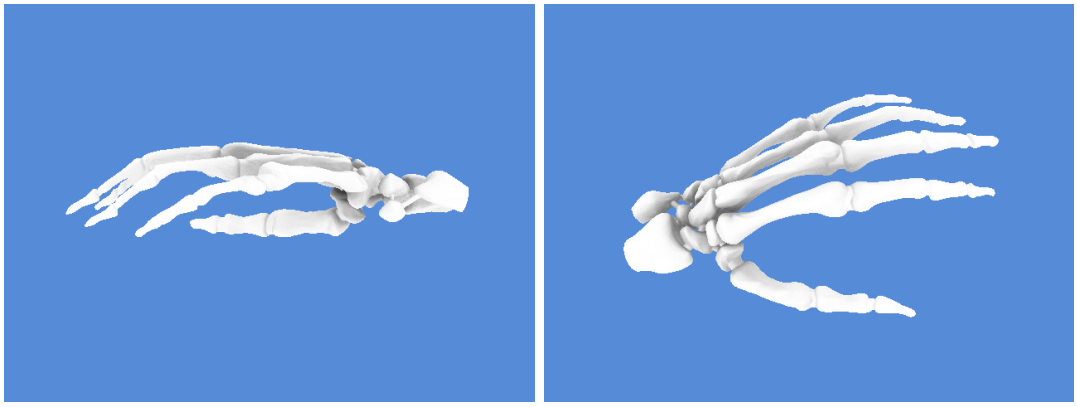
Qualidade	Pose 1	Pose 2
Baixa	3.03 ms 330.03 fps	4.02 ms 248.76 fps
Média	5.94 ms 168.35 fps	9.16 ms 109.17 fps
Alta	13.30 ms 75.19 fps	21.94 ms 45.58 fps



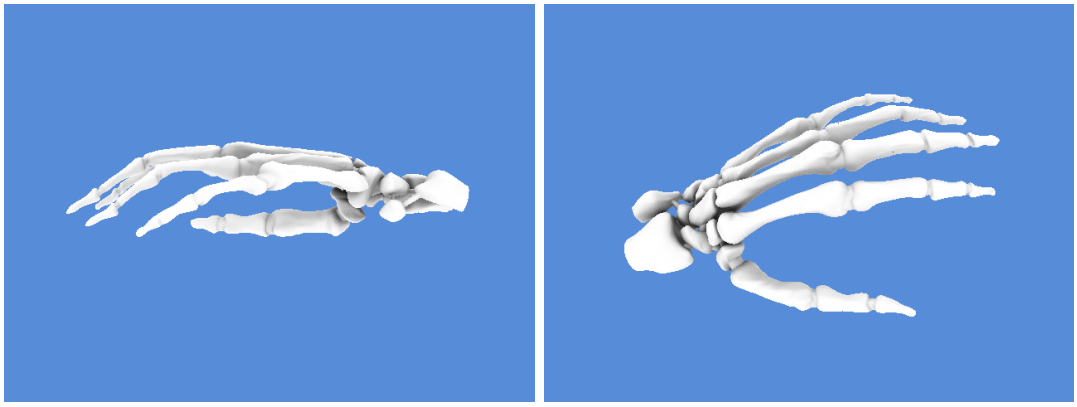
Baixa



Média



Alta



Optix

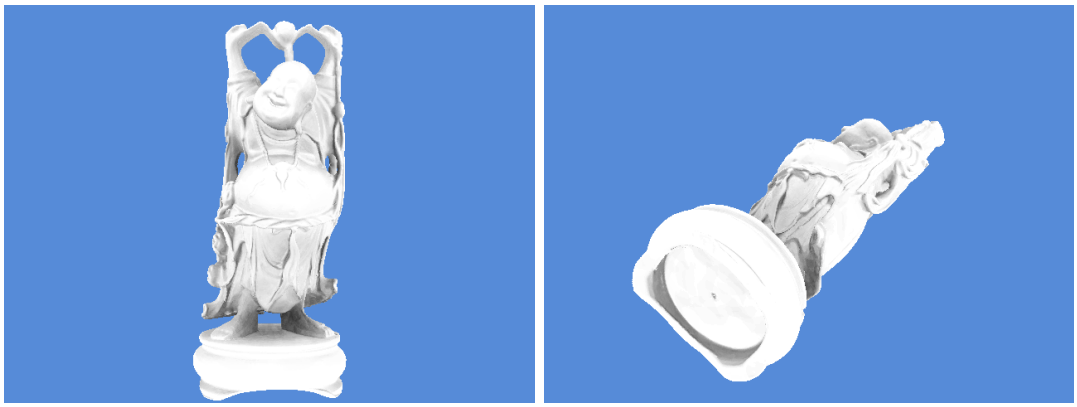
Figura 4.4: Comparação entre diferentes configurações de qualidade do modelo *hand*.

4.1.5 Modelo happyvrip

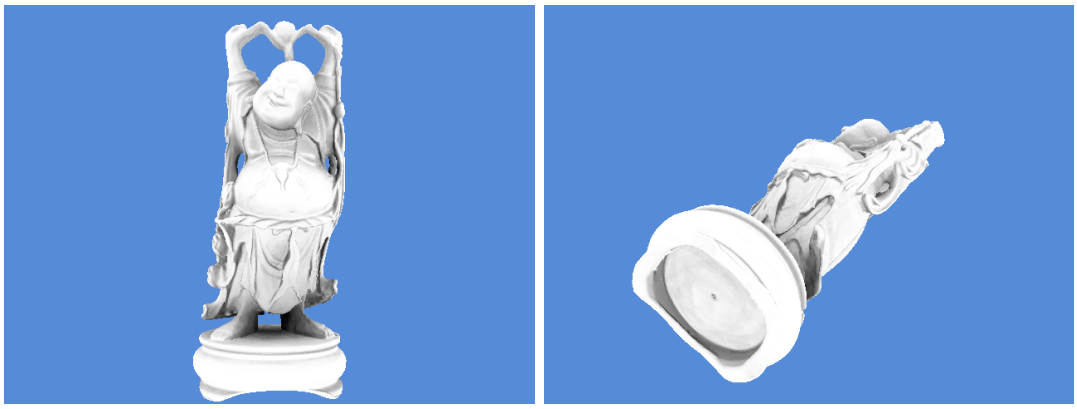
A cena utilizada neste teste é composta pelo modelo *happyvrip* e possui 543 mil vértices e 1.087 mil triângulos. A Tabela 4.6 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.5 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

Tabela 4.6: Comparação entre execuções de diferentes qualidades do modelo *happyvrip*.

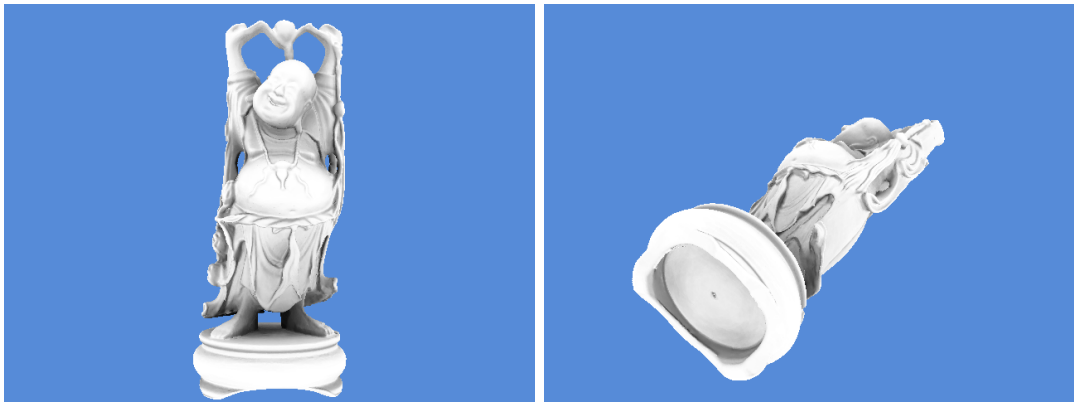
Qualidade	Pose 1	Pose 2
Baixa	6.59 ms 151.75 fps	5.63 ms 177.62 fps
Média	14.23 ms 70.27 fps	11.59 ms 86.28 fps
Alta	33.30 ms 30.03 fps	26.68 ms 37.48 fps



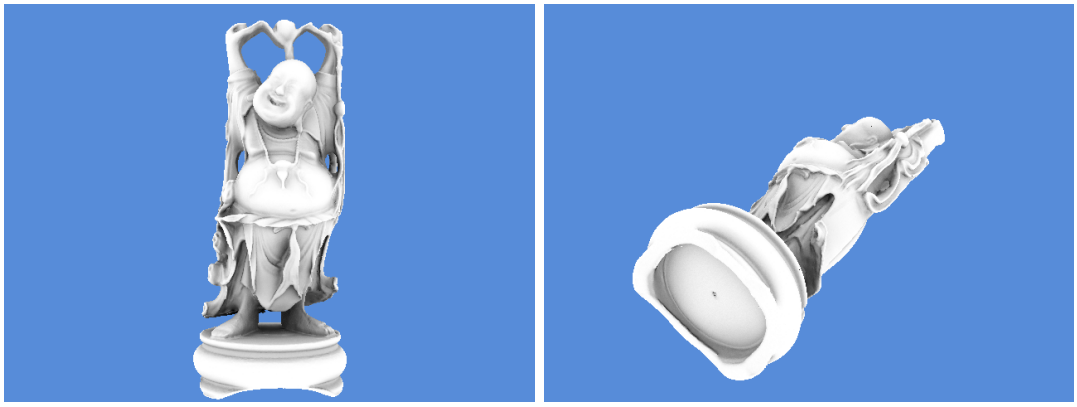
Baixa



Média



Alta



Optix

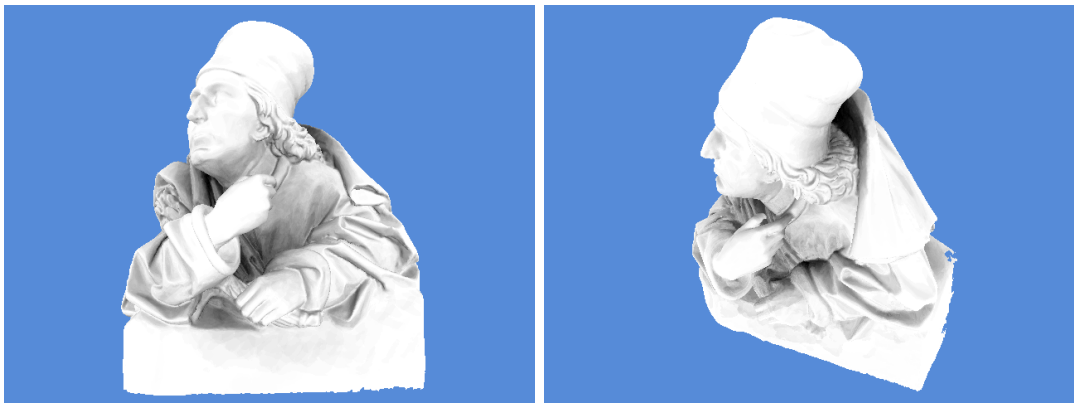
Figura 4.5: Comparação entre diferentes configurações de qualidade do modelo *happyvrip*.

4.1.6 Modelo jsyrlin

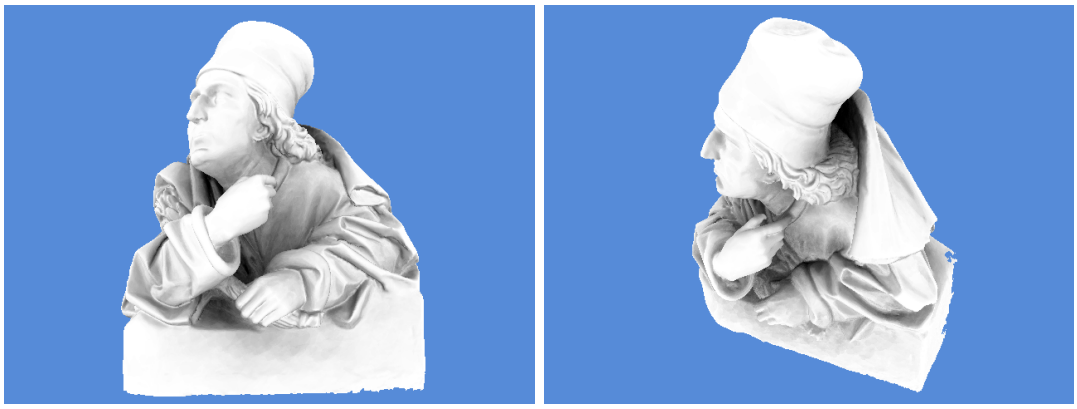
A cena utilizada neste teste é composta pelo modelo *jsyrlin* e possui 253 mil vértices e 505 mil triângulos. A Tabela 4.7 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.6 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

Tabela 4.7: Comparação entre execuções de diferentes qualidades do modelo *jsyrlin*.

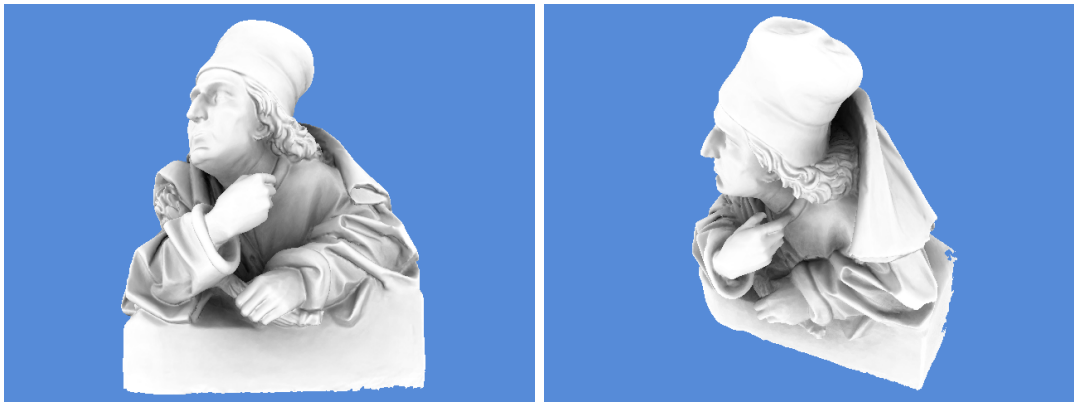
Qualidade	Pose 1	Pose 2
Baixa	6.31 ms 158.48 fps	5.59 ms 178.89 fps
Média	16.56 ms 60.39 fps	14.52 ms 68.87 fps
Alta	42.32 ms 23.63 fps	36.44 ms 27.44 fps



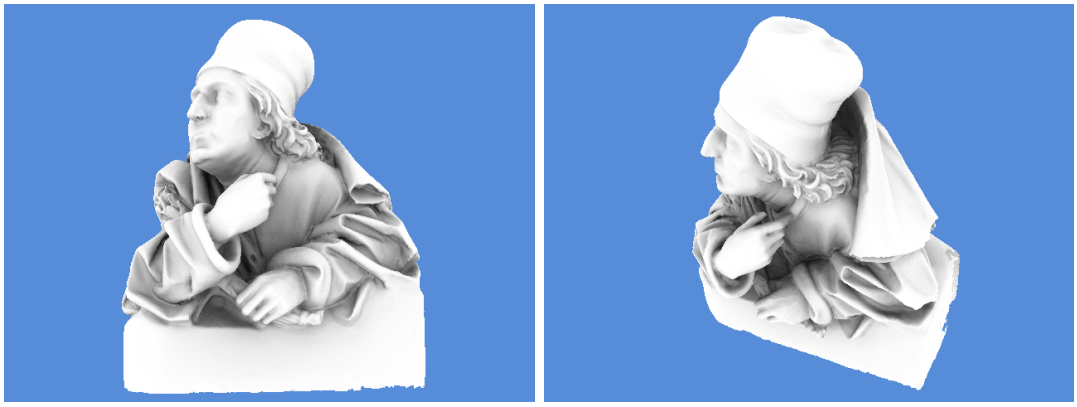
Baixa



Média



Alta



Optix

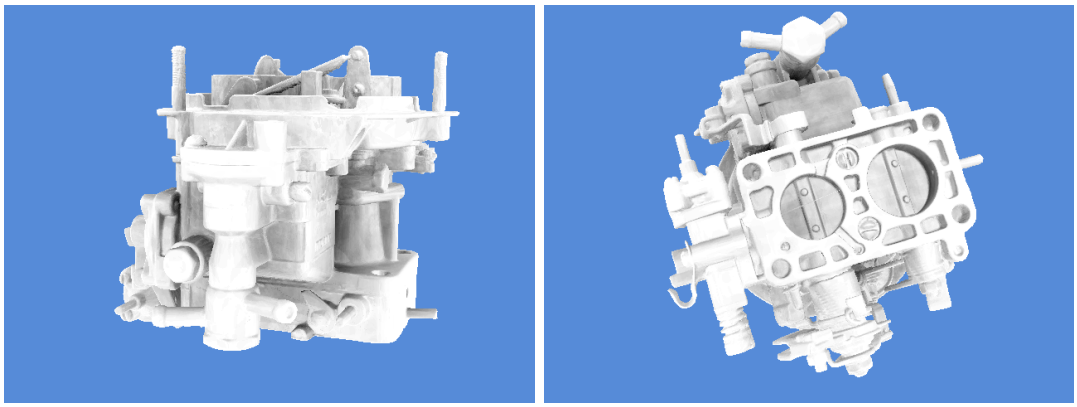
Figura 4.6: Comparação entre diferentes configurações de qualidade do modelo *jsyrlin*.

4.1.7 Modelo karburator-500k

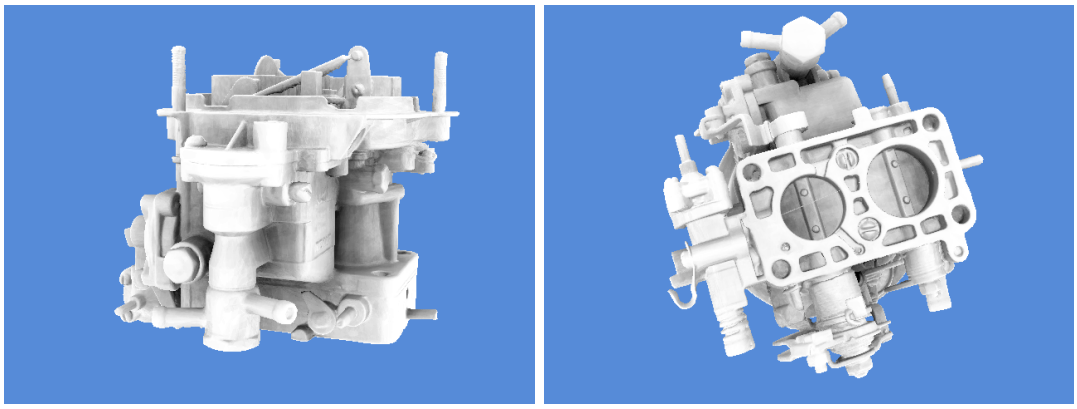
A cena utilizada neste teste é composta pelo modelo *karburator-500k* e possui 251 mil vértices e 500 mil triângulos. A Tabela 4.8 contém os resultados de tempo para a execução de cada uma das configurações de qualidade. As imagens na Figura 4.7 visam apresentar uma comparação entre a qualidade de cada uma das configurações.

Tabela 4.8: Comparação entre execuções de diferentes qualidades do modelo *karburator-500k*.

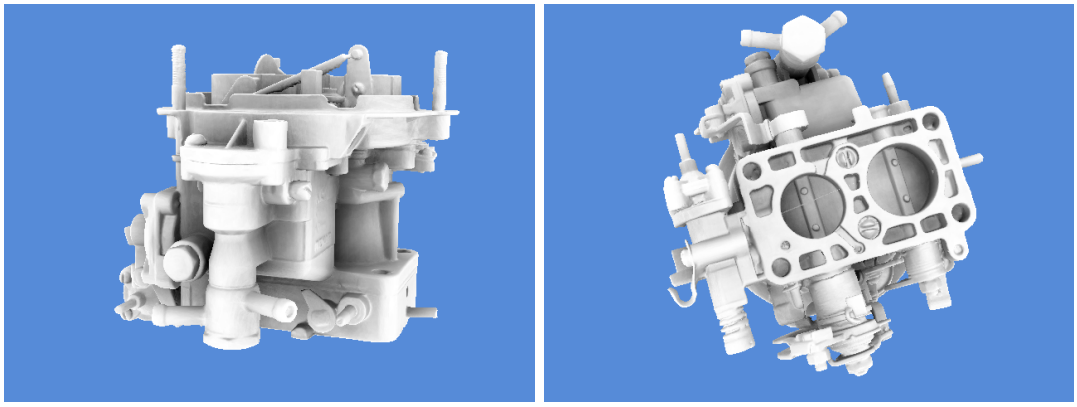
Qualidade	Pose 1	Pose 2
Baixa	6.22 ms 160.77 fps	6.79 ms 147.28 fps
Média	16.13 ms 62.00 fps	17.93 ms 55.77 fps
Alta	41.00 ms 24.39 fps	45.92 ms 21.78 fps



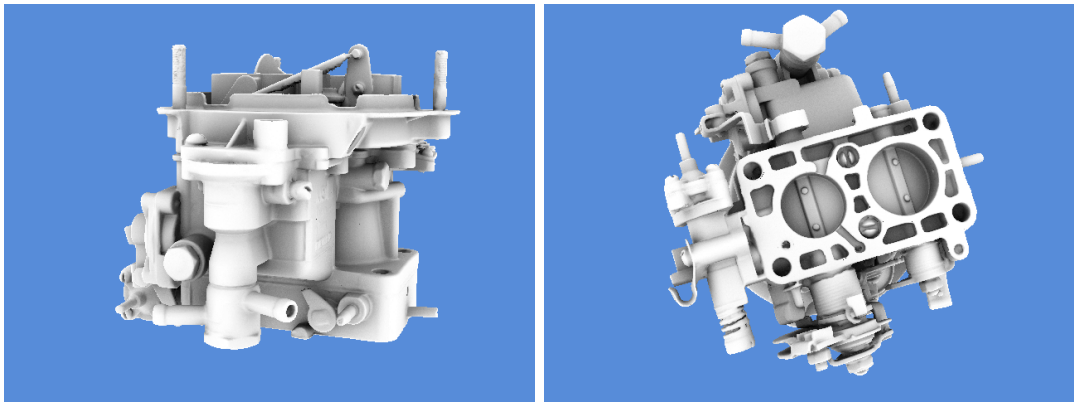
Baixa



Média



Alta



Optix

Figura 4.7: Comparação entre diferentes configurações de qualidade do modelo *karburator-500k*.

4.1.8 Análise

Os resultados apresentados nas figuras anteriores permitem perceber que as imagens produzidas pelo algoritmo proposto tendem a se aproximar cada vez mais do Optix à medida que a qualidade aumenta. Algumas cenas chegam a produzir resultados muito próximos mesmo na qualidade baixa.

Ao avaliar as tabelas, nota-se que a medida que a qualidade aumenta o desempenho decai, contudo, com a evolução do *hardware* a configuração de alta qualidade tende a substituir a configuração de baixa qualidade mantendo um bom nível de interatividade e qualidade.

Uma restrição do método proposto se encontra na resolução da dimensão z da grade. No estado atual ela se encontra limitada a 128 *voxels* devido ao número máximo de *bits* de um *texel*. Esta restrição faz com que detalhes em interiores, principalmente orifícios pequenos, sejam perdidos. Pode-se perceber este efeito na Figura 4.7; na imagem à esquerda, um orifício na parte inferior da peça se encontra bastante ocluso na imagem de referência e apenas levemente ocluso nas imagens produzidas pelo algoritmo. A resolução da grade pode ser estendida ao utilizar mais texturas para construí-la. Entretanto, isto impacta no desempenho geral da aplicação devido à maior necessidade de tempo para construir a grade e mais acessos à textura na etapa do cálculo da oclusão, estes detalhes serão investigados em pesquisas futuras.

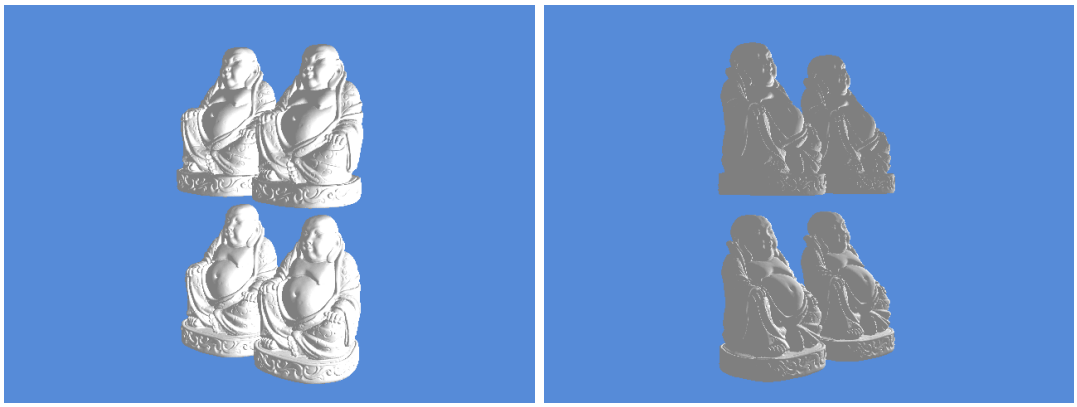
A Tabela 4.1 exibe as variáveis que podem ser alteradas no algoritmo para modificar a qualidade. O número de cones determina as direções que serão amostradas, o valor foi fixado em 6 pois este se mostrou satisfatório na amostragem do hemisfério. O número de esferas por cone influencia a qualidade significativamente, entretanto a utilização de muitas esferas força a criação de esferas de raios muito pequenos próximos ao vértice do cone, contribuindo pouco para a qualidade final. O número de amostras por esferas é o atributo que pode ser acrescido livremente e faz com que o resultado aproxime-se do resultado de maior qualidade, muito próximo ao obtido pelo Optix. Entretanto, o número de amostras afeta o desempenho sensivelmente e valores muito altos eliminam a interatividade do algoritmo. Ainda assim, a convergência do resultado indica que a aproximação proposta nesta dissertação possui a capacidade de gerar resultados próximos do cálculo fisicamente correto.

A partir desta análise optou-se pela utilização da configuração de qualidade média nos testes subsequentes devido a sua boa relação entre qualidade e desempenho.

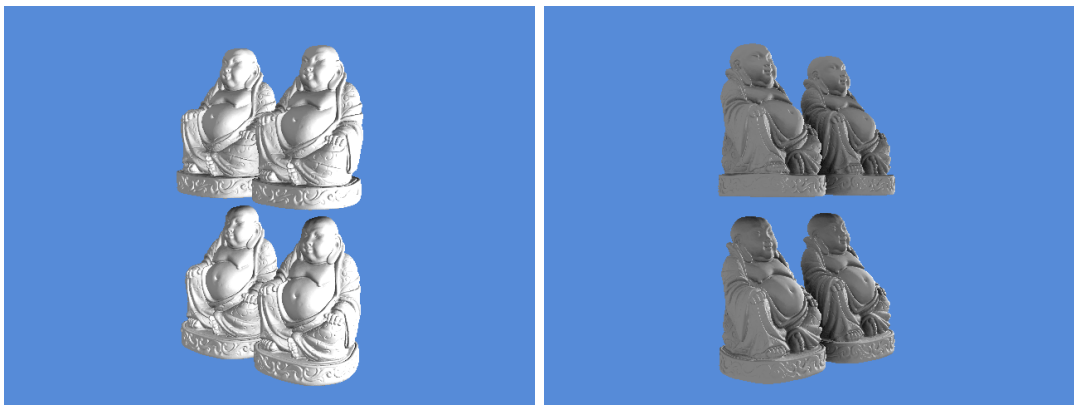
4.2 Iluminação difusa

A oclusão ambiente objetiva aumentar a qualidade da iluminação em ambientes tridimensionais. Para ilustrar este ganho de qualidade as imagens exibidas na Figura 4.8 comparam uma cena iluminada com uma iluminação difusa por *pixel* usando iluminação ambiente constante com uma iluminação difusa por *pixel* usando iluminação ambiente calculada pelo método proposto.

A cena utilizada neste teste é composta pelo modelo *buddha15M*, repetido 4 vezes, e possui 3 milhões de vértices e 6 milhões de triângulos. As imagens na Figura 4.8 visam apresentar uma comparação entre a iluminação difusa e oclusão ambiente.



Iluminação difusa sem oclusão ambiente



Iluminação difusa com oclusão ambiente

Figura 4.8: Comparação entre utilização de oclusão ambiente e iluminação difusa.

Pode-se notar o aumento na percepção da forma e dos detalhes da cena, principalmente nas regiões não atingidas diretamente pela iluminação difusa. Além disso, a posição dos objetos se torna mais evidente devido ao escurecimento das regiões próximas.

4.3 Tempo por etapa

O algoritmo proposto é composto por três etapas distintas: construção dos *buffers* de geometria, voxelização da cena e traçado de cones. A Tabela 4.9 apresenta o tempo que cada etapa necessita para a sua execução individual para uma cena composta por 3 milhões de vértices e 6 milhões de triângulos na qualidade média. A coluna da direita da tabela apresenta valores percentuais relacionando o quanto cada etapa consome do tempo total.

Tabela 4.9: Resultados de tempo para a execução de cada uma das etapas.

Etapa	Tempo (ms)	Percentual
Obtenção dos *pixels* visíveis	5.12	17.7 %
Voxelização	9.28	32.0 %
Traçado de Cones	14.61	50.4 %
Total	29.01	–

A partir da tabela pode-se notar que a etapa de traçado de cones consome pouco mais que a metade do tempo de execução de um quadro. Como as etapas anteriores dependem da geometria nota-se que o algoritmo exibe um esforço maior por *pixel* mas ainda apresenta uma dependência da geometria.

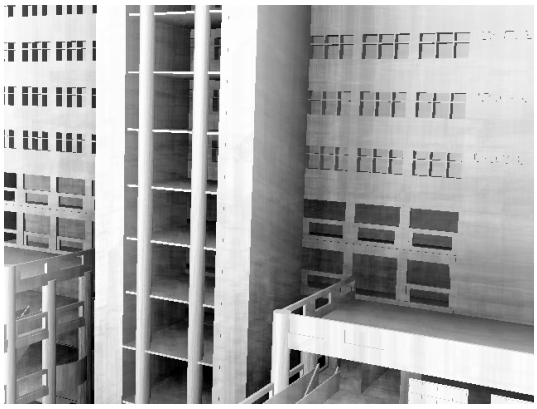
4.4 Influência do jitter

A técnica de *jitter* é utilizada para remover a formação de padrões nas imagens geradas pelo algoritmo proposto. Estes efeitos visuais ocorrem devido a utilização de amostras com distribuições iguais em todos os *pixels*.

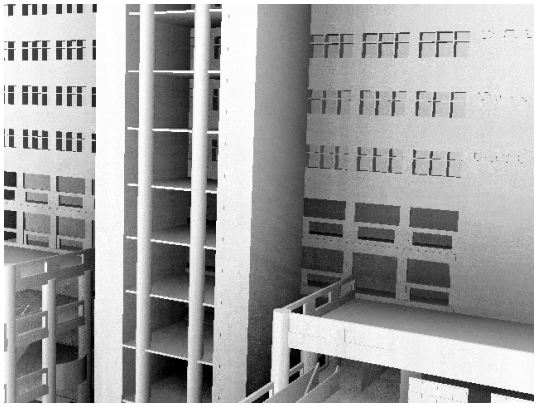
A Figura 4.9 apresenta a mesma cena sem a utilização de *jitter* (imagem superior) e com *jitter* (imagem inferior). Esta cena é composta pelo modelo *mba1* e possui 251 mil vértices e 84 mil triângulos. A Tabela 4.10 apresenta informações de tempo de execução com e sem *jitter*.

Tabela 4.10: Resultados de tempo para a execução com e sem *jitter*.

Parâmetros	Tempo (ms)	FPS
Jitter desativado	36.81	27.17
Jitter ativado	43.52	22.98



Jitter desativado



Jitter ativado

Figura 4.9: Comparação entre execução com e sem *jitter*.

Nota-se a presença de manchas horizontais nas paredes do modelo da imagem superior. Ao utilizar *jitter* estes padrões somem e são substituídos por um leve pontilhado, esse pontilhado é comum em técnicas que utilizam diversas amostragens e pode ser atenuado por técnicas de pós-processamento.

O resultado visual obtido com a utilização de *jitter* possui uma qualidade superior, contudo, ao observar-se os tempos de execução percebe-se que existe uma queda no desempenho com a utilização desta técnica. Acredita-se que essa variação ocorre devido a menor coerência no acesso a textura entre *pixels* vizinhos.

4.5 Comparação entre algoritmos

Os resultados a seguir apresentam imagens e tempo de dois algoritmos que propõem a resolução da oclusão ambiente. Ambos foram implementados juntamente com o trabalho de pesquisa desta dissertação.

O primeiro algoritmo corresponde a implementação proposta por Shanmugam e Arikan (15), e é executado totalmente em espaço de tela. Essa

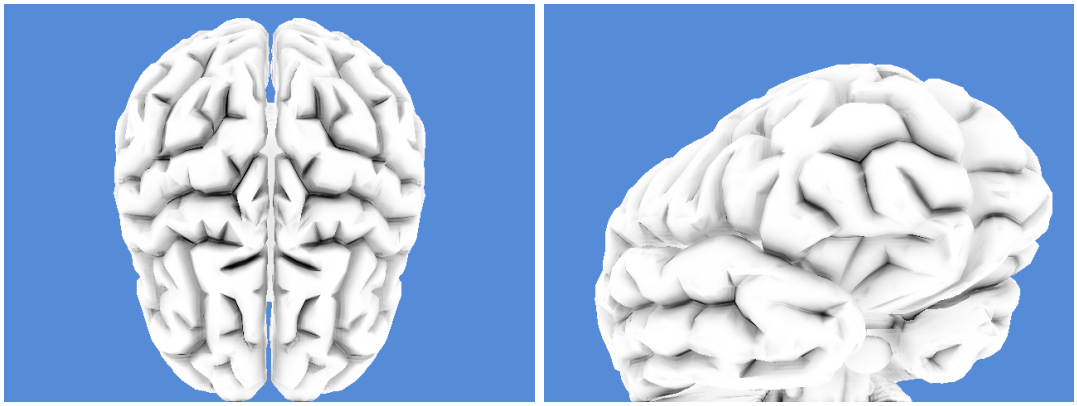
implementação é parcial e leva em conta somente a oclusão de *pixels*, desconsiderando objetos distantes. Além disso, somente é utilizado um nível de *depth-peeling*. O segundo método corresponde ao trabalho apresentado por Papaioannou et al. (13). Ele realiza a voxelização da cena e utiliza *ray-marching* no espaço voxelizado. Além dos algoritmos a mesma cena foi executada no Optix, e é utilizada como referência de qualidade.

A Figura 4.10 apresenta poses diferentes de uma mesma cena para cada um dos métodos. A Tabela 4.11 exibe os tempos para cada algoritmo em cada uma das poses de câmera.

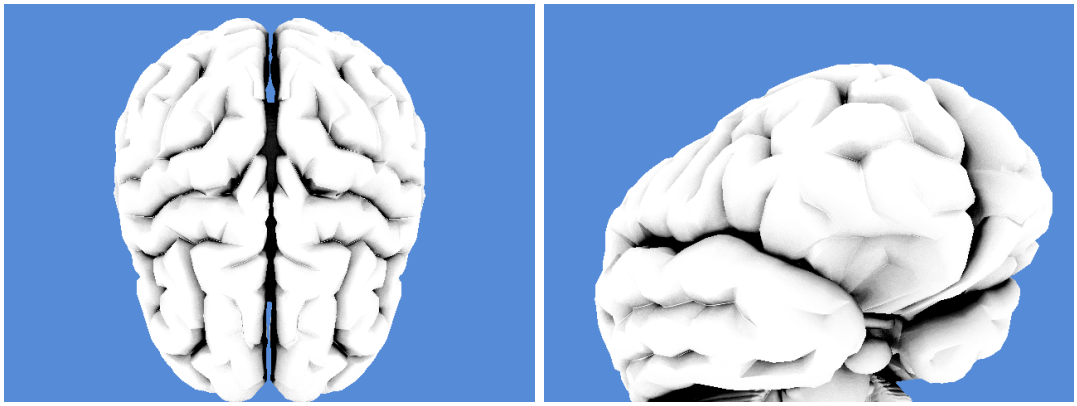
A cena utilizada neste teste é composta pelo modelo *brain* e possui 110 mil vértices e 36 mil triângulos.

Tabela 4.11: Resultados de tempo para cada aloritmo.

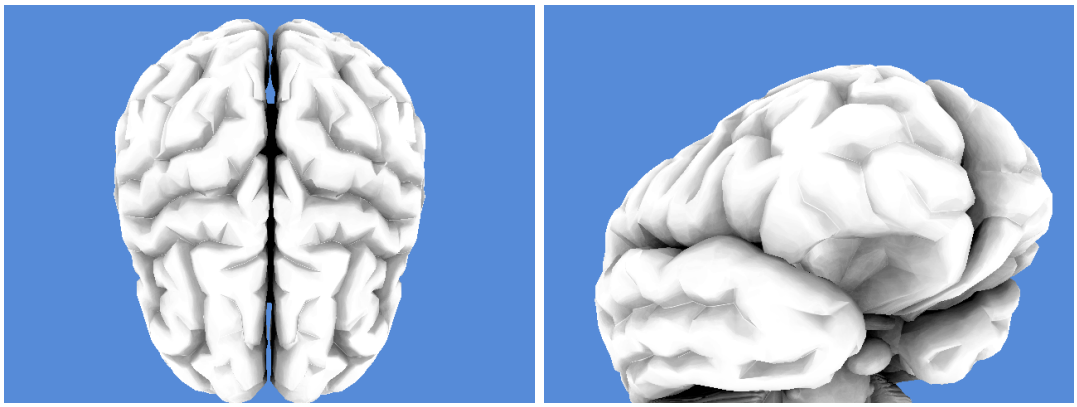
Algoritmo	Pose 1	Pose 2
Shanmugam e Arikan	7.86 ms 127.23 fps	8.98 ms 111.36 fps
Papaioannou et al.	26.52 ms 37.71 fps	30.97 ms 32.29 fps
Método proposto	17.06 ms 58.62 fps	18.51 ms 54.02 fps



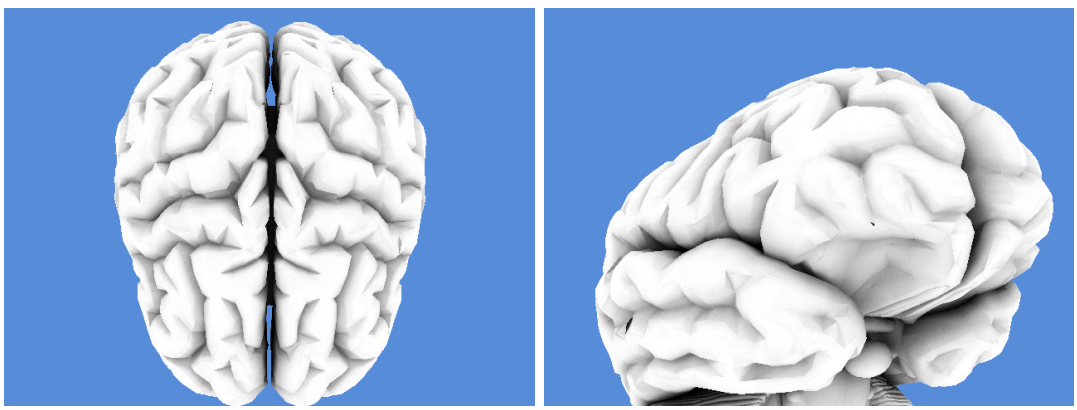
Shanmugam e Arikan



Papaioannou et al.



Método proposto



Optix

Figura 4.10: Comparação entre os algoritmos.

A partir dos resultados obtidos pode-se perceber que o algoritmo de Shanmugam e Arikan (15) apresenta o melhor desempenho. Isto se deve à sua execução ser totalmente em espaço de tela. Contudo, a oclusão no vão central presente no modelo não é estimada corretamente, consequência da ausência de informações da geometria, pois os pontos que causam a oclusão não estão visíveis a partir do ponto de vista do observador.

Os resultados do algoritmo de Papaioannou et al. (13) são mais corretos que a abordagem em espaço de tela devido a utilização da voxelização, entretanto o desempenho é inferior.

O método proposto nesta dissertação apresenta bom desempenho, e obtém qualidade comparável ao traçado de raios.

4.6 Comparação com aplicativo de oclusão em espaço de tela

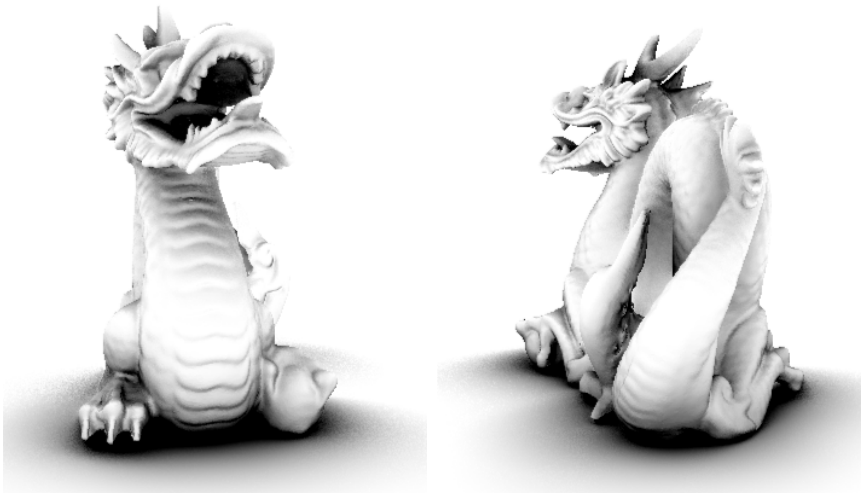
Para melhor comparar o funcionamento do método proposto foi utilizada uma aplicação desenvolvida pela *Nvidia* que realiza o cálculo da oclusão ambiente em espaço de tela. A aplicação faz parte dos exemplos do *SDK* do *Direct3D* disponíveis pela *Nvidia* (11).

A comparação somente pôde ser feita utilizando o modelo *dragon* pois a aplicação não permite o carregamento de malhas externas. Desta forma, não foi identificado se a geometria do modelo disponibilizado pela aplicação é equivalente à utilizada nesta dissertação. O posicionamento de câmera foi aproximado. Esta aplicação permite a configuração de diversos parâmetros, os quais foram selecionados para qualidade alta.

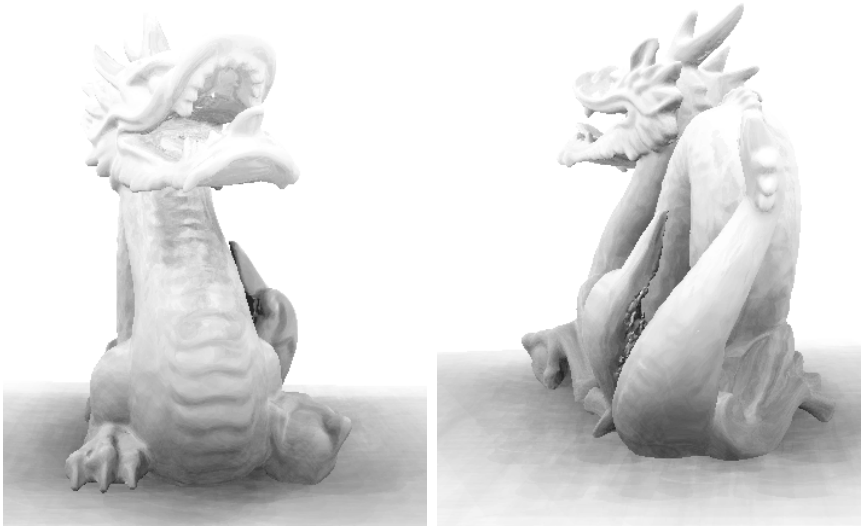
A Figura 4.11 apresenta imagens comparando algumas poses de câmera entre a aplicação em espaço de tela da *Nvidia*, o algoritmo proposto nesta dissertação e o Optix. A Tabela 4.12 apresenta a quantidade de quadros por segundo que a aplicação e o método proposto obtêm para cada pose de câmera. Os tempos de execução reproduzem o resultado esperado, um desempenho superior da aplicação em espaço de tela.

Tabela 4.12: Comparação entre os tempos obtidos utilizando a aplicação de oclusão ambiente em espaço de tela da *Nvidia* e o método proposto.

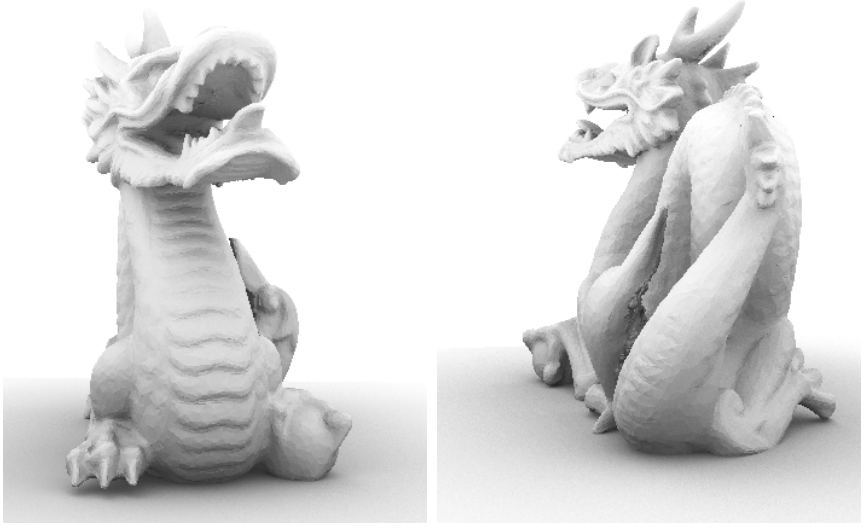
Algoritmo	Pose 1 (FPS)	Pose 2 (FPS)
Aplicação de oclusão ambiente em espaço de tela da *Nvidia*	54	56
Método proposto	40	41



Aplicação de oclusão ambiente em espaço de tela da *Nvidia*



Método proposto

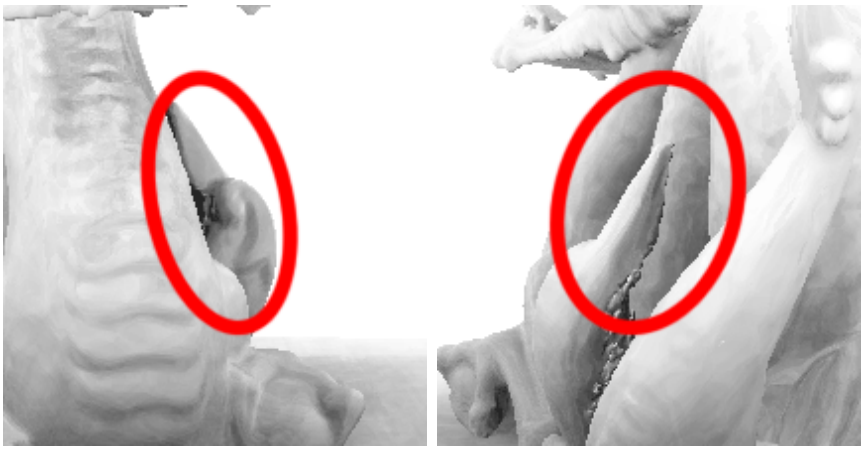


Optix

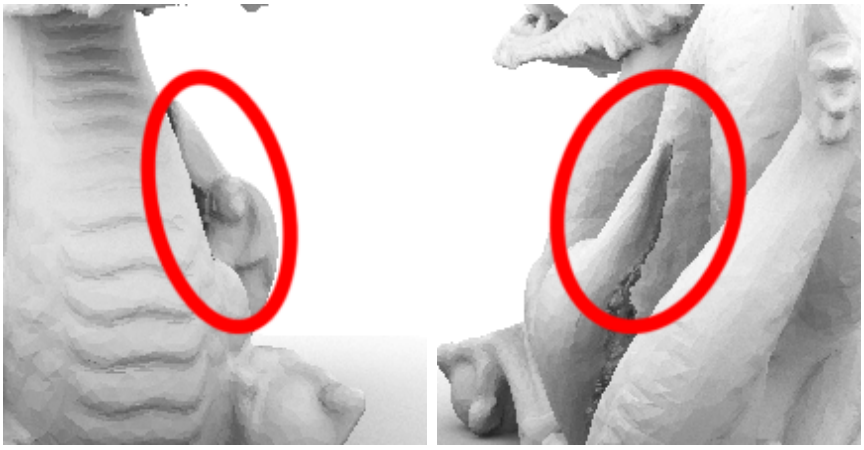
Figura 4.11: Comparação entre resultados obtidos utilizando a aplicação de oclusão ambiente em espaço de tela da *Nvidia* e o método proposto.

Os resultados dispostos nas imagens e na tabela permitem notar que ambos os métodos produzem resultados de boa qualidade, contudo o método proposto se aproxima mais da referência. A Figura 4.12 destaca as diferenças entre o algoritmo em espaço de tela em comparação com a utilização de dados volumétricos.

Aplicação de oclusão ambiente em espaço de tela da *Nvidia*



Método proposto



Optix

Figura 4.12: Detalhe das regiões não estimadas corretamenteo pela a aplicação de oclusão ambiente em espaço de tela da *Nvidia* em comparação com o método proposto.

4.7 Escalabilidade

Visando compreender o comportamento do método proposto em diversas situações, realizou-se a análise do algoritmo ao aumentar o número de *pixels* e ao aumentar a complexidade da geometria.

A Tabela 4.13 apresenta os tempos obtidos ao executar uma cena composta por 130 mil vértices e 65 mil triângulos em diversas resoluções. A Figura 4.13 exibe um gráfico destes tempos, demonstrando que o algoritmo tem desempenho linearmente proporcional ao número de *pixels*, tal como esperado.

Tabela 4.13: Resultados de tempo para a execução em diversas resoluções.

Resolução	Tempo (ms)	FPS
640x480	9.00	111.11
800x600	13.41	74.57
1024x768	21.44	46.64
1280x960	32.12	31.13
1600x1200	50.03	19.99
2048x1536	81.54	12.26

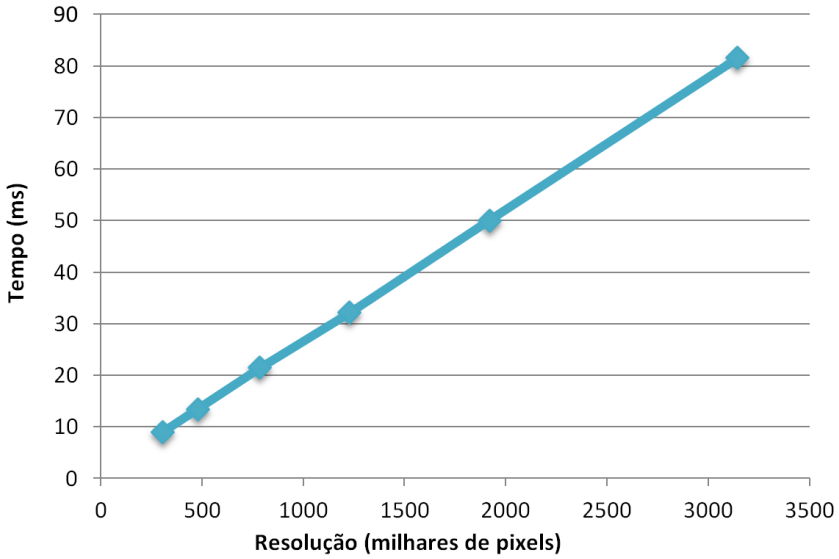


Figura 4.13: Gráfico exibindo o tempo em milissegundos para a execução de uma mesma cena em diversas resoluções.

A Tabela 4.14 dispõe os tempos alcançados com a execução de diversas cenas de complexidade geométrica variável. A geometria de cada cena também é apresentada na tabela. Para que o número de *pixels* gerados alterasse o mínimo possível a análise, para cada modelo foi realizado a contagem de *pixels*

através de *occlusion query* e a posição da câmera foi escolhida visando manter este valor aproximadamente constante.

Tabela 4.14: Resultados de tempo para a execução de cenas de diversas complexidades geométricas.

Modelo	Vértices (mil)	Triângulos (mil)	Tempo (ms)	FPS
bunzipper	35.95	69.45	7.85	127.39
jsyrlin	253.54	505.55	10.95	91.32
hand	327.32	654.67	10.78	92.76
bareliefply	506.51	1000.00	14.16	70.62
happyvrip	543.65	1087.72	13.29	75.24
buddha15M	757.49	1514.96	13.60	73.53

A Figura 4.14 apresenta o gráfico dos tempos obtidos. Nota-se uma tendência de um pequeno crescimento linear do tempo com o crescimento da complexidade da geometria, o que é esperado, devido às duas etapas inicias dependerem da geometria. Acredita-se que a flutuação apresentada no gráfico seja consequência da relação não proporcional entre o número de vértices e número de triângulos de cada cena.

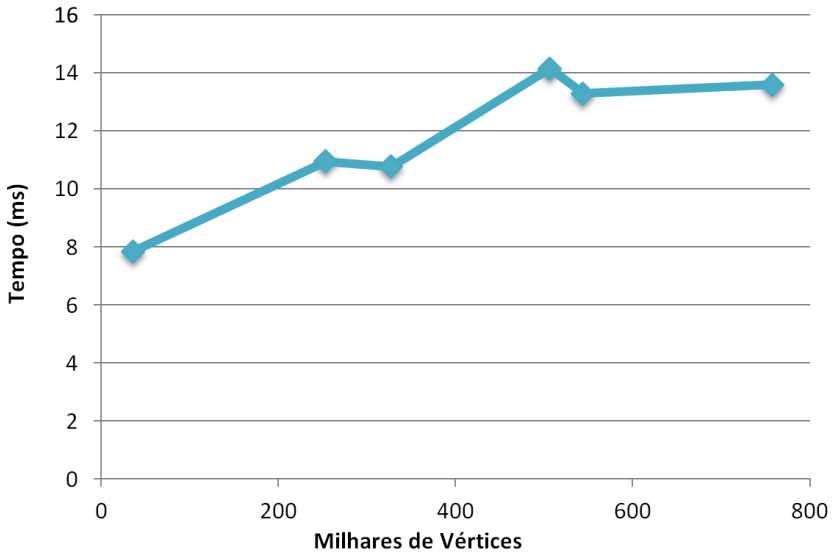


Figura 4.14: Gráfico exibindo o tempo em milissegundos para a execução de cenas de complexidades geométricas diferentes.