План дипломной работы
Коррекция многогранников с помощью локальных деформаций

Палачев Илья, 510 группа

23 марта 2012 г.

Содержание

1 Введение

1.1 Происхождение вопроса

Данная работа возникла из вопроса автоматизации производственного процесса огранки драгоценных камней.

Огранка подразумевает создание на поверхности камня ряда геометрически правильных плоскостей, граней, от которых будет отражаться попадающий внутрь кристалла свет. После многократного внутреннего отражения и преломления лучи света разделяются на спектральные составляющие и, покидая камень, создают игру оттенков на его поверхности. Огранка различается по сложности выполнения. В наиболее сложной и привлекательной бриллиантовой огранке может быть до 240 граней. Простая огранка создает на поверхности камня 30 плоскостей. Качественно выполненная огранка существенно повышает стоимость драгоценного камня, в то же время неправильная огранка может «убить» камень, внести в него дефекты. [из Wiki]

В конечном счете красота бриллианта определяется оптическими свойствами, как присущими веществу - алмазу, так и проявляющимися в результате огранки и полировки. Совокупность поверхностных и внутренних оптических свойств бриллианта человеческий глаз воспринимает как попавшие в глаз разнообразные лучи — белые и цветные, в целом создающие картину бриллианта. [из http://3dbook.octonus.com/data/3dbook.html]

Форма огранки бриллианта зависит от формы исходного кристалла алмаза. Чтобы получить бриллиант максимальной стоимости, огранщики стараются свести к минимуму потери алмаза при обработке. В зависимости от формы кристалла алмаза, при его обработке теряется 55—70 % веса. [из Wiki] Возможно ли автоматизировать этот процесс? Если да, то насколько? Разница всего в 1/10 карата может менять стоимость бриллианта в разы.

1.2 Особенности процесса сканирования алмазов

Чтобы автоматизировать процесс огранки, необходимо реализовать построение трехмерных моделей реальных алмазов. Для их создания используются сканеры, которые фотографируют камень. Как это происходит? Камень помещают перед источником света и фотографируют. При этом получается так называемый **теневой профиль** камня. Затем камень немного поворачивают и фотографируют еще раз. Так делается много раз и в результате на основе фотографий теневого профиля вращающегося алмаза строится 3D-модель. Опишем этот процесс подробнее. [http://www.octonus.com/oct/support/doc/scanning.html]

Сначала камень устанавливается на подставке - **держателе**. Эта подставка вращается и периодически производится фотографирование теневого профиля с определенным угловым шагом. Например, в сканере алмазов "Helium Polish"фотографирование может производиться в трех режимах [$http://www.octonus.com/oct/download/files/Helium\_system\_manual.pdf$]:

Метод	Число контуров	Время сканирования, с	Время сборки, с	Полное время, с
Quick	100	12	3	15
Accuracy	400	47	11	58
Hi accuracy	800	92	30	122

Из этой таблицы видно, что в разных режимах делается от 100 до 800 фотографий одного камня, то есть модель строится по 100, 400 или 800 теневым контурам. Может ли оказаться так, что два различных камня могут быть восприняты системой как одинаковые? Конечно, это возможно, если разница между ними так мала, что все теневые контуры у них будут идентичны. Всякую модель подобного рода можно понимать как пространственное пересечение нескольких бесконечных цилиндров, порожденных теневыми контурами.

Может возникнуть коллизия такого рода, когда сам камень имеет какую-то простую структуру, но система при фотографировании заменяет его на более сложный многогранник. Напимер, модель может иметь такой вид, как на рис. 1, тогда как в самом деле камень имеет вид, как на рис. 2. Это происходит за счет того, что угол α между осями проекции очень мал (порядка шага фотографирования).

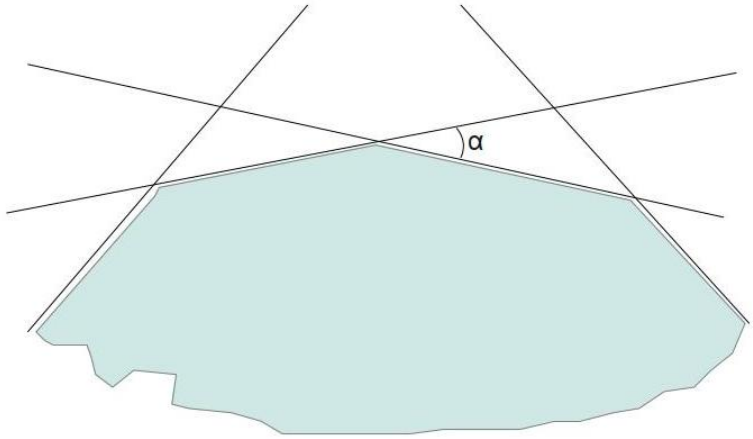


Рис. 1: Смоделированный многогранник

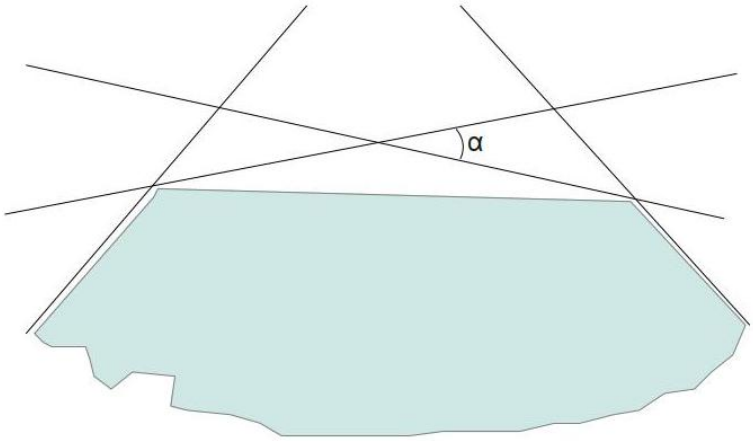


Рис. 2: Реальный камень

В таких ситуациях происходит заведомо неоправданное усложнение структуры многогранника. В 3D-моделях камней происходит разделение крупных граней на 2, 3, 4 более мелких. Это происходит за счет добавления в модель несуществующих в камне ребер. Как это может повлиять на дальнейшую работу с моделью? Если идет работа с одним камнем, то разница несущественна, но при массовом анализе

увеличение числа ребер в 2, 3 раза приводит к катастрофическому увеличению времени работы алгоритмов. Кроме того, лишние ребра препятствуют анализу характеристик камня по его модели. Так, если нужно будет найти площадь самой большой грани, то когда в модели эта грань будет разделена на 2, то характеристика будет вычислена неверно.

Эти соображения и послужили поводом написания данной работы. В ней рассмотрены различные способы коррекции многогранников при помощи локальных деформаций, без повторного анализа теневых контуров, то есть по уже построенной 3D-модели камня.

1.3 Что понимается под многогранником

Прежде всего следует сказать о том, каким образом задаются многогранники. Во-первых, зададается количество вершин и количество граней. Во-вторых, перечисляются координаты всех вершин (это тройки чисел x, y, z). В-третьих, перечисляются грани. При этом указывается, сколько вершин находится в грани, в какой плоскости она лежит (четырьмя числами a, b, c, d из уравнения плоскости $ax + by + cz + d = 0$), и список вершин, которые ее составляют.

Например, куб с центром в начале координат $(0, 0, 0)$ со сторонами длины 2, параллельными осям координат (см. рис. 3), задается в данной системе следующим образом:

Число вершин	Число граней
8	6

Номер вершины	x	y	z
0	-1	-1	-1
1	1	-1	-1
2	1	1	-1
3	-1	1	-1
4	-1	-1	1
5	1	-1	1
6	1	1	1
7	-1	1	1

Номер грани	Число вершин в грани	a	b	c	d	Список вершин
0	4	0	0	-1	-1	1, 0, 3, 2
1	4	0	-1	0	-1	0, 1, 5, 4
2	4	1	0	0	-1	1, 2, 6, 5
3	4	0	1	0	-1	2, 3, 7, 6
4	4	-1	0	0	-1	3, 0, 4, 7
5	4	0	0	1	-1	4, 5, 6, 7

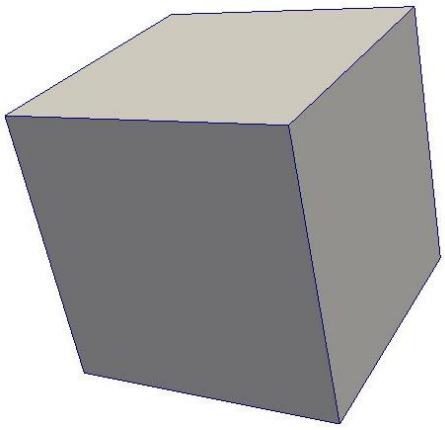


Рис. 3: Куб

Таким же образом задаются и прочие многогранники

2 Различные подходы к коррекции многогранников

2.1 Подход первый: сечение многогранника плоскостью

Постановка задачи

2.1.1 Простейший случай: выпуклый многогранник (сделано - 3 курс)

Описание алгоритма

2.1.2 Случай невыпуклого многогранника (сделано - 4 курс)

Описание алгоритма и демонстрация особых случаев (невуклости, многокомпонентность)

2.2 Подход второй: сглаживание двух смежных граней

Постановка задачи

2.2.1 Построение уравнения новой плоскости методом наименьших квадратов (используется готовый алгоритм)

Описание алгоритма

2.2.2 Построение уравнения новой плоскости методом наименьших квадратов с весами

Имеется в виду, что масса грани должна быть равномерно распределена. В обычном подходе этому мешают скопления точек, которые вынуждают плоскость пройти близко к ним, увеличивая погрешность на изолированных точках. Для устранения этого нужно придать изолированным точкам больший вес. Сделать это можно несколькими способами.
1. Можно считать вес по длинам ребер, исходящих из точки.
2. Можно считать вес по площади (какой?)
3. Может быть, следует обратиться к осям инерции (?).

2.2.3 Поднятие граней, которые лежат ниже новой плоскости (недоделано - 4 курс)

При этом возможно два подхода.
1. Строить грани итерационно (обходить контур).
2. Сначала все, что можно поднять - поднимаем, а потом - рассекаем многогранник. Этот подход пытался реализовать. Подъем граней должен быть итерационным.

2.3 Подход второй: подвижка вершины и дальнейшая перестройка многогранника

2.3.1 Идея задачи

2.3.2 Первый вариант формализации задачи (неудачный)

Имеется ввиду, что варьируются только координаты вершин. В общем случае система перепределена.

2.3.3 Второй вариант формализации задачи (неудачный)

Имеется ввиду, что варьируются и вершины, и грани. Тогда функционал является многочленом четвертой степени и его минимизировать проблематично.

2.3.4 Третий вариант формализации задачи (удачный)

Здесь же вариация происходит поочередно:
1. Вычисляем плоскости методом наименьших квадратов
2. Изменяем координаты вершин таким образом, чтобы минимизировать функционал

$$\sum_{i=1}^{N-1}(|\Delta x_i|^2 + |\Delta y_i|^2 + |\Delta z_i|^2) + K\sum_{j=1}^{M}\sum_{i:A_i\in\pi_j}|a_j x_i + b_j y_i + c_j z_i + d_j|^2 \to \min$$

При этом нужно помнить, что коэффициенты уравнений плоскостей на этом шаге уже фиксированы, и менять их не нужно. Переменная K называется штрафом
3. Возвращаемся к пункту 1. и т. д.

Это всего лишь примерный план реализации алгоритма. В действительности есть много его модификаций: от тех, которые дают довольно адекватный результат, до тех, которые полностью 'выворачивают' многогранник наизнанку.

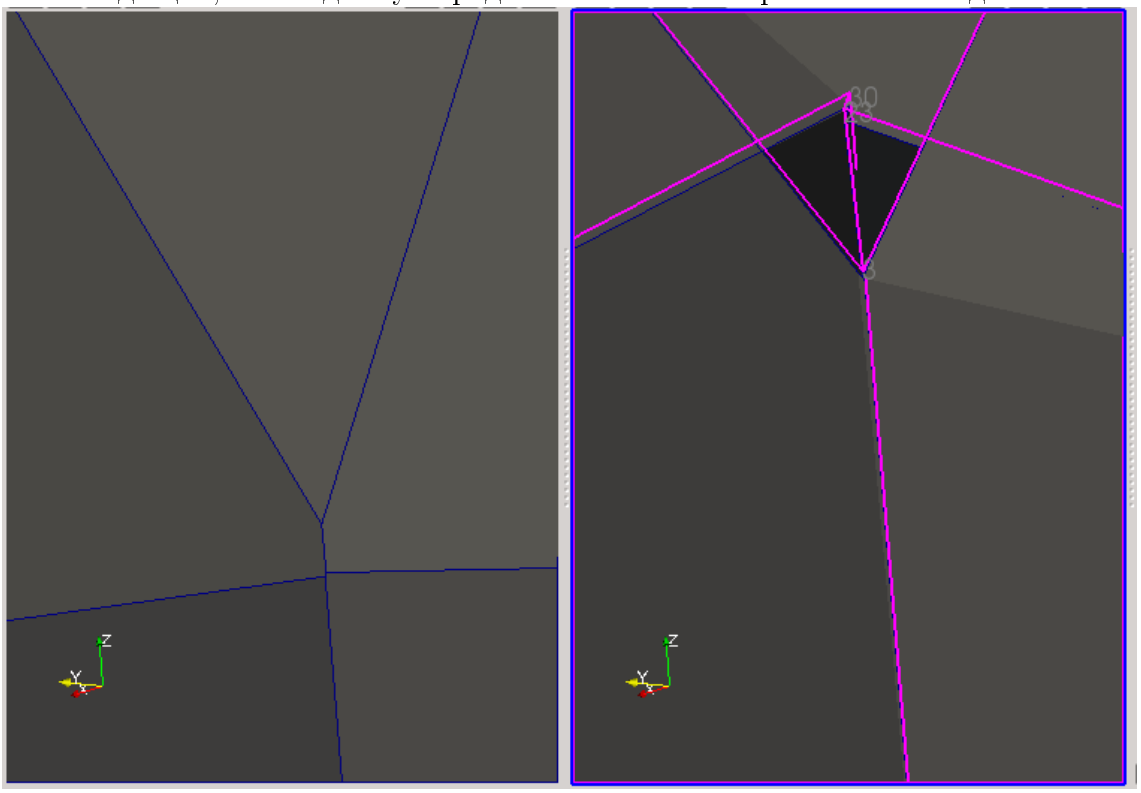
Множественность возникает из-за того, что задаче есть два изменяемых параметра:
1. Штраф K. Каким его сделать? Если слишком большим, то алгоритм перестанет 'чувствовать' смещения вершин, а если слишком малым, то наоборот - алгоритм будет плохо решать задачу восстановления плоскостности многогранника.
2. Как чередовать шаги 1 и 2? Менять ли при этом штраф K?

В результате тестирования наиболее приемлемым оказался следующий подход:
1. $K = 100$
2. $\epsilon = 10^{-7}$
3. Цикл 1. Делать 10 раз:
3.1. Делать цикл 2:
3.1.1. Вычислить новые плоскости методом наименьших квадратов
3.1.2. Вычислить координаты вершин, минимизирующих функционал со штрафом, равным K
3.2.3. Вычислить погрешность попадания вершин в плоскости
3.2 ...пока эта погрешность больше, чем ϵ
3.3 Конец цикла 2
3.4 Увеличиваем K в два раза
3.5 Уменьшаем ϵ в два раза
4. Конец цикла 1.

Как этот алгоритм работает? На тестируемых многогранниках он выдал следующее поведение. Программа заходит в цикл 1, затем в нем - в цикл 2, делает примерно 30 шагов, выходит из 2 цикла, снова начинает 2 цикл (с увеличенным в 2 раза штрафом и с уменьшенной в 2 раза погрешностью), но теперь уже делает только 1 шаг, снова выходит из 2 цикла и т. д. - всего примерно 10 раз. Вот таблица результатов алгоритма (Δ - сумарный сдвиг вершин, ϵ - погрешность попадания вершин в плоскости, ϵ_0 - граница погрешности, пересечение которой обеспечивает конец цикла 2, K - штраф):

цикл 1	цикл 2	Δ	ϵ	ϵ_0	K
0	0	0.114942	4.220043e-02	1.000000e-07	100.000000
0	1	0.048647	2.640706e-02	1.000000e-07	100.000000
0	2	0.018739	1.637491e-02	1.000000e-07	100.000000
0	3	0.007239	1.015491e-02	1.000000e-07	100.000000
0	4	0.002856	6.344031e-03	1.000000e-07	100.000000
0	5	0.001151	3.991388e-03	1.000000e-07	100.000000
0	6	0.000472	2.528785e-03	1.000000e-07	100.000000
0	7	0.000196	1.634950e-03	1.000000e-07	100.000000
0	8	0.000082	1.064094e-03	1.000000e-07	100.000000
0	9	0.000035	6.940496e-04	1.000000e-07	100.000000
0	10	0.000015	4.535194e-04	1.000000e-07	100.000000
0	11	0.000006	2.967911e-04	1.000000e-07	100.000000
0	12	0.000003	1.944614e-04	1.000000e-07	100.000000
0	13	0.000001	1.275375e-04	1.000000e-07	100.000000
0	14	0.000001	8.370982e-05	1.000000e-07	100.000000
0	15	0.000000	5.497753e-05	1.000000e-07	100.000000
0	16	0.000000	3.612497e-05	1.000000e-07	100.000000
0	17	0.000000	2.374594e-05	1.000000e-07	100.000000
0	18	0.000000	1.561315e-05	1.000000e-07	100.000000
0	19	0.000000	1.026795e-05	1.000000e-07	100.000000
0	20	0.000000	6.753804e-06	1.000000e-07	100.000000
0	21	0.000000	4.442928e-06	1.000000e-07	100.000000
0	22	0.000000	2.923018e-06	1.000000e-07	100.000000
0	23	0.000000	1.923204e-06	1.000000e-07	100.000000
0	24	0.000000	1.265446e-06	1.000000e-07	100.000000
0	25	0.000000	8.326840e-07	1.000000e-07	100.000000
0	26	0.000000	5.479375e-07	1.000000e-07	100.000000
0	27	0.000000	3.605724e-07	1.000000e-07	100.000000
0	28	0.000000	2.372805e-07	1.000000e-07	100.000000
0	29	0.000000	1.561485e-07	1.000000e-07	100.000000
0	30	0.000000	1.027587e-07	1.000000e-07	100.000000
0	31	0.000000	6.762423e-08	1.000000e-07	100.000000
1	32	0.000000	2.585086e-08	5.000000e-08	200.000000
2	33	0.000000	9.229164e-09	2.500000e-08	400.000000
3	34	0.000000	3.159339e-09	1.250000e-08	800.000000
4	35	0.000000	1.053913e-09	6.250000e-09	1600.000000
5	36	0.000000	3.462331e-10	3.125000e-09	3200.000000
6	37	0.000000	1.131674e-10	1.562500e-09	6400.000000
7	38	0.000000	3.720410e-11	7.812500e-10	12800.000000
8	39	0.000000	1.253741e-11	3.906250e-10	25600.000000
9	40	0.000000	4.448228e-12	1.953125e-10	51200.000000

Нужно реализовать следующие идеи:
1. Попробовать разную динамику изменения штрафа. Не только экспоненциальный рост, но и линейный, логарифмический...
2. Решить проблему “выворачивания” многогранника. Дело в том, что если начальная деформация точки слишком велика, то нарушается корректность многогранника (грани выворачиваются или наезжают друг на друга). В продемонстрированном выше примере точка двигалась на расстояние порядка 0.1, тогда как одно из ребер, из нее исходящих, имеет длину порядка 0.03. Вот что при этом наблюдается:



При решении проблемы выворачнивания нужно решить, вообще говоря, три задачи:
2.1. На какое максимальное расстояние расстояние можно двигать точку, не изменяя при этом топологической структуры многогранника?
2.2. Попытаться делать деформацию не разом за одно применение программы, а, скажем, делить вектор сдвига на 10 частей и за 10 применений программы реализовать сдвиг.
2.3. Если эти две задачи решены, то может быть попытаться решить задачу, когда сдвиг больше допустимого, то есть когда мы сначала сдвигаем точку на максимально допустимое расстояние, затем меняем немного структуру многогранника, а затем сдвигаем еще раз. (???)

2.3.5 Анализ сходимости и трудоемкости алгоритма

Про сходимость алгоритма до сих пор ничего не известно, доказать не удалось. (TODO)