МОУ лицей №31 г. Челябинска

Исследовательская работа в номинации: «Адсорбент».

Конкурс исследовательских работ школьников "Первый шаг в наномир".

Выполнил:

Ученик 10в класса МОУ лицей №31

г. Челябинск

Устинов Илья Николаевич

Научный руководитель:

Аспирант ЧелГУ

Преподаватель МОУ лицей №31

Красников Василий Сергеевич

Челябинск 2009

Введение

Целью данной работы является нахождение площади поверхности образца.Для работы предоставлено изображение поверхности, полученное с помощью атомно-силового микроскопа.

Предоставленное изображение содержит хорошо различимые глазом шумы – точки или группы точек с сильно отличающимся от соседних значением высоты, которые могут вносить значительные искажение результатов. Для устранения шумов был разработан двухшаговый метод, включающий в себя градиентное и матричное сглаживание. На первом шаге – градиентном сглаживании - происходит сглаживание резких выбросов, второй шаг – матричное сглаживание – используется для устранения мелкой ряби. Подсчёт площади осуществляется с помощью метода треугольников, адаптивно выбирающего направление наименьшего искривления поверхности при разбиении. Разработанные алгоритмы реализованы в виде программы на языке Borland Pascal.

Для конвертирования bmp файлов в txt и наоборот использовалась программа Горшенина Владимира Викторовича, http://physolymp.fml31.ru/olymp/files/f264.exe)

Преобразование BMP-файла

Я работал не непосредственно с графическим файлом, а с соответствующим ему текстовым файлом цветов пикселей рисунка. Изображение предоставлено в BMP-файле, BMP-файл содержит служебную информацию, содержащуюся в заголовке файла и следующим за ним матрицу интенсивностей цветов изображения. В случае черно-белого изображения интенсивность цвета линейно связана с высотой. Для преобразования BMP-изображения в текстовый файл использовалась Горшенина Владимира Викторовича, http://physolymp.fml31.ru/olymp/files/f264.exe), созданная с использованием стандартных библиотек языка C++. После выполнения этой программы BMP-изображение преобразуется в текстовый файл, содержащий таблицу высот точек исходного изображения.

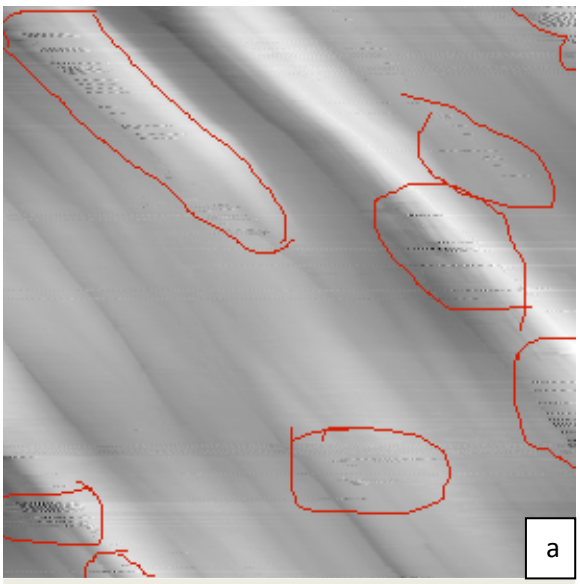
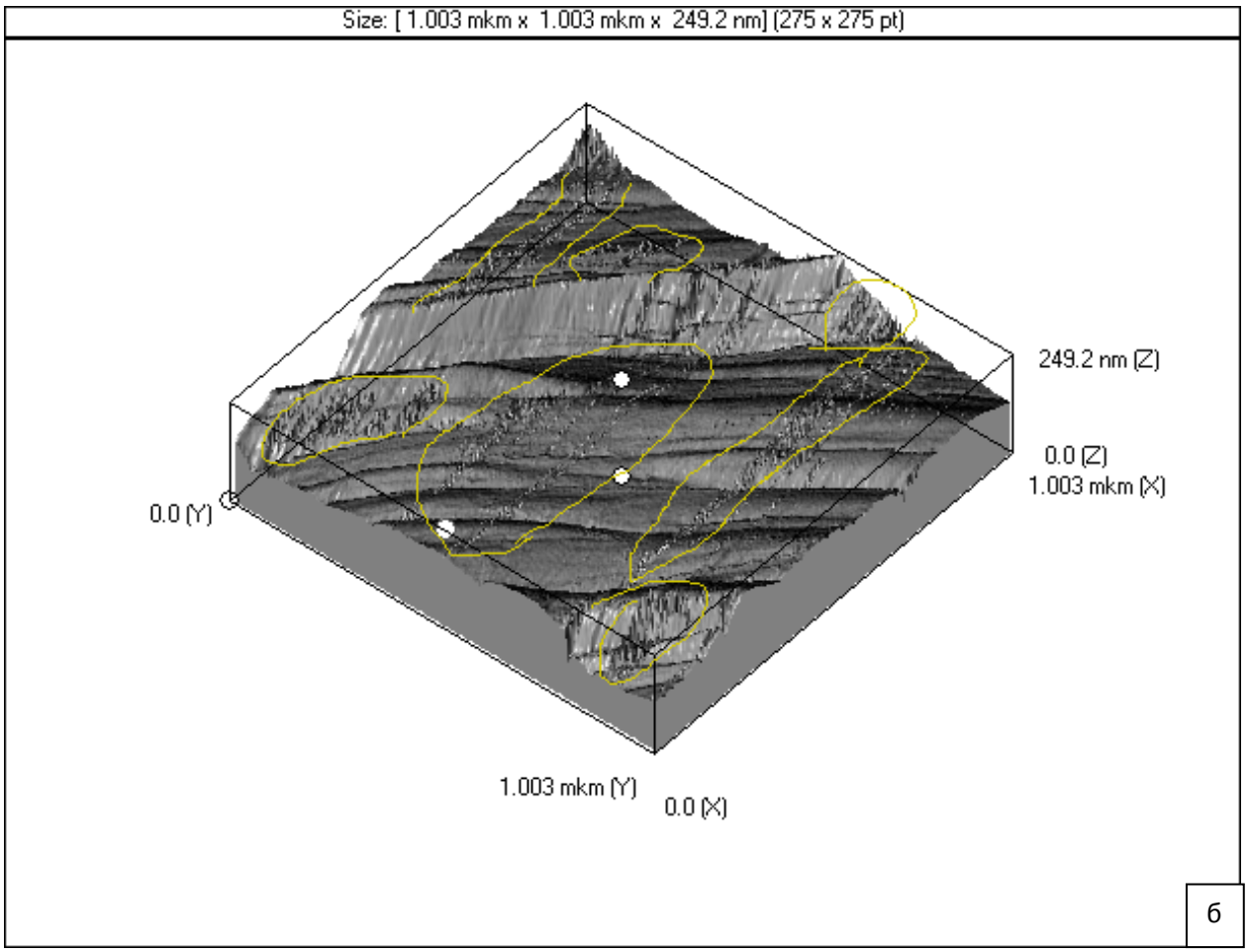


Рис. 1. Исходное изображение с отмеченными областями, содержащими резкие выбросы.
а – 2-d изображение.
б – 3-d изображение.

Методы анализа данных

1.Визуальное сравнение поверхности.

2.После просмотра фотографий (рис.1а,б) поверхности в 3d и 2d режимы обнаружились точки (или группы точек) с сильно отличающимся от соседних значением высоты (цвета) – шумы.



Из-за больших различий в высотах они могут вносить существенный вклад в значение площади поверхности, которое необходимо получить, следовательно, их необходимо устранить.

Способ устранения шумов

Для устранения шумов использовались 2 способа: градиентный и матричный. Рассмотрим их отдельно.

Алгоритм градиентного устранения шумов

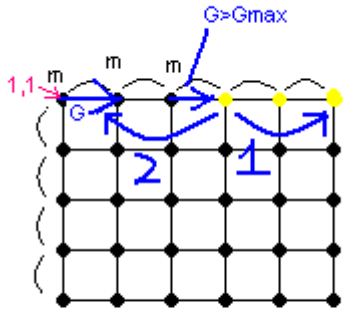
Смысл это способа состоит в том, чтобы заменить высоты точек, при переходе к которым градиент по модулю больше порогового значения, которое задается вручную, высотами, значения которых соответствуют окружающим незашумленным точкам.

Градиент – производная, вычисляемая вдоль некоторого направления. Поскольку в нашем случае поверхность задана в виде дискретного набора точек - это величина, показывающая тангенс угла между прямой, проведенной через 2 точки, имеющие определенную высоту, и плоскостью рисунка.

Градиент можно вычислить: G= (B-A)/R, где R-расстояние между точками (в плоскости 2d снимка, а A и B – значения их высот. Видно, что градиент может быть как отрицательный, так и положительный, и необходимо это учитывать.

В программе создается матрица, содержащая в своих ячейках высоты точек, координаты которых на плоскости снимка (каждому пикселю соответствуют свои порядковые номера от начала рисунка (левый верхний угол) по 2-ум осям, содержащим края рисунка) соответствуют номерам ячейки матрицы.

1)Просматриваем матрицу высот точек графического изображения построчно (рис.2), изначально статус «хорошей точки» m (m-просто обозначение; в программе существует переменная m, содержащая порядковый номер хорошей точки (по строке или столбцу, в зависимости оттого, что мы просматриваем – строку или столбец)) присваиваем самой первой точке в строке. «Хорошая точка» - точка, которая не зашумлена.



Жёлтые точки- шумы

Рис. 2. Схема сглаживания изображения вдоль строки.

Проходим матрицу по строкам, высчитывая градиенты между соседними точками (второй и первой, третьей и второй, и т.д.). Если градиент Gi (текущее значение градиента)<=Gmax (пороговое значение), то статус «хорошей» (m) передаём рассматриваемой точке. Если значение градиента больше порогового, то необходимо найти «хорошую точку», но не являющуюся m. Для этого проходим строку вначале в сторону увеличения порядкового номера точек, высчитывая градиенты между текущей точкой и точкой m. Как только найдется «нормальный» градиент, не превосходящий по значению пороговый, значение высоты зашумленной точки находим как значение «хорошей» сложенное с градиентом:

A [i+1]=A[i]+G; где A [i+1] - зашумленная точка, A[i]- хорошая точка, G - «нормальный» градиент.

Если же такого градиента не нашлось, то аналогично просматриваем строку в сторону уменьшения порядкового номера точек. В случае если «нормального» градиента не нашлось – то рассматриваемый шум неустраним (по строке).

3)Аналогичным способом проходим по столбцам.

Повторное прохождение матрицы по столбцам необходимо для тех случаев, когда вся строка, или зашумлена, или в тех случаях, когда есть большая строка шумов (точно также прохождение по строкам устраняет большие зашумленности в столбцах). Также прохождения по столбцам и строкам компенсируют ошибки друг друга, связанные с выбором первой точки как «хорошей точки», хотя их не должно быть много, если считать появление зашумленной точки – случайным явлением.

Матричное сглаживание

Однако в градиентном методе есть и свои минусы – он позволяет убрать шумы, резко отличающиеся от окружающих по высоте («пики»), и не затрагивает мелкие шумы (мелкая «рябь»). Этот способ не влияет на шумы, градиент которых меньше максимального (максимальный задается пользователем).

На самом деле наличие мелких шумов пока только предположение, поскольку их наличие, опираясь на графическое изображение, доказать невозможно.

Для сглаживания возможных мелких шумов используется дополнительно матричный способ.

Он заключается в том, что каждой точке присваивается значение высоты, равное среднему между значением этой точки и окружающих её (в матрице размера 3x3).

Алгоритм: проходим матрицу от 2-ой до предпоследней строки, и от 2-го до предпоследнего столбца, изменяя значения высот точек. Основная особенность алгоритма в том, что брать значения для усреднения необходимо из исходной матрицы , а получившиеся значения записывать в другую , иначе будет происходить излишнее плавное уменьшение средней высоты точек на рисунке.

Подсчёт площади

Площадь рисунка находится как сумма площадей составляющих его маленьких участков - четырехугольников, образованных четырьмя соседними точками, такое разбиение поверхности выбирается для обеспечения максимальной точности. Предполагается, что на таких масштабах (расстояние между соседними точками ≈ 3,66 нм) поверхность достаточно гладкая.

Площадь же четырехугольников находится как сумма площадей составляющих их треугольников. Отдельным вопросом является способ разбиения на треугольники. Очевидно, что при выборе разбиения нужно выбрать такую диагональ, чтобы поверхность в наплавлении диагонали вела себя более спокойно, что уменьшит ошибку подсчёта, т.е. градиент в ее направлении был наименьшим (по модулю), поскольку это указывает на более спокойное поведение

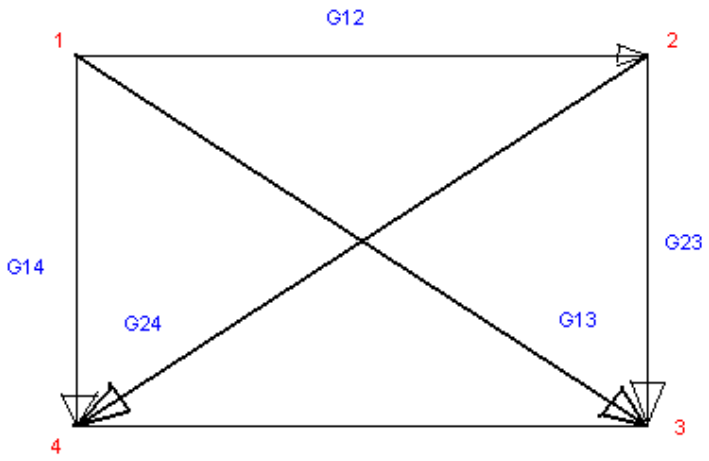


Рис. 3. Схема вычисления градиентов

поверхности, а значит и более точный подсчёт площади, потому что при выборе направления в котором поверхность ведёт себя более спокойно модель из двух треугольников больше совпадает с реальной ситуацией.

В силу особенностей табличного задания функций градиент по диагонали недостаточно полно описывает искривленность поверхности, поэтому при подсчёте определим термин «градиент в направлении диагонали» как среднеквадратичное из градиентов в направлении сторон, прилежащих к данной точке. Этот градиент вычисляется для каждой из точек четырехугольника, так для точки 1:

$$G13 = \sqrt{G12^2 + G14^2},$$

а для точки 2:

$$G24 = \sqrt{G12^2 + G23^2}.$$

Аналогично вычисляются градиенты для точек 3 и 4. Такой способ расчета градиентов позволяет учитывать, что градиент в каждой точке зависит от градиентов 2-ух сторон, причём простое среднеарифметическое не подойдёт, поскольку, как мы выяснили раньше градиент может быть как положительный так и отрицательный, и, если градиент

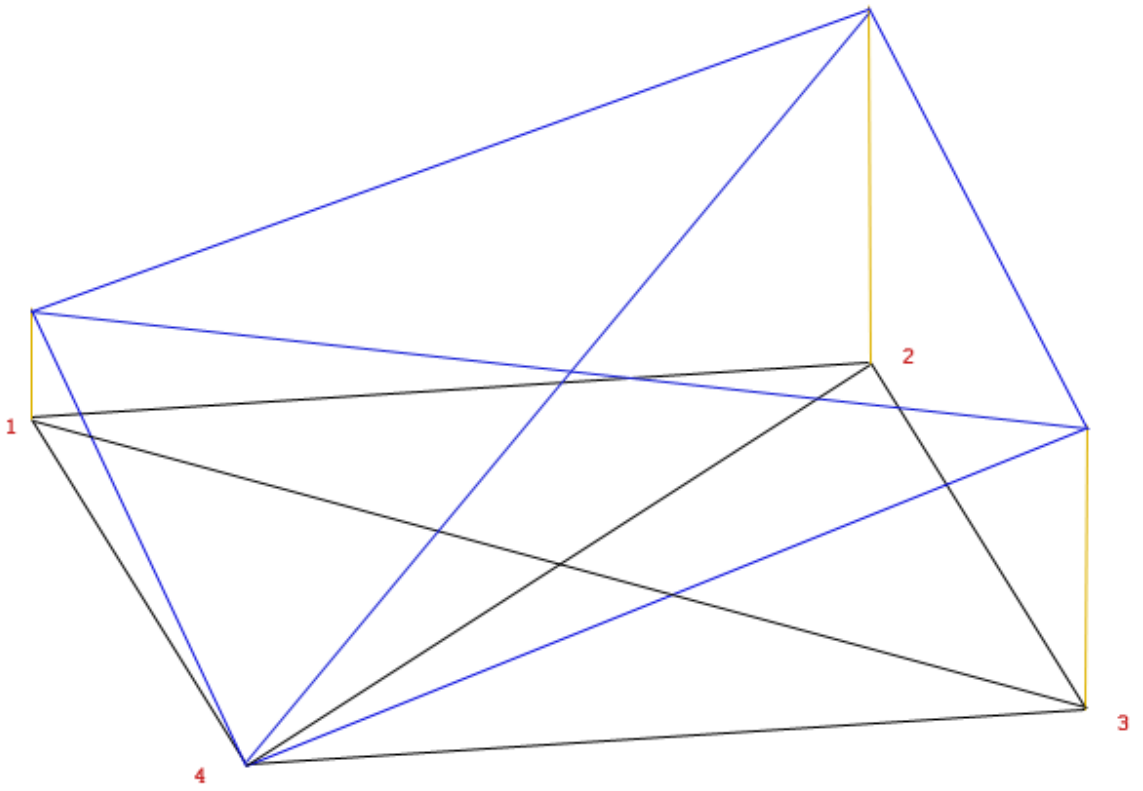


Рис. 4. К алгоритму выбора диагонали.

одной стороны G, а другой – G, то среднеарифметическое равно нулю, но очевидно что наклон диагонали в этом случае не равен нулю (т.е. необходимо отбросить знаки для подсчёта величины градиента).

Далее изложен алгоритм выбора диагонали разбиения. Рассмотрим на примере рис. 4. На картинке изображен четырёхугольник, высота точки 2 которого сильно отличается от высот точек 1 и 4,но высота точки 3 чуть больше половины высоты точки 2(принимая нулевой уровень – высоту точки 4) причём высота точки 1 не сильно отличается от высоты точки 4. При использовании вышеописанной наименьшим градиентом, очевидно, будет градиент 4-2.Из-за большой высоты точки 2 градиенты 1-2 будет очень большим ,2-3 и 3-4 будут большим, а значит и градиенты 1-3 и 3-1, по сравнению с 4-2, а градиент 2-4 вообще складывается из 2-ух больших градиентов(2-1,2-3). Но очевидно, что поверхность в гораздо меньшей степени изменяет свои характеристики вдоль линии 1-3 , в то время, как вдоль линии 2-4 она очень непредсказуема. И более правильным будет разбиение по диагонали 1-3

Из всего вышесказанного следует, что при выборе диагонали разбиения следует сравнивать противоположенные точки каждой диагонали, и градиенты от них.

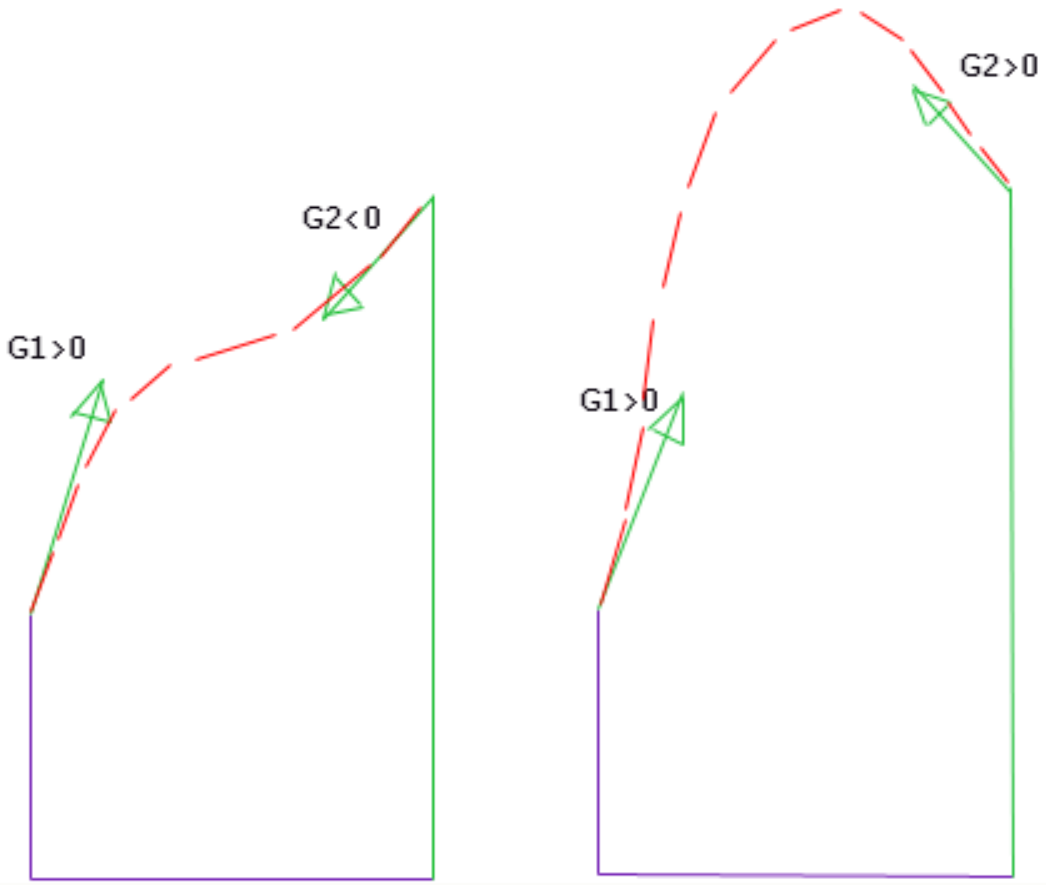


Рис.5. Возможные поведения поверхностей в зависимости от направления градиентов.

На рис.5 показаны два сечения поверхности, для которой известны величины градиентов её двух крайних точек, отличаются 2 изображения сечения только направлением градиентов вдоль одной диагонали, однако видно, что это также значительно влияет на поведение поверхности и, как следствие, её площадь (пунктиром показано возможное

сечение поверхности). Тогда можно внести поправку в алгоритм: предпочтительней диагональ, на концах которой градиенты имеют разный знак (или равны нулю). Т.е. при сравнении диагоналей сначала смотрится - какие имеют на концах градиенты разных знаков.

В вышеприведенных рассуждениях учитывается знак градиента. Рассмотрим, как можно определить его знак. Представим градиент сторон четырёхугольника вектором, который совпадает с вектором, связывающим две точки четырёхугольника. Этот метод используется только для определения знака градиента, величина определяется по прежним формулам, описанным выше. Для оценки будем считать градиент вдоль диагонали четырёхугольника, как векторную сумму двух прилежащий градиентов (градиентов сторон чётырёхугольника). Знак градиента диагонали примем совпадающим со знаком координаты соответствующего вектора по вертикальной оси. Тогда знак градиента по диагонали определяется только перепадом высот по смежным сторонам (поскольку он и является координатой вектора по вертикальной оси).

Алгоритм

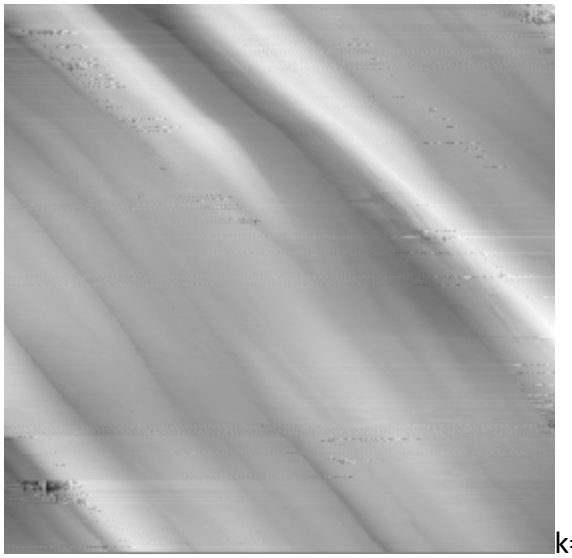
1) Переменной, содержащей общую площадь, изначально присваиваем 0.
2) Просматриваем матрицу, выделяя четырехугольники.
3) Рассматриваем четырехугольник. Ищем диагональ, вдоль которой будем разбивать четырёхугольник на треугольники:
   Находим величины градиентов на концах диагоналей, и их знаки.
   Если градиенты разных знаков по обеим диагоналям, то выбираем диагональ, для которой сумма градиентов по диагонали противоположенных концов меньше по модулю.
   Если у одной диагонали градиенты вдоль неё с двух сторон разных знаков, а у другой – нет, то выбираем диагональ с разными знаками градиентов.
   Если вдоль обеих диагоналей градиенты одинаковых знаков, то выбираем диагональ, у которой сумма градиентов на концах меньше по модулю (сравниваются именно суммы, потому что сравнивать разности неправильно, например: на концах первой диагонали градиенты очень большие и одинаковые, а на концах второй – меньше, но разные по величине, в этом случае разность для первой диагонали равна нулю, а для второй – не равна нулю, и выбрав диагональ с меньшей разностью градиентов мы сделаем ошибку, а выбирать диагональ с большей разностью вообще не логично ,сумма же учитывает величины градиентов на обоих концах).
4) Разбиваем на два треугольника вдоль выбранной диагонали. Считаем площади треугольников по формуле Герона.
5) Увеличиваем значение переменной содержащей общую площадь на полученные величины.

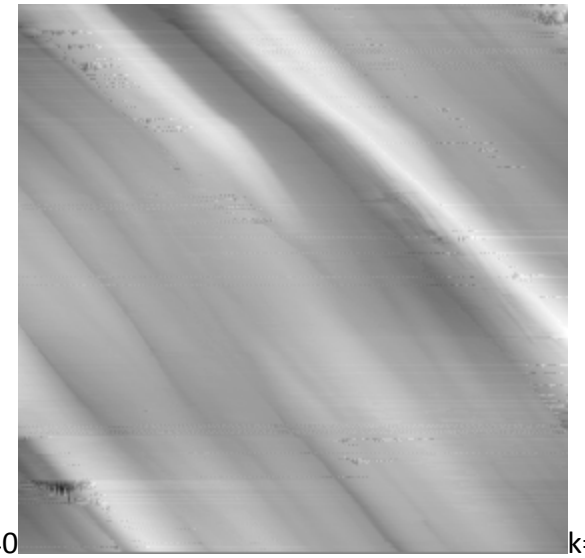
Результаты анализа данных

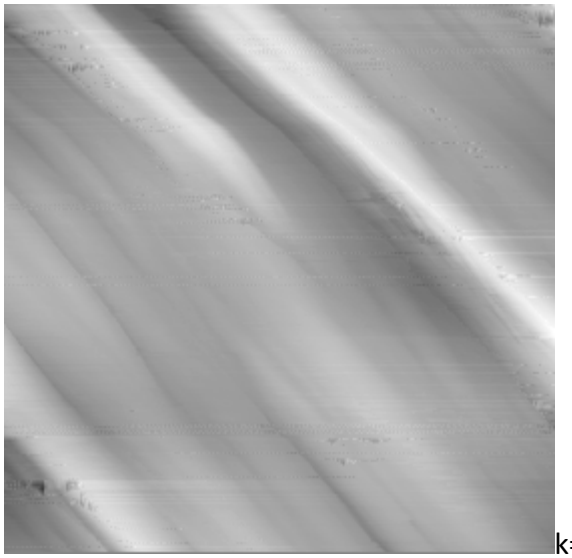
Программа написана на Borland Pascal 7.0 .

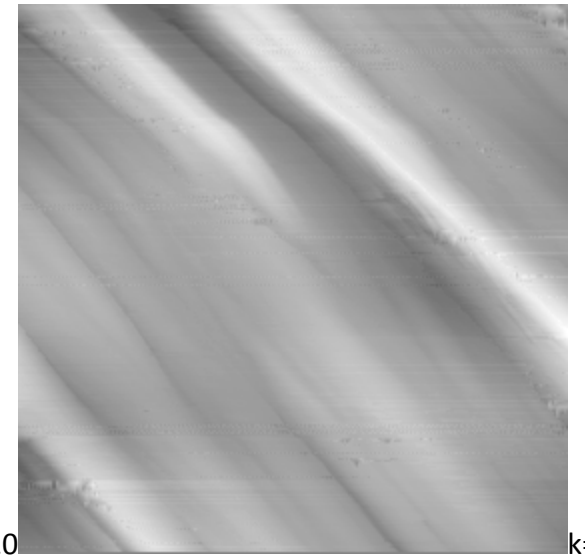
1.Сравнение картинок, сглаженных с разными значениями максимального допустимого градиента, приводит к выводу, что при пороговом значении градиента равном 5 ед. на рисунке наименьшее количество шумов, и он наиболее чёткий.

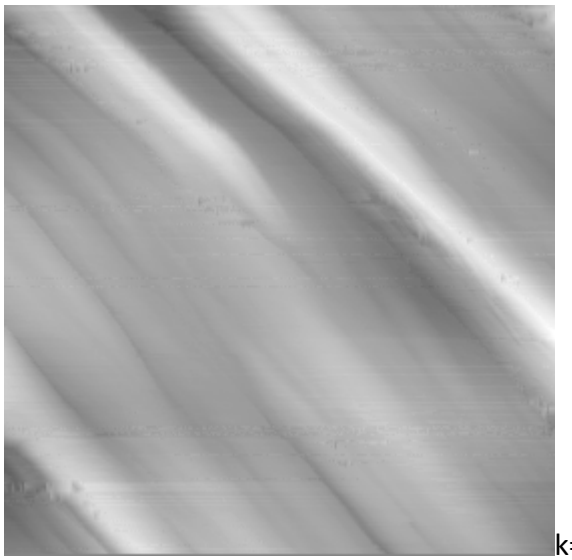
(k – величина порогового значения градиента)

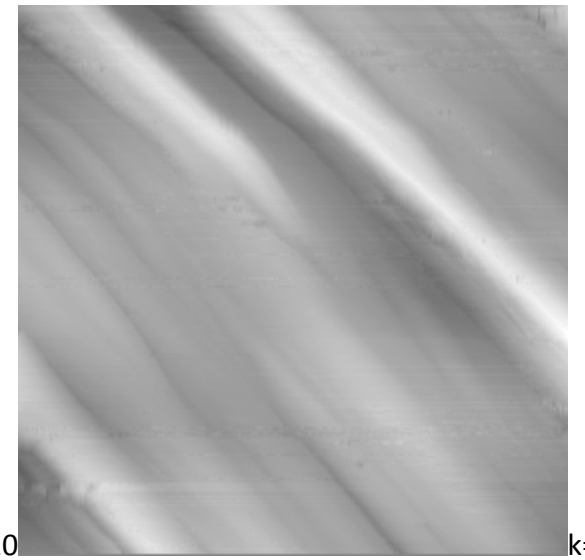

k=40

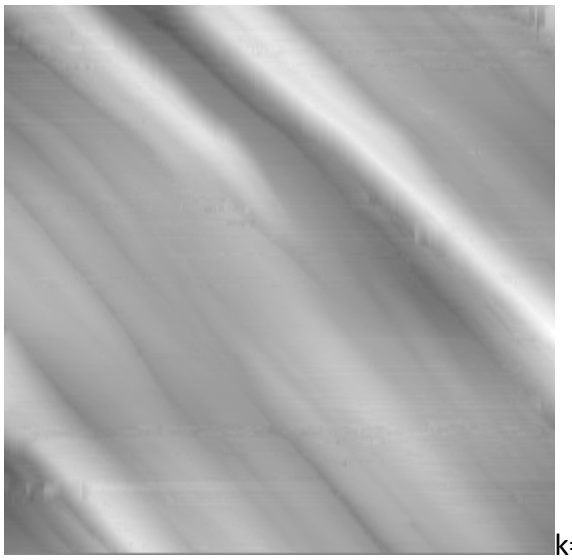

k=30

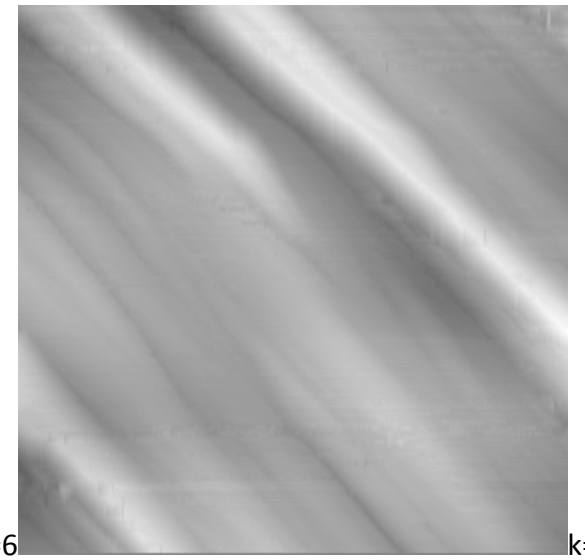

k=20

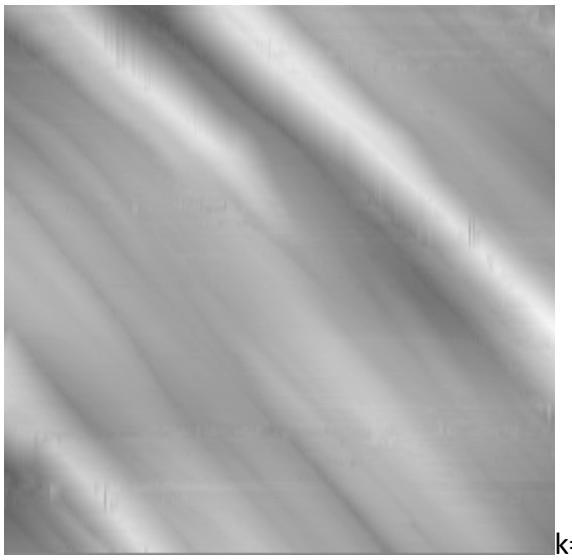

k=15

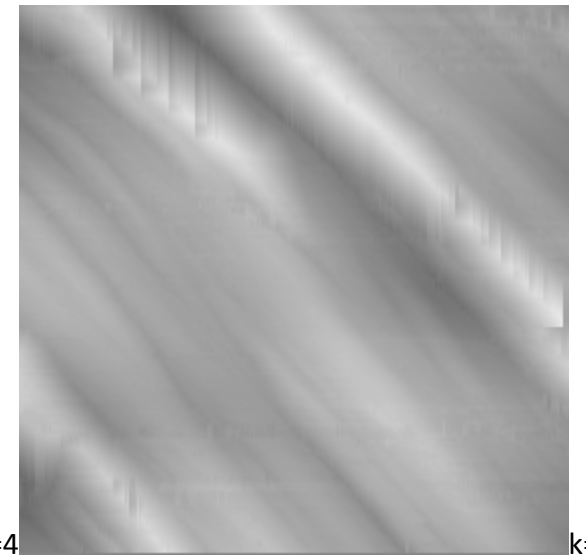

k=10

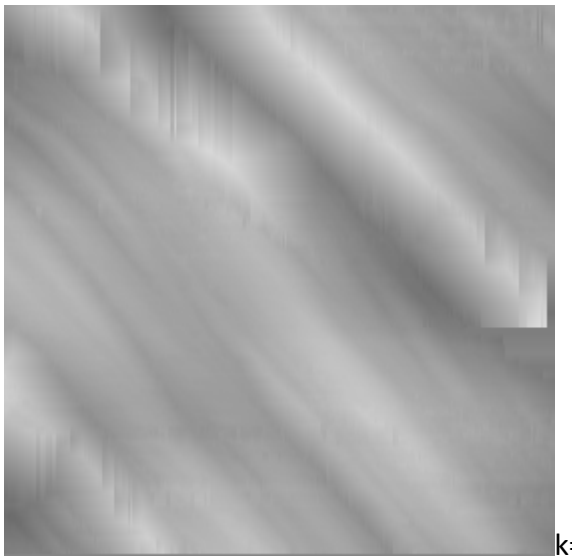
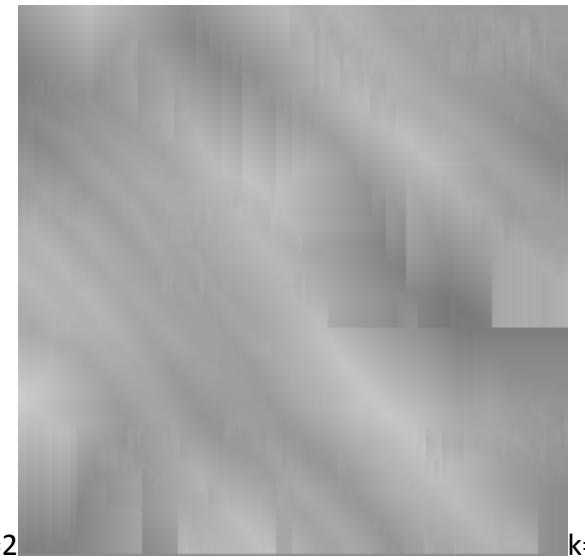

k=8


k=6

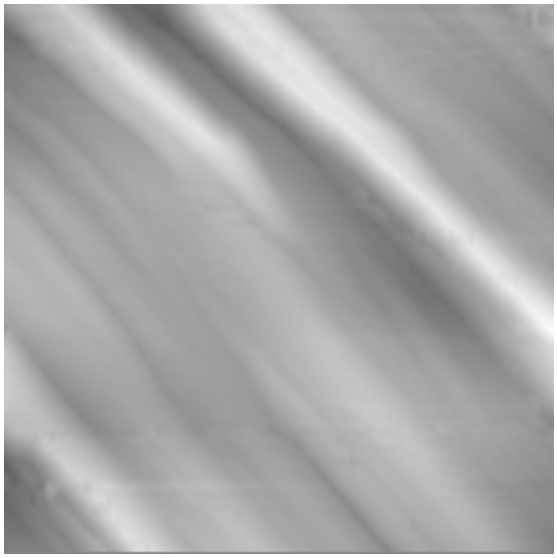

k=5


k=4


k=3


k=2

k=1

(Видно, что при использовании k<5 картинка начинает размываться, а при k>5 сильно выделяются полосы и группы шумов)



Этот рисунок – после сглаживания матричным методом поверх градиентного метода с k=5:

Видно, что поверхность стала ещё более чёткой.

Для такого порогового значения площадь сглаженной поверхности равна 1201698.0706 нм², а отношение площадей – 1.1945.

При нахождении отношения площадь части идеального шара можно считать ровной поверхность (квадрат), поскольку его радиус много больше размеров рассматриваемого участка, и кривизной можно пренебречь.

2.Во всех пунктах анализа данных может содержаться ошибка.

Вклад в ошибку может вноситься при: считывании, сглаживании, и подсчете площади.

Ошибка при считывании высот из файла считается нулевой.

Тогда остается только 2 источника ошибки.

Для проверки возможного влияния ошибок на результат проведено пару тестов:

1) Для сглаживания: посчитаны дисперсии высот точек на поверхности до сглаживания, и после.

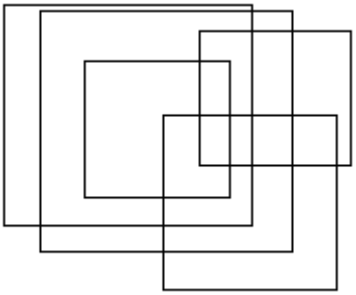
Дисперсия показывает среднеквадратичное отклонение от среднего значения, в нашем случае - отклонение высот точек, и определяется по формуле:

$$= \sqrt{\frac{\sum_{i=1}^{n} (X[i] - Xcp)^2}{n}}$$

В этой формуле: X[i]-значение i элемента, Xcp – среднеарифметическое значение всех X[i], n – количество элементов.

Начальная дисперсия (до сглаживания):24.7291 нм.
Конечная дисперсия (после сглаживания):23.4001 нм.
Видно, что дисперсия уменьшилась, что свидетельствует о сглаживании поверхности.

Проверкой эффективности градиентного устранения шумов может служить рисунок, содержащий тонкостенные квадраты на ровной плоскости (рис.6а). В этом случае стороны квадрата должны устраняться программой, как шумы.



До : а После: б

Рис.6. Тест для градиентного сглаживания
а – исходный рисунок.
б – после сглаживания.

Действительно, видно, что резкие шумы, расположенные в произвольном порядке устраняются полностью (рис.6б).

Также, в программе созданы переменные – счётчики шумов, которых невозможно устранить в процессе сглаживания (по столбцам и по строкам), после работы программы значения этих переменных оказались равными нулю. Т.е. механизм сглаживания эффективен в данном случаё ещё и из-за отсутствия неустранимых шумов.

2) Для площади: создадим 2 рисунка, площадь которых относительно просто посчитать аналитически, и сравним с результатами подсчёта площади программой. 1-ый рисунок – чёрный квадрат на белом фоне, его просто получить в Paint (рис.7).

Рис.7. Первый тест для метода подсчёта площади.

2-ой русинок – наклонная плоскость (рис.8), её зададим в виде текстового файла цветов с помощью созданной для этой цели программы.

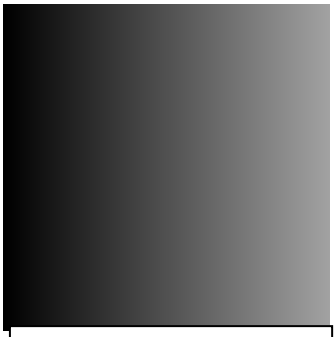


Рис.8. Второй тест для метода подсчёта площади.

Квадрат на плоскости:

Его площадь будет складываться из площади ровной поверхности, и площади стенок квадрата. (Я изменил масштаб при выполнении теста, для удобных подсчётов)

Аналитически: 86052.6003 ед. квадратных.

Программа: 85760.6215 ед. квадратных.

Наклонная плоскость:

Аналитически:26244.2018 ед. квадратных.

Программа:26244.2018 ед. квадратных.

Видно, что максимальная погрешность всего 0.3%, и то она возможно связанна с проблемами подсчёта площади поверхности около углов квадрата (1-ый тестовый рисунок).

Т.е. в простых случаях выдаётся довольно точный результат.

Площадь поверхности без сглаживания получилась равной 1,751 мкм².

Только с градиентным сглаживанием: 1,286 мкм²

С градиентным и матричным: 1,197 мкм²

Т.е. применение градиентного сглаживания уменьшает площадь на 26,6% от площади несглаженной поверхности, а применение матричного метода поверх градиентного уменьшает площадь ещё на 5%, теперь можно сказать, что предположение о мелких шумах было верно.

Описание программных средств

Основная программа называется Physics. Программа для создания текстового файла, содержащего данные о рисунке наклонной плоскости, называется Create. При запуске основной программы необходимо указать название текстового файла, содержащего информацию о рисунке, с которым необходимо работать, вместе с расширением (.txt). Результаты сглаживания основной программой сохраняются в виде нового текстового файла. Его название – output.txt.

Текстовый файл created.txt содержит информацию о рисунке с наклонной плоскостью. Test.txt – о рисунке для первого теста площади (черный квадрат на белом фоне). А sh_2d.txt содержит информацию об исходном снимке.

В результате выполнения программы Physics будет выдана на печать следующая информация: disp – дисперсия после сглаживания, disp0 – дисперсия без сглаживания, d – отношение площади поверхности к площади квадрата размерами 1.003 мкм на 1.003 мкм,S_sum – суммарная площадь поверхности,Stmax – максимальная из посчитанных за всю работу программы площадь четырёхугольника образованного четырьмя близлежащими точками, Stcurrent – последняя из посчитанных площадей четырёхугольника (По сути две последние величины не необходимы для получения результатов исследовательской работы, я их использовал только для улучшения алгоритма подсчёта площади).

Выводы

1) Разработан и реализован в виде программы алгоритм сглаживания и вычисления площади поверхности по исходному изображению.
2) Сглаживание поверхности осуществляется в два этапа, каждый из которых направлен на устранение определенного типа шумов.
3) Использование матричного и градиентного методов сглаживания позволяют с большей точностью определить площадь поверхности.
4) Использование адаптивного метода выбора направления разбиения четырехугольника также уменьшает ошибку в определении поверхности.
5) По результатам расчетов площадь поверхности шарика адсорбента увеличена на 28,2% от площади идеально-ровной поверхности, стоит заметить, что эта площадь равна идеально-ровной поверхности шара радиусом в 1.13 раз больше.

Приложение 1. Код основной программы.

```
{В этом тексте все сообщения выводимые на экран переведены с английского.
Они сразу не выполнены на русском языке, поскольку моя версия Borland
Pascal не поддерживает русской кодировки.}
program physics;
uses crt;
const m_xy=1003/274;{масштаб по осям x,y}
      m_z=249.2/255;{масштаб по вертикальной оси }
      K=5;{пороговое значение градиента}
      N1max=230;{максимальное количество строк в матрице}
      N2max=275;{максимальное количество столбцов в матрице}
type mas=array[1..N2max] of byte;
var b: array[1..2] of mas;{вспомогательная матрица для хранения рез-татов
матричного сглаживания}
    a: array[1..N1max] of mas ;{основная матрица}
    N1,N2,{размеры изображения}
    N1t,{текущее количество строк в матрице}
    i,
    j: integer;{вспомогательные переменные}
    name: string;{название входного файла}
    l12,l13,l14,l23,l24,l34,{переменные, хранящие длины сторон
четырёхугольника и его диагоналей}
    Stmax,Stcurrent,{дополнительные переменные для отладки программы,
хранящие последнюю посчитанную площадь и максимальную посчитанную площадь
в течение действия программы}
    S_sum,{Сумарная площадь поверхности}
    sr,disp,disp0,{переменные необходимые для подсчёта дисперсии}
    d,{искомое отношение площадей}
    G: real;{градиент}
    f1,
    f2: text;{текстовые файлы}
    z,{вспомогательная переменная}
    m,{номер базовой хорошей точки]
    l,{номер не базовой хорошей точки}
    i1,j1,v,{вспомогательные переменные}
    err1,err2,{счётчики неустранимых шумов}
    r1,r2: integer;{счётчики устраненных шумов}
    flag_1_4: boolean;{флаг разбиения четырехугольника по диагонали 1-4}
function avg3(i,j:integer):byte;{функция считающее среднее значение высоты
в матрице 3x3}
var n,p:integer;
    sum:integer;
    sr:real;
begin
  sum:=0;
  for n:=i-1 to i+1 do
   for p:=j-1 to j+1 do
    sum:=sum+a[n,p];
  sr:=sum/9;
  avg3:=round(sr);
end;
procedure find_grad(i,j:integer;var flag_1_4:boolean);{поиск диагонали
разбиения}
                                          {  1.(i,j)   2.(i,j+1)    }

                                          {  3.(i+1,j) 4.(i+1,j+1)  }
var grad_1_4,grad_4_1,grad_2_3,grad_3_2:real;
    f_4_1,f_2_3:boolean;
begin
  grad_4_1:=sqrt(abs(abs(sqr(a[i,j+1]-a[i+1,j+1]))+abs(sqr(a[i+1,j]-
a[i+1,j+1]))));
```

```
  grad_2_3:=sqrt(abs(abs(sqr(a[i+1,j+1]-a[i,j+1]))+abs(sqr(a[i,j+1]-
a[i,j]))));
  grad_1_4:=sqrt(abs(abs(sqr(a[i,j+1]-a[i,j]))+abs(sqr(a[i+1,j]-
a[i,j]))));
  grad_3_2:=sqrt(abs(abs(sqr(a[i+1,j+1]-a[i+1,j]))+abs(sqr(a[i+1,j]-
a[i,j]))));
  if (a[i+1,j]+a[i,j+1]-2*a[i+1,j+1])<>0 then begin

grad_4_1:=grad_4_1*(a[i+1,j]+a[i,j+1]-2*a[i+1,j+1]);

grad_4_1:=grad_4_1/abs(a[i+1,j]+a[i,j+1]-2*a[i+1,j+1]);
                                              end;

  if (a[i+1,j+1]+a[i,j]-2*a[i,j+1])<>0 then begin

grad_2_3:=grad_2_3*(a[i+1,j+1]+a[i,j]-2*a[i,j+1]);

grad_2_3:=grad_2_3/abs(a[i+1,j+1]+a[i,j]-2*a[i,j+1]);
                                            end;
  if (a[i+1,j+1]+a[i,j]-2*a[i+1,j])<>0 then begin

grad_3_2:=grad_3_2*(a[i+1,j+1]+a[i,j]-2*a[i+1,j]);

grad_3_2:=grad_3_2/abs(a[i+1,j+1]+a[i,j]-2*a[i+1,j]);
                                            end;
  if (a[i+1,j]+a[i,j+1]-2*a[i,j])<>0 then begin

grad_1_4:=grad_1_4*(a[i+1,j]+a[i,j+1]-2*a[i,j]);

grad_1_4:=grad_1_4/abs(a[i+1,j]+a[i,j+1]-2*a[i,j])
                                           end;
  f_4_1:=((grad_4_1>0)and(grad_1_4<0))or((grad_4_1<0)and(grad_1_4>0));
  f_2_3:=((grad_2_3>=0)and(grad_3_2<=0))or((grad_2_3<=0)and(grad_3_2>=0));
  if (f_2_3)and(f_4_1) then
flag_1_4:=abs(grad_1_4+grad_4_1)<abs(grad_2_3+grad_3_2)
  else if f_2_3 then flag_1_4:=false;
       if f_4_1 then flag_1_4:=true;
       if not(f_2_3 or f_4_1) then
flag_1_4:=abs(grad_1_4+grad_4_1)<abs(grad_2_3+grad_3_2);
end;
function length(a1,a2:byte):real;{длина отрезка между двумя точками
четырёхугольника, но не лежащих на одной диагонали}
var a:real;v:integer;
begin
  v:=a2-a1;
  a:=sqrt(sqr((m_z)*v)+sqr(m_xy));
  length:=a;
end;
function length1(a1,a2:byte):real;{длина отрезка между двумя точками
лежащими на одной диагонали}
var v:integer;a:real;
begin
  v:=a2-a1;
  a:=sqrt(sqr((m_z)*v)+2*sqr(m_xy));
  length1:=a;
end;
function St(l1,l2,l3:real):real;{площадь треугольника}
var a,p,p1,p2,p3:real;
begin
  p:=(l1+l2+l3)/2;
  p1:=p-l1;
```

```
  p2:=p-l2;
  p3:=p-l3;
  a:=sqrt(abs(p*p1*p2*p3));
  St:=a;
end;
begin
  disp:=0;
  disp0:=0;
  sr:=0;
  Stmax:=0;{начальные присвоения}
  clrscr;
  writeln('Введите название входного файла (*.txt)');
  readln(name);
  assign(f1,'name');{связываем входной файл с переменной}
  reset(f1);
  readln(f1,N1);
  readln(f1,N2);{считываем размеры}
  writeln('Размеры изображения:  ',N1,'x',N2);
  n1t:=1;
  {нахождение средней высоты точек на исходном снимке}
  while not EOF(f1) do
  begin
     N1t:=0;
     for i:=1 to N1max do
     begin
      for j:=1 to N2 do
       begin
        read(f1,v);
        a[i,j]:=v;
       end;
      readln(f1);
      if EOF(f1) then begin N1t:=i; break; end;
     end;
    if N1t=0 then N1t:=N1max;
    for i:=1 to N1t do
     for j:=1 to n2 do
      sr:=sr+a[i,j];
   end;
  sr:=sr/275;
  sr:=sr/275;
  close(f1);reset(f1);
  readln(f1);
  readln(f1);
  n1t:=1;
  {нахождение дисперсии на исходном снимке}
  while not EOF(f1) do
   begin
    N1t:=0;
     for i:=1 to N1max do
     begin
      for j:=1 to N2 do
       begin
        read(f1,v);
        a[i,j]:=v;
       end;
      readln(f1);
      if EOF(f1) then begin N1t:=i; break; end;
     end;
    if N1t=0 then N1t:=N1max;
    for i:=1 to N1t do
     for j:=1 to n2 do
```

```
      disp0:=disp0+sqr(a[i,j]-sr);
    end;
  close(f1);
  reset(f1);
  sr:=0;
  disp0:=disp0/275;
  disp0:=disp0/275;
  disp0:=sqrt(disp0);
  if N2>N2max then
    begin
     writeln('Ошибка ввода данных (Максимальное число точек в строке -
',N2max,').');
     writeln('Нажмите любую клавишу');
     readkey;
     exit;
    end;
  assign(f2,'output.txt');{связываем выходной файл с переменной}
  rewrite(f2);
  writeln(f2,N1);
  writeln(f2,N2);
  close(f2);
  append(f2);
  err1:=0; r1:=0;
  err2:=0; r2:=0;
  S_sum:=0;
  sr:=0;
  readln(f1);
  readln(f1);
  for i:=1 to 2 do
   for j:=1 to n2 do
    begin
      read(f1,v);
      a[i,j]:=v;
    end;
  readln(f1);
  N1t:=2;
  while not EOF(f1) do {основной цикл}
   begin
    a[1]:=a[N1t-1];
    a[2]:=a[N1t];
    N1t:=0;
    {считывание с разбиением на несколько матриц}
     for i:=3 to N1max do
     begin
      for j:=1 to N2 do
       begin
        read(f1,v);
        a[i,j]:={255-}v;
       end;
      readln(f1);
      if EOF(f1) then begin N1t:=i; break; end;
     end;
    if N1t=0 then N1t:=N1max;
    for i:=1 to N1t do {устранение шумов по строкам}
     begin
      m:=1;{базовая хорошая точка}
      for j:=2 to N2 do
       begin
         G:=(a[i,j]-a[i,m])/(j-m);
         if Abs(G)<K then m:=m+1
         else
```

```
            begin{поиск хорошей точки, но не m}
              l:=0;
              for j1:=j+1 to N2 do
                if Abs((a[i,j1]-a[i,m])/(j1-m))<K then
                  begin
                    l:=j1;
                    break;
                  end;
              if l=0 then
                for j1:=m-1 downto 1 do
                  if Abs((a[i,m]-a[i,j1])/(m-j1))<K then
                    begin
                      l:=j1;
                      break;
                    end;
              if l>0 then
                begin
                 if l>m then G:=(a[i,l]-a[i,m])/(l-m)
                 else G:=(a[i,m]-a[i,l])/(m-l);
                 a[i,j]:=a[i,m]+round(G);
                 m:=m+1;
                 r1:=r1+1;{увеличиваем счётчик устраненных шумов по
строкам}
                end
              else err1:=err1+1;{увеличиваем счётчик неустранимых шумов по
строкам}
            end;  {else}
        end;
    end;
    for j:=1 to N2 do
    begin {устранение шумов по столбцам}
     m:=1;
     for i:=2 to N1t do
       begin
          G:=(a[i,j]-a[m,j])/(i-m);
          if Abs(G)<K then m:=m+1
          else
           begin
              l:=0;
              for i1:=i+1 to N1t do
                if Abs((a[i1,j]-a[m,j])/(i1-m))<K then
                  begin
                    l:=i1;
                    break;
                  end;
              if l=0 then
                for i1:=m-1 downto 1 do
                  if Abs((a[m,j]-a[i1,j])/(m-i1))<K then
                    begin
                      l:=i1;
                      break;
                    end;
              if l>0 then
                begin
                 if l>m then G:=(a[l,j]-a[m,j])/(l-m)
                 else G:=(a[m,j]-a[l,j])/(m-l);
                 a[i,j]:=a[m,j]+round(G);
                 m:=m+1;
                 r2:=r2+1;
                end
              else err2:=err2+1;
```

```
           end; {else}
       end;
      end;
{матричное сглаживание}
      b[1]:=a[2];
      for j:=2 to n2-1 do
        b[1,j]:=avg3(2,j);
      for i:=3 to N1t-1 do
      begin
       b[2]:=a[i];
       for j:=2 to N2-1 do
        b[2,j]:=avg3(i,j);
       a[i-1]:=b[1];
       b[1]:=b[2];
      end;
      a[i]:=b[1];
{считается сумма значений высот в сглаженной матрице, для дальнейшего
нахождения среднего значения, матрица записывается в выходной файл}
    for i:=1 to N1t-2 do
     begin
      for j:=1 to N2 do
      begin
       sr:=sr+a[i,j];
       v:=a[i,j];
       write(f2,v,chr(9));
      end;
      writeln(f2);
     end;
   for i:=1 to N1t-3 do{подсчёт площади}
    for j:=1 to N2-1 do
     begin
      find_grad(i,j,flag_1_4);{находится направление наименьшей
искривленности поверхности}
      l12:=length(a[i,j],a[i,j+1]);{рассчитываются стороны в
четырёхугольнике}
      l13:=length(a[i,j],a[i+1,j]);
      l14:=length1(a[i,j],a[i+1,j+1]);
      l23:=length1(a[i,j+1],a[i+1,j]);
      l24:=length(a[i,j+1],a[i+1,j+1]);
      l34:=length(a[i+1,j],a[i+1,j+1]);
      if flag_1_4{непосредственно находится площадь одного
четырёхугольника}
       then
        begin
         S_sum:=S_sum+St(l12,l24,l14)+St(l13,l34,l14);
         Stcurrent:=St(l12,l24,l14)+St(l13,l34,l14);
         if Stcurrent>Stmax then Stmax:=Stcurrent;
        end
      else
      begin
       S_sum:=S_sum+St(l12,l13,l23)+St(l24,l34,l23);
       Stcurrent:=St(l12,l13,l23)+St(l24,l34,l23);
       if Stcurrent>Stmax then Stmax:=Stcurrent;
      end;
     end;
    end;{while}
   for i:=N1t-1 to N1t do
    begin
     for j:=1 to N2 do
      begin
       sr:=sr+a[i,j];
```

```
      v:=a[i,j];
      write(f2,v,chr(9));
     end;
     writeln(f2);
    end;
     end;
  {учитываем вклад в сумму значений высот точек снимка и сумму площадей от
двух последних строк в  файле (из-за способа разбиения на несколько матриц
и использования дополнительной матрицы при матричном сглаживании эти две
строки не были учтены ранее), и записываем их в выходной файл}
  for i:=N1t-2 to N1t-1 do
   for j:=1 to N2-1 do
    begin
     find_grad(i,j,flag_1_4);
     l12:=length(a[i,j],a[i,j+1]);
     l13:=length(a[i,j],a[i+1,j]);
     l14:=length1(a[i,j],a[i+1,j+1]);
     l23:=length1(a[i,j+1],a[i+1,j]);
     l24:=length(a[i,j+1],a[i+1,j+1]);
     l34:=length(a[i+1,j],a[i+1,j+1]);
     if flag_1_4
      then
       begin
        S_sum:=S_sum+St(l12,l24,l14)+St(l13,l34,l14);
        Stcurrent:=St(l12,l24,l14)+St(l13,l34,l14);
        if Stcurrent>Stmax then Stmax:=Stcurrent;
       end
     else
     begin
      S_sum:=S_sum+St(l12,l13,l23)+St(l24,l34,l23);
      Stcurrent:=St(l12,l13,l23)+St(l24,l34,l23);
      if Stcurrent>Stmax then Stmax:=Stcurrent;
     end;
    end;
  sr:=sr/275;
  sr:=sr/275;
  close(f2);
  reset(f2);
  readln(f2);
  readln(f2);
  while not EOF(f2) do
   begin
    N1t:=0;
     for i:=1 to N1max do
     begin
      for j:=1 to N2 do
       begin
        read(f2,v);
        a[i,j]:=v;
       end;
      readln(f2);
      if EOF(f2) then begin N1t:=i; break; end;
     end;
    if N1t=0 then N1t:=N1max;
    for i:=1 to N1t do {считается дисперсия для сглаженного снимка}
     for j:=1 to n2 do
      disp:=disp+sqr(a[i,j]-sr);
   end;
  disp:=disp/275;
  disp:=disp/275;
  disp:=sqrt(disp);
```

```
  d:=S_sum;{находится отношение площадей}
  d:=S_sum/1003;
  d:=d/1003;
  close(f1);
  close(f2);
  writeln('Поиск по строкам: устранено шумов ',r1,'; найдено неустранимых
шумов ',err2);
  writeln('Поиск по столбцам: устранено шумов ',r2,'; найдено неустранимых
шумов ',err2);
  writeln('Произведена запись в выходной файл.');
  writeln('disp        :   ',disp:3:4,'       nm');
  writeln('disp0       :   ',disp0:3:4,'       nm');
  writeln('d           :   ',d:10:4);
  writeln('S_sum       :   ',s_sum:10:4,' nm*nm');
  writeln('Stmax       :   ',Stmax:4:4,'      nm*nm');
  writeln('Stcurrent   :   ',Stcurrent:4:4,'      nm*nm');
  writeln('Нажмите любую клавишу ...');
  readkey;
end.
```