АЛГОРИТМЫ ПОИСКА РАССТОЯНИЙ ДО ОБЪЕКТНЫХ ПИКСЕЛОВ НА БИНАРНЫХ ИЗОБРАЖЕНИЯХ

Н.Л. Казанский, В.В. Мясников, Р.В. Хмелев
Институт систем обработки изображений РАН

Постановка задачи

Одной из важнейших задач обработки изображений является задача измерений на изображении [1], в частности, поиск расстояний между точками. Пусть есть бинарная картинка, значения 0 будем называть фоном, значения 1 – объектом. Необходимо для каждой фоновой точки найти расстояние (в пикселах) до ближайшей к ней объектной (будем называть это расстояние расстоянием привязки), и сделать это нужно, используя минимальное количество операций. Эта задача возникает, например, при сравнении с эталоном по отклонению контуров [2]. В данной статье описываются два алгоритма решения этой задачи.

Рекурсивный алгоритм

<u>Простейший алгоритм</u>

Рассмотрим простейший алгоритм поиска наиболее близкого объектного пиксела для текущего фонового пиксела *X* (пример работы алгоритма на рис. 1). Итеративно сканируем периметры квадратов с длиной стороны $2n+1$ (где n – номер итерации, начиная с 1); расстояние от точки *X* до стороны квадрата равно n. Квадрат расстояния от точки *X* до сканируемой объектной точки *Y* (рис. 1, справа) находится по формуле: $r^2 = n^2 + m^2$, где m - расстояние от середины стороны квадрата, на которой лежит точка *Y*, до самой точки *Y* (будем называть середину стороны квадрата «срединной точкой»). При этом m^2 находится рекурсивно по следующим формулам:

- от угла до середины
  $m_{i+1}^2 = (m_i - 1)^2 = m_i^2 - 2m_i + 1$;
- от середины до угла
  $m_{i+1}^2 = (m_i + 1)^2 = m_i^2 + 2m_i + 1$.

Сканирование сторон квадратов продолжается до тех пор, пока n^2 не становится большим или равным минимуму из найденных квадратов расстояний.

Сокращенное сканирование

Способ поиска, при котором периметр квадрата сканируется последовательно, не самый экономичный. На рис. 2 показан поиск объектной точки от срединной точки *Z* к краю для верхней стороны квадрата (сканирование нижней и боковых сторон происходит аналогично). Как только найдена объектная точка *A* на расстоянии m_1 от *Z*, дальше в эту сторону сканировать не следует, так как расстояние от точки *X* при этом только увеличится. Аналогично просканируем в другую от *Z* сторону. Предположим, найдена точка *B* на расстоянии m_2 от *Z*, и $m_1 < m_2$. Из двух расстояний $r_1 = \sqrt{n^2 + m_1^2}$ и $r_2 = \sqrt{n^2 + m_2^2}$ предпочтительным является r_1. Таким образом, расстояние до ближайшего объектного пиксела на данной стороне квадрата находится по формуле

$$r = \sqrt{n^2 + (min(m_1, m_2))^2}\ . \tag{1}$$

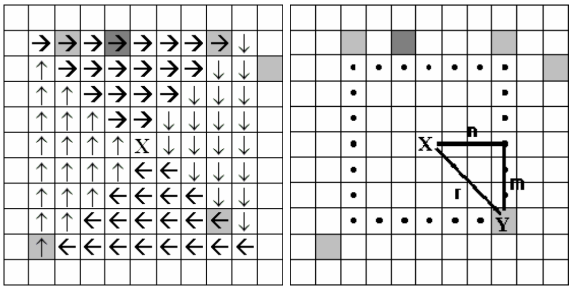


Рис. 1. Поиск объектной точки, ближайшей к фоновой точке X. Слева: сканирование границ квадратов вокруг точки X (стрелками указано направление обхода, серые точки – объектные, темная точка – ближайшая объектная к X); справа: определение расстояния до объектной точки Y (точками указан периметр текущего квадрата, n-номер итерации, m-расстояние от середины стороны квадрата до точки Y, r-расстояние между точками X и Y)

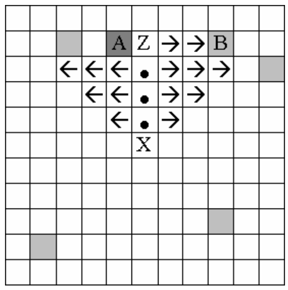


Рис. 2. Сокращенное сканирование верхней стороны квадрата от середины к краю

Выражение $min(m_1, m_2)$ есть расстояние от срединной точки *Z* до ближайшего к нему по горизонтали объектного пиксела *A*.

Предварительный поиск привязки по горизонтали и вертикали

Теперь будем привязывать не один фоновый пиксел к объектному, а все. Применим сокращенный метод поиска ко всей картинке, при этом многие точки неоднократно становятся срединными точками для разных пикселов (рис. 3.).

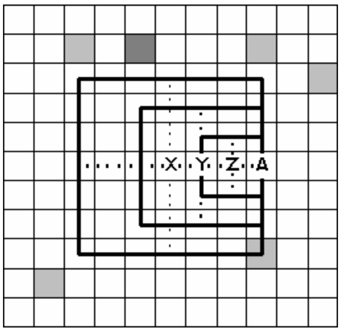


Рис. 3. При сканировании точка A является срединной для точек X,Y,Z

Чтобы пользоваться формулой (1), не прибегая к поиску $min(m_1,m_2)$, каждый раз, когда точка становится срединной, предварительно просканируем картинку по горизонтали и вертикали и найдем для каждого фонового пиксела ближайший объектный в строке и столбце (рис. 4), при этом считаем, что за границами картинки фон (т.е. все пикселы привязываются к внутренним точкам картинки).

0	0	0	←1	←4	←9
0	0	←1	←4	→1	0
0	←1	←4	→4	→1	0
→4	→1	0	←1	0	0
∞	∞	∞	∞	∞	∞

0	0	0	∞	↓9	↓1
0	0	↑1	∞	↓4	0
0	↑1	↓1	∞	↓1	0
↑1	↑4	0	∞	0	0
↑4	↑9	↑1	∞	↑1	↑1

Рис. 4. Предварительная привязка фоновых пикселов к объектным. Числа в клетках равны квадратам расстояний до точек привязки, стрелки указывают направление привязки. Слева: привязка по горизонтали; справа: привязка по вертикали

Рассмотрим подробнее, как происходит сканирование строк (сканирование столбцов происходит аналогично). Объектные пикселы привязываются сами к себе и расстояние привязки для них равно нулю. Найдем непрерывный фрагмент фона в строке. Возможны четыре варианта ограничения фрагмента.

а) Фрагмент ограничен объектными пикселами с двух сторон. Тогда делим его пополам, левую часть привязываем к левому объектному пикселу, правую часть - к правому. Квадраты расстояний в каждом фрагменте рассчитываются рекурсивно без умножений по следующей формуле:

$$x_i^2 = (x_{i-1}+1)^2 = x_{i-1}^2 + 2x_{i-1} + 1,$$

где x_i - расстояние привязки текущей точки, x_{i-1} - расстояние привязки предыдущей точки (левый фрагмент заполняется слева направо, правый – справа налево).

б) Фрагмент слева ограничен объектным пикселом, а справа – границей картинки. Тогда все пикселы фрагмента привязываются к левому объектному пикселу.

в) Фрагмент справа ограничен объектным пикселом, а слева – границей картинки (аналогично пункту б)).

г) Фрагмент справа и слева ограничен границей картинки. Пикселы фрагмента не могут быть к чему-либо привязаны и расстояние привязки для них считается равным бесконечности.

Теперь, когда для каждой точки известно расстояние привязки по горизонтали и вертикали, сканирование стороны квадрата не производится; вместо этого берется квадрат расстояния от середины до точки ее привязки (рис. 5) и прибавляется к квадрату номера итерации (он равен расстоянию от точки *X* до срединной точки). Затем следует сравнение полученного квадрата расстояния с текущим минимумом и т.д.

				←1				
				∞				
				∞				
				∞				
↓4	↑4	∞	↑4	X	∞	∞	↓3	∞
				∞				
				∞				
				→3				
				←4				

Рис. 5. Сканирование середин сторон квадратов. Число – расстояние привязки пиксела, стрелка указывает направление привязки

Использование предыдущих расстояний привязки для оценки текущего расстояния привязки

Посмотрим, как можно использовать расстояние привязки, полученное на предыдущем шаге, для нахождения текущего расстояния привязки. Пусть два соседних фоновых пиксела привязываются к разным объектным пикселам (рис. 6).

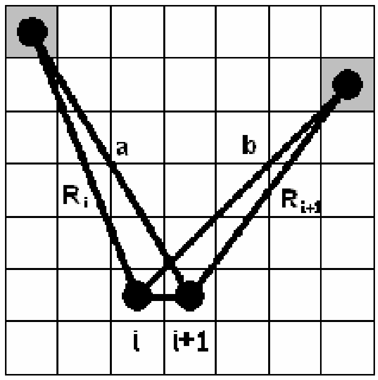


Рис. 6. Привязка двух соседних пикселов (с номерами i и i+1). R_i *- расстояние привязки i-го фонового пиксела,* R_{i+1} *- расстояние привязки i+1 фонового пиксела*

Запишем неравенства:

- $R_i \le b$, $R_{i+1} \le a$, поскольку R_i и R_{i+1} - расстояния до ближайших объектных пикселов для *i*-го и i+1-го пикселов соответственно;
- $b \le R_{i+1} + 1$, $a \le R_i + 1$ - из неравенства треугольников;

следовательно $R_i \le R_{i+1} + 1$, $R_{i+1} \le R_i + 1$ и отсюда $R_i - 1 \le R_{i+1} \le R_i + 1$.

Таким образом, мы получили нижнюю и верхнюю оценку расстояния привязки для i+1-го фонового пиксела. Если i+1-ый фоновый пиксел привязывается к тому же объектному пикселу, что и предыдущий, то можно легко показать, что неравенство будет таким же. Покажем, как можно использовать это неравенство для уменьшения числа сканирований приблизительно в три раза.

Предположим, что мы сканируем середины сторон квадратов для i+1-го фонового пиксела, пусть *n* - номер итерации, на которой найден ближайший объектный пиксел, *m* - расстояние от середины стороны квадрата до ближайшего к ней объектного пиксела, тогда $R_{i+1}^2 = n^2 + m^2$. На рис. 7 показано, что один и тот же объектный пиксел *Z*, ближайший к текущей фоновой точке *A*, может быть найден двумя способами: при сканировании вверх от точки *A* (через точку *B*) и при сканировании вправо от точки *A* (через точку *C*).

Выберем тот способ, при котором выполняется $n \ge m$ (т.е. при котором длина большего из катетов равна номеру итерации). Тогда $\frac{R_{i+1}^2}{2} + \frac{R_{i+1}^2}{2} = n^2 + m^2$, следовательно $\frac{R_{i+1}}{\sqrt{2}} \le n \le R_{i+1}$. Подставим в последнее неравенство оценки R_{i+1}: $\frac{R_i - 1}{\sqrt{2}} \le n \le R_i + 1$ или приблизительно $0.707(R_i - 1) \le n \le R_i + 1$. Таким образом, можно найти расстояние до ближайшего объектного пиксела, проводя итерации только в указанных пределах, что при больших R_i составляет приблизительно 30% исходного числа итераций.

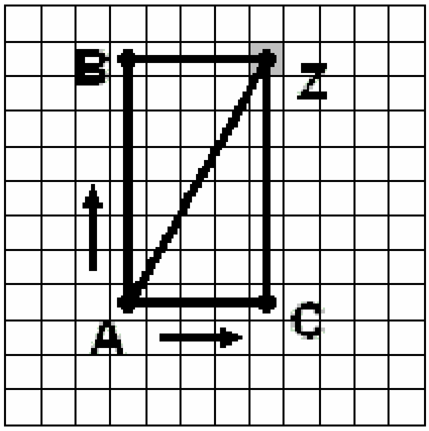


Рис. 7. Точка Z (ближайшая объектная для фоновой точки A) может быть найдена при сканировании двумя способами: через точку B (при сканировании вверх от точки A) и через точку C (при сканировании вправо от точки A)

Сравним этот метод поиска с простейшим алгоритмом по числу операций для конкретного пиксела. Если *R* - расстояние от текущего фонового пиксела до ближайшего объектного, то в простейшем алгоритме для поиска необходимо провести порядка R^2 операций, а в последнем методе порядка $4 \times 0{,}3R$ (сканирование в 4 стороны) плюс дополнительные операции на предварительную привязку по горизонтали и вертикали, вычислительная сложность которой линейно зависит от размеров картинки.

Использование предыдущих расстояний привязки для исключения лишних сканирований

Проанализируем, как еще можно использовать информацию, полученную на предыдущих шагах. Пусть найдены расстояния привязки для двух точек *A* и *B* выше текущей точки *X* и для двух точек *C* и *D* левее точки *X* (рис. 8а).

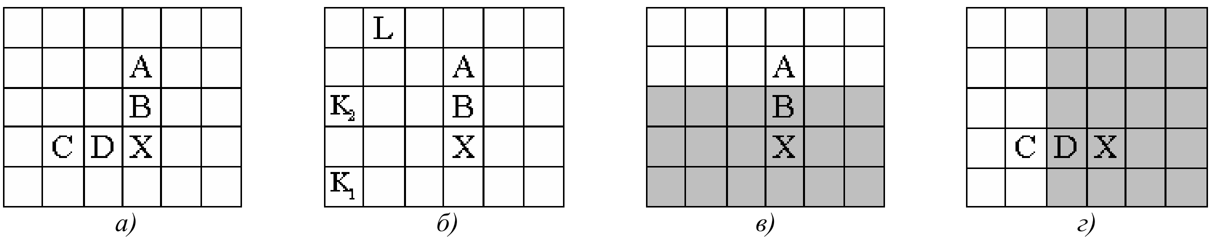


Рис. 8. а) Поиск расстояния привязки для точки X. На предыдущих шагах были найдены расстояния привязки до точек A,B,C и D. б) Иллюстрация к поиску точки привязки для X при $R_A > R_B$. в) Если $R_A > R_B$, то точка привязки для X лежит ниже точки A (в полуплоскости B, отмеченной серым). г) Если $R_C > R_D$, то точка привязки для X лежит правее точки C (в полуплоскости D, отмеченной серым).

Обозначим расстояния привязки для каждой точки соответственно R_A, R_B, R_C, R_D и R_X. Очевидно, что любая объектная точка, лежащая в полуплоскости выше точки B, ближе к точке *A*, чем к точке *B* (обозначим эту полуплоскость *A*). Аналогично, любая объектная точка, лежащая в полуплоскости ниже точки *A*, ближе к точке *B*, чем к точке *A* (обозначим эту полуплоскость *B*). Аналогично, любая объектная точка, лежащая в полуплоскости ниже точки *B*, ближе к точке *X*, чем к точке *B* (обозначим эту полуплоскость X). Предположим, что $R_A > R_B$ (рис. 8.б). Пусть объектная точка K является точкой привязки для B. Она обязана лежать в полуплоскости B, иначе $R_A \le AK < BK = R_B$, что противоречит тому, что $R_A > R_B$. Докажем, что объектная точка, ближайшая к X, лежит в полуплоскости B. Предположим противное. Пусть объектная точка L, принадлежащая полуплоскости A, является ближайшей для X. Рассмотрим два варианта положения точки K.

1. Пусть точка K лежит в полуплоскости X (ниже точки B), тогда $XK<BK$, $BK=R_B \le BL$, $BL<XL$. Следовательно, $XK<XL$, т.е. точка L не является ближайшей объектной точкой для X.
2. Пусть точка K лежит на горизонтальной прямой, проходящей через точку B. Тогда $BL^2 < XL^2 = R_X^2 \le XK^2 = BK^2+1$, т.е. $BL^2 < BK^2+1$. Отметим, что BL^2 и BK^2 – целые числа (поскольку координаты всех точек целые). Это означает, что $BL^2 \le BK^2$, или $BL \le BK$. $R_A \le AL < BL \le BK = R_B$, т.е., $R_A < R_B$, что противоречит исходному условию (что $R_A > R_B$).

Таким образом, доказано, что если выполняется условие $R_A > R_B$, то точка привязки для X лежит в полуплоскости ниже точки A (рис. 8в). Аналогично доказывается, что если выполняется условие $R_C > R_D$, то точка привязки для X лежит в полуплоскости правее точки C (рис. 8г). Используя эту информацию, можно исключить часть сканирований влево и вверх, что ускорит выполнение задачи. Насколько именно, зависит от структуры картинки (рис. 9) .

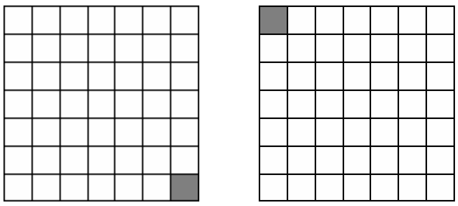


Рис. 9. Слева: точка в нижнем правом углу – для большинства точек не производится сканирование влево и вверх. Справа: точка в левом верхнем углу – для большинства точек производится сканирование во все стороны

Назовем вышеописанный алгоритм со всеми процедурами уменьшения числа операций рекурсивным алгоритмом поиска квадратов расстояний.

Возвратный алгоритм

Сформулируем и исследуем еще один алгоритм. Его отличие от предыдущего заключается в том, что ищется только горизонтальная привязка (рис. 4, слева), сразу после этого начинается поиск квадратов расстояний. Предварительно докажем следующее утверждение.

Пусть в двумерном евклидовом пространстве заданы две точки $A=(x_1,y_1)$ и $B=(x_2,y_2)$, причем $y_1<y_2$, и вертикальная прямая $x=x_0$. На этой прямой лежит точка C с плавающей координатой y (рис. 10).

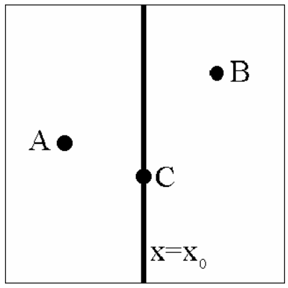


Рис. 10. Две точки и вертикальная прямая

Докажем, что существует некоторая координата $y=y_0$, такая что при $y<y_0$ $AC<BC$, при $y>y_0$ $AC>BC$.

$$AC^2 = \underbrace{(x_1 - x_0)^2}_{\Delta\xi_0} + (y_1 - y)^2 = \Delta\xi_0^2 + y_1^2 - 2y_1y + y^2$$

$$BC^2 = \Delta\xi_0^2 + y_2^2 - 2y_2y + y^2$$

Запишем неравенство:

$$AC^2 < BC^2,$$

$$\Delta\xi_0^2 + y_1^2 - 2y_1y + y^2 < \Delta\xi_0^2 + y_2^2 - 2y_2y + y^2,$$

$$2y(y_2 - y_1) < \Delta\xi_0^2 + y_2^2 - \Delta\xi_0^2 - y_1^2,$$

$$y < \frac{\Delta\xi_0^2 + y_2^2 - \Delta\xi_0^2 - y_1^2}{2(y_2 - y_1)} = y_0.$$

Таким образом, искомая координата y_0 найдена. Опишем поиск квадратов расстояний после того, как найдена горизонтальная привязка (рис. 11а).

Сканируем столбец сверху вниз. Выделим этапы сканирования.

1. Предварительно расстояния привязки в столбце устанавливаются равными расстояниям горизонтальной привязки.
2. Пропускаются точки, для которых горизонтальная привязка не найдена. Если пропущен до конца весь столбец, то переходим к сканированию снизу вверх. Если в столбце нет ни одной горизонтальной привязки, то на исходной картинке не было ни одной объектной точки.
3. Пусть найдена привязка по горизонтали к точке A (рис. 11б). Пробуем привязать к ней все точки в столбце, которые находятся ниже или на уровне ее. Для каждой точки привязка к A осуществляется, только если текущее расстояние привязки больше, чем расстояние до точки A. Процесс продолжается до тех пор, пока не будет найдено расстояние привязки по горизонтали не большее, чем текущее расстояние до точки A (найдена привязка к точке D).
4. Из выше доказанного утверждения следует, что все точки в столбце ниже точки D ближе к D, чем к A, поэтому привязка к A заканчивается. Из того же

утверждения следует, что для точек A и B существует координата y_0, такая, что любая точка в текущем столбце с вертикальной координатой ниже y_0 ближе к точке B, чем к точке A, поэтому необходимо вернуться в строку с точкой B и попробовать найти лучшие расстояния привязки. Для этого возвращаемся на строку, следующую за строкой с точкой A и переходим к пункту 2.

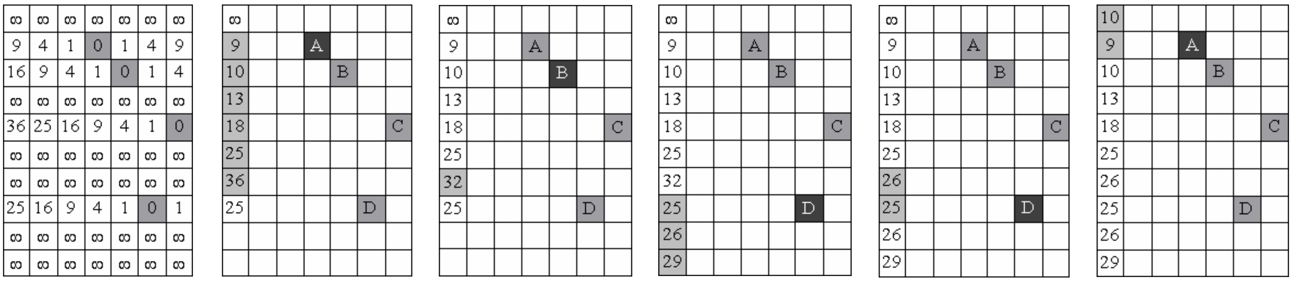


Рис. 11. Пример привязки столбца возвратным алгоритмом. Числа равны квадратам расстояний привязки. Горизонтальная привязка (а), сканирование вниз, привязка к точке A (б), сканирование вниз, привязка к точке B (в), сканирование вниз, привязка к точке D (г), сканирование вверх, привязка к точке D (д), сканирование вверх, привязка к точке A (е).

В пункте 2 не пропускается ни одной строки, затем находится горизонтальная привязка к точке B. Повторяем пункты 2, 3, 4 для точек B (рис. 11в), C (к ней не привязывается ни одной точки в текущем столбце) и D (рис. 11г) до тех пор, пока в пункте 2 не будет достигнут нижний конец столбца. После этого необходимо просканировать столбец снизу вверх, повторяя последовательно пункты 2, 3, 4 (рис. 11д, рис. 11е) затем перейти к следующему столбцу и так далее для всех столбцов.

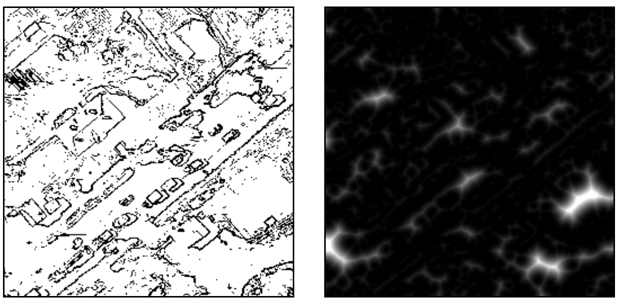
Поскольку в данном алгоритме при сканировании постоянно приходится возвращаться назад, назовем его «возвратным алгоритмом».

Заключение

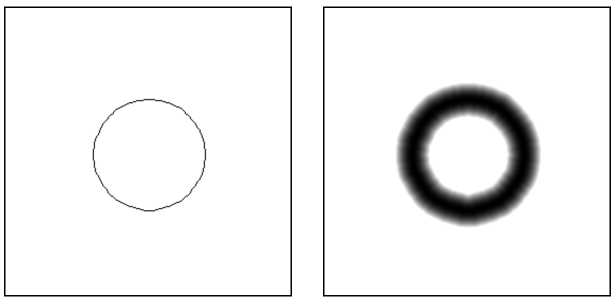
Очевидно, что структурно возвратный алгоритм намного проще рекурсивного, использует в два раза меньше памяти для хранения промежуточных данных (поскольку в рекурсивном алгоритме хранятся квадраты расстояний привязки по горизонтали и вертикали, а в возвратном – только по горизонтали) и не требует вычисления квадратного корня для каждой точки. Вычислительную сложность возвратного алгоритма однозначно оценить невозможно, однако очевидно, что он должен хорошо работать на разреженных картинках (тогда в пункте 2 пропускается много пикселов) и на картинках, где много вертикальных линий (вблизи них в пункте 3 повторных сканирований точек не производится). На рис. 12 приведены результаты сравнения работы алгоритмов по времени выполнения и среднему числу сканирований на точку на машине (PII-400, системная шина 100 МГц). На всех исследованных изображениях возвратный алгоритм показал лучшие результаты, чем рекурсивный.

Литература

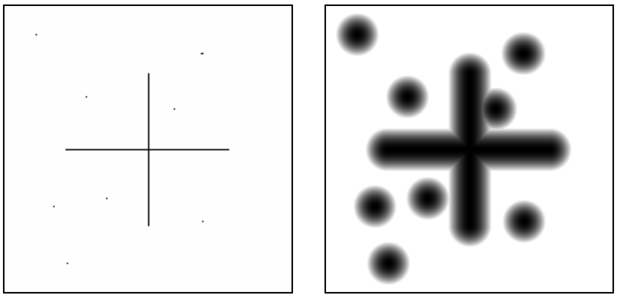
1. Сойфер В.А. Проблемы обработки изображений и компьютерной оптики. Стенограмма научного сообщения на заседании Президиума Российской академии наук 19.10.1999 // Компьютерная оптика, № 19, 1999, с. 6-20.
2. Казанский Н.Л., Хмелев. Р.В. Сравнение объекта и эталона по отклонению контуров // Компьютерная оптика № 20, 2000.



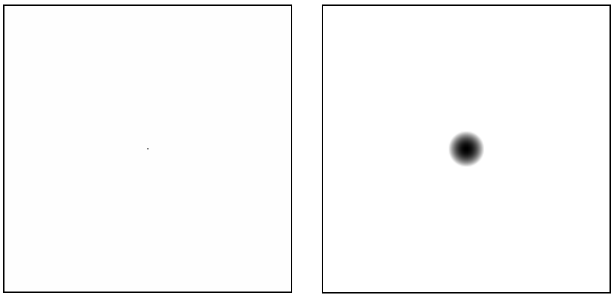
а) Контурное изображение снимка местности (256× 256): $\bar{n}_p$ =10.42 , $\bar{n}_в$ =8.61, $\bar{R}$ =2.66, t_p =0.055, $t_в$ =0.036



б) Круг (256× 256): $\bar{n}_p$ =46.04, $\bar{n}_в$ =21.43, $\bar{R}$ =51.99, t_p =0.114, $t_в$ =0.107



в) Крест и несколько точек вокруг него (200 × 200): $\overline{n}_{р}$ =29.28, $\overline{n}_{в}$ =7.16, $\overline{R}$ =24.05, $t_{р}$ =0.048, $t_{в}$ =0.009



г) Одна точка в центре (255 × 255): $\overline{n}_{р}$ =71.55, $\overline{n}_{в}$ =4.99, $\overline{R}$ =96.57, $t_{р}$ =0.113, $t_{в}$ =0.011

Рис. 12. Примеры поиска квадратов расстояний. Слева: исходное изображение, справа: картинка квадратов расстояний (значения, большие или равные 255 показаны белым цветом). $\overline{n}_{р}$ *– среднее число сканирований на точку в рекурсивном алгоритме,* $\overline{n}_{в}$ *– среднее число сканирований на точку в возвратном алгоритме,* $\overline{R}$ *– среднее расстояние привязки,* $t_{р}$ *– время работы рекурсивного алгоритма,* $t_{в}$ *– время работы возвратного алгоритма.*

Algorithms for computing distances to object pixels on binary images

N.L. Kazanskiy, V.V. Myasnikov, R.V. Khmelev
Image Processing Systems Institute of RAS

Abstract

One of the most important tasks in image processing is the task of performing the measurements on the image [1], in particular, measuring the distances between the image points. Let us assume that there is a binary picture, in this picture the values "0" will be referred to as the background, and the values "1" will be referred to as the object. For each point of the background, it is required to find the distance (in pixels) to the nearest point of the object (we will call it the snap distance), and this shall be done in a minimum number of operations. This problem arises, for example, when an object is compared with the standard in terms of deviation of the contours [2]. This article describes two algorithms for solving this problem.

Citation: Kazanskiy NL, Myasnikov VV, Khmelev RV. Algorithms for computing distances to object pixels on binary images. Computer Optics 2000; 20: 134 - 139.

References

[1] Soifer VA. Problems of digital image processing and computer optics. Transcripts of the academic report made at the meeting of the RAS Presidium on October 19, 1999. Computer Optics 1999; 19: 6-20.
[2] Kazanskiy NL, Khmelev RV. Comparison of an object and a sample for the deviation of the contours // Computer Optics 2000; 20: 134-139.