Известия Гомельского государственного университета
имени Ф. Скорины, №4(25), 2004

УДК 681.3

Алгоритм поиска наилучшей траектории при лазерной гравировке

А. М. Евтухов, В. Д. Левчук

Лазерная установка, схематически изображенная на рис. 1, состоит из следующих компонентов:

1) Сам **лазер** непосредственно. Во время гравировки лазер работает непрерывно, однако путь лучу закрывает специальная шторка. Для того, чтобы луч попал на образец, необходимо открыть шторку.

2) Так как лазер слишком тяжело перемещать и, к тому же при любом перемещении лазера зеркала внутри его могут перекоситься, то вместо того, чтобы перемещать лазер, перемещают сам образец. Для этого его закрепляют на **координатный стол**. Вся задача координатного стола заключается в том, чтобы двигать образец. На рисунке схематично показан один из вариантов стола – своеобразный слоёный пирог из двух гибких лент и двух пластин. Ленты приводятся в движение каждая своим двигателем. В общем случае можно перемещать стол по произвольной траектории и менять скорость движения когда и каким угодно образом.

3) От LPT-порта персонального компьютера отходит шлейф к **блоку управления координатным столом.** Его задача – просто преобразовывать сигнал, поступающий от компьютера. Например, для открытия/закрытия шторки требуется напряжение 12 В, однако напряжение в LPT–порте всего лишь 5 В. Поэтому для шторки блок управления повышает напряжение. Блок управления выполняет ещё ряд посреднических функций, но с точки зрения рассматриваемой задачи это не представляет интереса.

4) Всё управление движением координатного стола и открытием/закрытием шторки осуществляется с **персонального компьютера** (последний компонент). С точки зрения программиста это выглядит следующим образом. Для того, чтобы стол сдвинулся на 1 шаг (например, 0.025 мм) параллельно, допустим, оси X в положительном направлении, передается определённый код через LPT-порт. Чем чаще передается этот код, тем чаще поворачивается на 1 шаг двигатель, и тем быстрее движется стол. Обычно при гравировке частота порядка 100-1000 шагов в секунду. Для глаза это сливается в непрерывное движение.

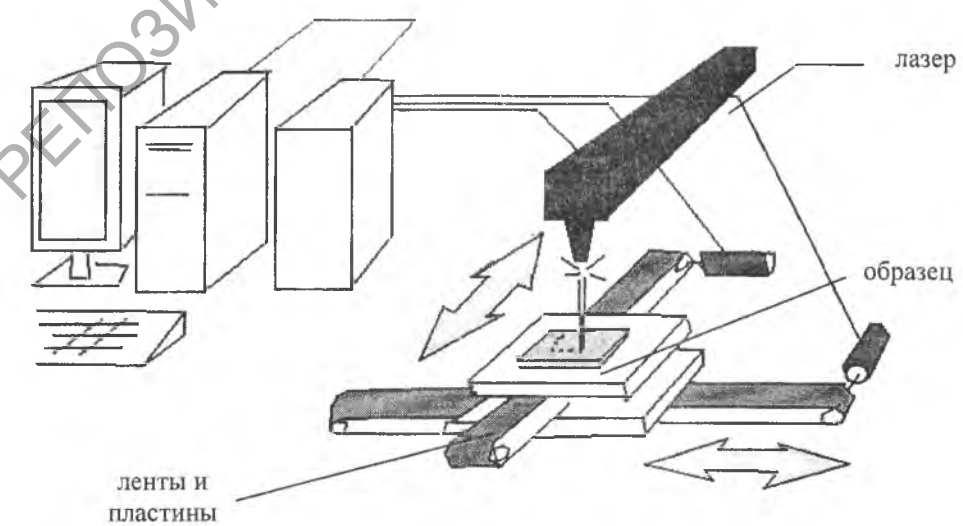


Рис. 1. Схема лазерной установки

Рисунок, подготовленный для гравировки, представляет собой набор графических примитивов, которые могут быть: точкой, незамкнутой кривой или замкнутой кривой. Для

уменьшения затрат времени на гравировку лазер перемещается от примитива к примитиву по прямой, однако при этом возникает вопрос: в каком порядке соединять примитивы между собой. Будем считать, что лазер в выключенном состоянии движется с постоянной скоростью – в таком случае для сокращения общих затрат времени на гравировку одного рисунка нам нужно уменьшить суммарную длину соединительных линий. Если в рисунке нет пересечений, то количество соединительных линий равно N – 1, где N – количество всех примитивов на рисунке (точек, незамкнутых и замкнутых кривых). Назовём точку, соединяющую примитив с соединительной линией, контактной точкой. Если мы находим координаты контактной точки одной соединительной линии, принадлежащей примитиву, то координаты контактной точки второй линии находятся автоматически. N примитивов можно расположить друг за другом N! способом, и нам нужно найти именно тот порядок обхода всех примитивов, при котором достигается наименьшее время.

Рассмотрим пока в качестве примитивов только точки. Проименуем их малыми буквами латинского алфавита. Последовательности точек (упорядоченные множества точек) будем обозначать L, например последовательность L = amnb (рис. 2). Неупорядоченные множества точек будем обозначать Q, например Q = {a, m, n, b, p}.

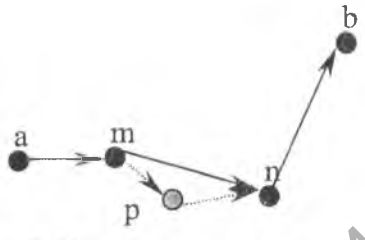


Рис. 2. Последовательность точек

При составлении программы на гравировку мы можем перебирать все примитивы рисунка только по одному, поэтому можно предположить, что алгоритм нахождения оптимальной траектории должен основываться на принципе последовательного внесения корректировок с каждым новым примитивом в начальную траекторию. Точным решением данной задачи было бы нахождение такого бинарного оператора от двух последовательностей, что результатом его действия станет последовательность с минимальной длиной соединительных линий, включающая в себя все точки обоих операндов. Очевидной реализацией такого оператора является простой метод перебора всех возможных перестановок всех точек на рисунке. Так как количество таких перестановок растёт как факториальная зависимость от числа точек, то это делает практическую реализацию точного оператора для большого количества точек фактически неосуществимой, и мы, поэтому, будем искать приближённый оператор. Введём следующие обозначения:

Таблица 1.

Обозначения, характерные для этапа 1.

Обозначение	Описание
$\|L\|$	Общая длина соединительных линий последовательности L
L^N	Последовательность, состоящая из N точек
$\|Q\|$	Количество точек во множестве Q
Perm(Q)	Результатом действия оператора Perm (*"permutation" – перестановка*) на неупорядоченное множество Q является множество всех упорядоченных последовательностей, которые можно получить, переставляя между собой точки последовательности Q
$\begin{bmatrix} L_1 \\ I_1 \end{bmatrix}\begin{bmatrix} L_2 \\ I_2 \end{bmatrix}$	Полное смешение L_1 и L_2 (необходимое лексическое пояснение: *смешивание* – это действие, а *смешение* – результат этого действия; эти термины даются по аналогии с принятыми в оптике *рассеиванием* (действием) и *рассеянием* (результатом этого действия)). Общая форма записи для создания новой последовательности на основе последовательностей L_1 и L_2. I_1 и I_2 представляют собой упо-

Обозначение	Описание
	рядоченные множества индексов, причём $\|L_1\| = \|I_1\|$, $\|L_2\| = \|I_2\|$. i-ый элемент кортежа I_s указывает номер позиции в результирующей последовательности, которую займёт i-ый элемент последовательности L_s. **Пример.** $\begin{bmatrix} a & b & c & d \\ 5 & 3 & 2 & 6 \end{bmatrix}\begin{bmatrix} m & n \\ 1 & 4 \end{bmatrix} = mcbnad$
$L_1\begin{bmatrix} L_2 \\ I_2 \end{bmatrix}$	Частичное смешение L_1 и L_2 (в данном случае – правое смешение). В случае, когда в результирующей последовательности L элементы L_1 сохраняют первоначальный порядок (т.е., при $\begin{bmatrix} L_1 \\ I_1 \end{bmatrix}\begin{bmatrix} L_2 \\ I_2 \end{bmatrix}$, $I_1 = (i_1, i_2, ..., i_N)$, $i_1 < i_2 < \ldots < i_N$), для определения результирующей последовательности нам достаточно знать индексы последовательности L_2. Аналогичным образом определяется левое смешение $\begin{bmatrix} L_1 \\ I_1 \end{bmatrix}L_2 = \begin{bmatrix} L_1 \\ I_1 \end{bmatrix}\begin{bmatrix} L_2 \\ I_2 \end{bmatrix}$, $I_2 = (i_1, i_2, ..., i_N)$, $i_1 < i_2 < \ldots < i_N$
$[L_1][L_2]$	Сокращённая форма записи полного смешения. Она применяется в случаях, когда нас не интересуют подробности относительно кортежей индексов обоих последовательностей. Аналогичным образом определяется сокращённая форма записи для частичного правого смешения, как $L_1[I_2]$
L_{opt}	Последовательность с наименьшей длиной соединительных линий
$[L_1][L_2]_{opt}$	Смешение $[L_1][L_2]$ с такими кортежами индексов, что справедливо тождество $L_{opt} \equiv [L_1][L_2]_{opt}$
$L_1 \circ L_2$	Явная запись конкатенации. **Замечание.** Любое смешение можно представить в виде перестановки конкатенации: $[L_1][L_2] \equiv [L_1 \circ L_2]$
$\hat{A}$	Оператор, результатом действия которого является последовательность L_{opt}. Пусть у нас задано множество точек $p_i \in Q^N, i = \overline{1,N}$, тогда справедливо тождество $L^N_{opt} \equiv \hat{\underset{i=1}{A}}^{N} p_i, L^N_{opt} \in Perm(Q^N)$. Оператор определяется по рекурсии: 1) $\hat{\underset{i=1}{A}}^{1} p_i \equiv p_1$; 2) если обозначить через $L^k = \hat{\underset{i=1}{A}}^{k} p_i$, то $\hat{\underset{i=1}{A}}^{k+1} p_i \equiv [L^k][p_{k+1}]_{opt}$ Мы также будем использовать $\hat{A}$ в качестве бинарного оператора: $L^{N+1}_{opt} \equiv \hat{A}(L^N, p)$

Таким образом, рассматриваемая задача записывается следующим образом: для заданного множества Q точек рисунка найти последовательность $L_{opt} \in Perm(Q)$.

Будем называть выражение вида $L[p]$ – вставкой, выражение $[L]\circ p$ – смешанной конкатенацией, а Lp – простой конкатенацией. В общем случае воздействие оператора $\hat{A}$ на его операнды представляет собой полное смешение, однако мы сперва рассмотрим более простые формы оператора $\hat{A}$ – формы вставки и конкатенации.

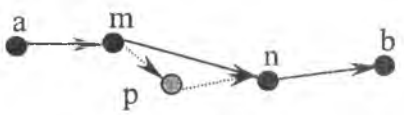
Вставка и простая конкатенация (рис. 3). Предположим, что специфика базовой последовательности такова, что после добавления новой точки порядок следования базовых точек не меняется, т.е. мы имеем дело со вставкой $L[p]$. Определим, в каких случаях вставка вырождается в простую конкатенацию. Обозначим базовую последовательность L^N (мы не

предполагаем, что базовая последовательность является оптимальной). Последовательность, которую мы получим в результате смешения, обозначим L_{opt}^{N+1} (однако, нужно учитывать, что эта последовательность оптимальна только с учётом запрета на перестановку между собой элементов последовательности L^N). Пусть L^N имеет вид L^N = a…mn…b. У нас есть два варианта действий:

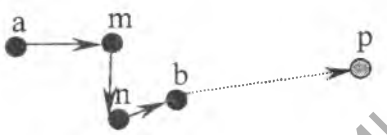
1) Найти такое звено mn внутри L^N, что его замена на звенья mp и pn приведёт нас к L_{opt}^{N+1}. При этом $\left\|L_{opt}^{N+1}\right\| = \left\|L^N\right\| + \|mp\| + \|pn\| - \|mn\|$.

2) Просто дописать в конец последовательности точку p. В этом случае $\left\|L_{opt}^{N+1}\right\| = \left\|L^N\right\| + \|bp\|$.

а) вставка



б) простая конкатенация



p – добавляемая точка; сплошная стрелка – исходная траектория; пунктирная стрелка – новое соединение.
а) При вставке одно из соединений внутри структуры заменяется на два.
б) При простой конкатенации к существующей траектории добавляется ещё одно соединение.

Рис. 3. Геометрическая интерпретация вставки и простой конкатенации

Мною был разработан алгоритм ***InsertPoint***, который находит наилучшее звено для вставки с учётом запрета на перестановку (рис. 4). Затраты времени прямо пропорциональны количеству точек в последовательности L.

```
/* код алгоритма InsertPoint */
struct point{double x, y; /* координаты точки */};
void InsertPoint(
    list<point> L, /* исходная последовательность длины N */
    point p /* добавляемая точка */) {
    int i; /* индекс цикла */
    point *prev, /* предыдущая точка последовательности */
      *curr, /* текущая точка последовательности*/
      *bestprev; /* по этой точке мы определяем, куда будем вставлять
точку p */
    double prevdist, currdist;
    double delta, mindelta;
    /* Функция вычисления расстояния между точками p1 и p2 */
    double Distance(point p1, p2)
    {return sqrt(sqr(p2.x - p1.x) + sqr(p2.y - p1.y));}
    /* Если в последовательности нет точек, то просто добавляем в неё
точку p */
    if(L.Size() == 0) {L.Add(p); return;}
/* Первую точку в последовательности нельзя переместить, поэтому позиция
точки p
    в этом случае определяется однозначно */
    if(L.Size() == 1) {L.InsertAfter(L.Start(), p); return;}
    prev = L.Start(); curr = L.Next(prev);
    prevdist = Distance(*prev, p);
    for(i = 1; i < L.Size(); i++) {
        currdist = Distance(*curr, p);
    /* Вычисляем, насколько изменится длина траектории после вставки точки
p */
        delta = prevdist + currdist - Distance(*prev, *curr);
        if(i == 1) mindelta = delta;
        if(delta <= mindelta) {bestprev = prev; mindelta = delta;}
        prevdist = currdist;
        prev = curr; curr = L.Next(curr);
    }
    /* Изменение длины траектории при простой конкатенации рассчитывается
```

```
особым образом */
          delta = prevdist;
          if(delta <= mindelta) {bestprev = prev;}
          L.InsertAfter(prev, p);
      }
```

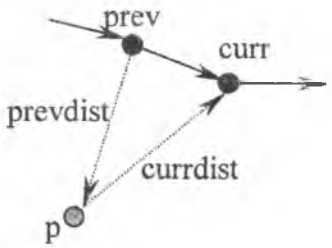
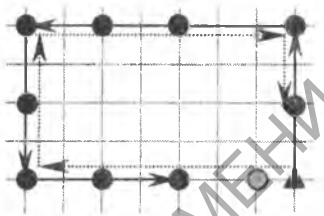


Рис. 4. Пояснительная схема к алгоритму ***InsertPoint***

Полное **смешение** ($[L]p$) (рис. 5). При этом виде смешения, в отличие от простой конкатенации и вставки, нарушается относительный порядок точек последовательности. Поскольку $[L^N]p \equiv [L^N \circ p]$, то у нас будет $(N+1)!$ способ образования новой последовательности методом полного смешения. С вычислительной точки зрения перебрать все эти варианты попросту нереально, поскольку в обычной гравюре содержится не менее нескольких тысяч точек.



Треугольник–начальная точка, серый кружок –добавляемая точка, сплошная стрелка – траектория обхода до добавления новой точки, пунктирная стрелка – траектория обхода после добавления.
После добавления новой точки относительный порядок точек исходной последовательности полностью меняется.

Рис. 5. Геометрическая интерпретация полного смешения

Обратим внимание на следующий интересный факт: если из последовательности L^N удалить одну точку (получив при этом последовательность L^{N-1}), а затем снова вставить эту точку в L^{N-1} по алгоритму ***InsertPoint***, получив при этом новую последовательность M^N, то $\|M^N\| \le \|L^N\|$. В самом деле, удаление и вставка точки – взаимообратные операции, а поскольку в ***InsertPoint*** мы перебираем все звенья L^{N-1}, в том числе и то, которое образовалось после удаления точки p из последовательности, то в самом худшем случае алгоритм попросту вернёт точку p на то место, из которого мы её удалили.

Таким образом, можно предложить следующий приближённый алгоритм ***MinTrack*** поиска кратчайшей траектории обхода точек:

1. Произвольным образом формируем последовательность L^N, содержащую все точки рисунка.
2. Заранее заданное число раз по какому-то эвристическому правилу отыскиваем наиболее худшие смежные звенья, удаляем соединяющую их точку, и заново вставляем её в последовательность по алгоритму ***InsertPoint***.
3. Конец алгоритма.

Abstract

The authors consider an approximate algorithm of the shortest track search at a laser engraving.

Гомельский государственный
университет им. Ф.Скорины

Поступило 20.03.04