АВТОМАТИЧЕСКОЕ ОБНАРУЖЕНИЕ ГЕОМЕТРИЧЕСКИХ ОШИБОК НА МАШИНОСТРОИТЕЛЬНЫХ 2D-ЧЕРТЕЖАХ

Д. Васин[1], С. Ротков[2]

[1] НИИ прикладной математики и кибернетики
Нижегородского государственного университета им. Н.И. Лобачевского
[2] Нижегородский государственный архитектурно-строительный университет

Аннотация

В статье рассмотрены алгоритмы автоматического обнаружения наиболее распространенных геометрических ошибок на машиностроительных 2D-чертежах.

Современные системы САПР широко представлены средствами, обеспечивающими моделирование объектов различной сложности в разных отраслях деятельности. Эти программные комплексы упрощают работу проектировщиков и систематизируют процесс создания цифровой модели изделия. Но при всей своей функциональности современное программное обеспечение САПР не учитывает один немаловажный факт. В этих системах предполагается, что разработчик должен быть квалифицированным и не допускать ошибки построения чертежей. Фактически же системой может пользоваться практически любой пользователь, не имеющий представления о требованиях ГОСТов к чертежам. С другой стороны, даже профессиональные разработчики допускают ошибки. Поэтому достаточно актуальной представляется задача автоматизации контроля метрической информации на машиностроительных чертежах.

В процессе автоматизированного проектирования оперируют геометрическими объектами (ГО), которые являются, как промежуточными, так и окончательными результатами проектирования. ГО характеризуются параметрами, определяющими их форму, а также многими другими сведениями: материалом, чистотой поверхности, термообработкой, допускаемыми отклонениями на размеры и т.д. Большинство этих сведений обычно бывает задано в алфавитно-цифровой форме, и поэтому не требует сложной переработки для представления в ЭВМ. Что касается формы ГО, то ее представление значительно сложнее. В дальнейшем нас будет интересовать только эта часть информации о ГО. Поэтому под моделью ГО мы понимаем совокупность сведений, однозначно определяющих его форму, то есть принципиально должна существовать возможность установить на основании сведений о ГО для каждой точки пространства, принадлежит она этому объекту или нет. Форма ГО однозначно определяется его чертежом. В модели ГО все сведения о нем должны быть представлены только в алфавитно-цифровой форме (в виде уравнений, таблиц данных, текстовых описаний, построенных по определенным правилам).

Описание структуры хранения векторных данных. Для хранения векторных данных и выполнения над ними различных операций предлагается следующий набор геометрических примитивов (ГП):

Примитив Point – точка: Point (int x, int y), где x, y – координаты в декартовой системе координат, связанной с чертежом.

Примитив Line – отрезок: Line (Point t1, Point t2, long Color, int Width, int View), где t1, t2 – координаты начала и конца отрезка; Color – цвет; Width – толщина; View – вид (пунктирная, сплошная и т.д.).

Примитив Polyline – ломаная линия: PolyLine (int N, Point point [N]), где N – число точек ломаной; point [N] – координаты узлов ломаной линии.

Примитив Circle – окружность: Circle (Point C, int R), где C – координаты центра окружности; R – радиус окружности.

Примитив Arc – дуга окружности: Arc (Point C, int R, float Angle1, float Angle2), где Point C – координаты центра окружности, которой принадлежит дуга; R – радиус дуги; Angle1, Angle2 – начальный и конечный углы дуги.

В настоящее время выделено три вида ошибочных ситуаций, наиболее распространенных при создании чертежа.

1. Ситуации совпадения / наложения

1.1. Ошибочное наложение отрезков.
1.2. Ошибочное наложение ломаных линий.
1.3. Ошибочное наложение дуг.
1.4. Ошибочное наложение окружностей.
1.5. Ошибочное наложение ломаной линии и отрезка.
1.6. Ошибочное наложение окружности и дуги.
1.7. Ошибочное наложение / переплетение контуров.

2. Ситуации угловых ошибок

2.1. Нарушение ортогональности отрезков / ломаных линий.
2.2. Нарушение параллельности отрезков / ломаных линий.

3. Ситуации нарушения топологии элементов чертежа

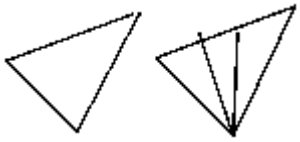
3.1. Нарушение топологии линейных ГО: отсутствие необходимого примыкания (пересечения) линейных элементов / ошибочное примыкание (пересечение) линейных элементов.
3.2. Нарушения топологии окружности и отрезков прямых.
3.3. Разрывы в контуре.

Ошибки 1-го вида возникают для всех представленных ГП. В процессе автоматического контроля необходимо учитывать, что наложение друг на друга одинаковых примитивов не всегда является ошибочным, а наложение различных ГП в большинстве случаев будет являться ошибкой, при этом для линейных ГП в зависимости от типа линии устанавливается факт допустимости наложения.

Ошибки 2-го вида возникают только для ГП Line и ГП Polyline. Они возникают не только между одинаковыми примитивами, но и между различными примитивами. Наиболее часто наблюдаются небольшие отклонения от состояний ортогональности или параллельности между двумя отрезками, принадлежащих разным линиям.

Ошибки 3-го вида возникают для всех ГП. Наиболее распространены ошибки отсутствия необходимой общей точки между различными ГП либо, наоборот, наличие недопустимой общей точки.

Существует вероятность неточного совмещения концевых точек линий. Линии должны образовывать замкнутый контур, то есть начальная и конечная точки, определяющие линию, обязательно должны накладываться на начальные и конечные точки другой линии или же должны накладываться на другую линию (принадлежать ей).



Рассмотрим алгоритмы обнаружения указанных ошибочных метрических состояний.

Алгоритм ProcessLineLine – находит и фиксирует ошибки 1.1, 2.1, 2.2, 3.1, возникающие с ГП «отрезок». Входные параметры алгоритма: Line1, Line2 – анализируемые ГП; tolerance – допустимая погрешность в градусах, для ортогональности \ параллельности ГП; tolerance_lenght – допустимое расстояние, для выявления ошибок типа 3.1; tolerance_lenght1 – допустимое расстояние для фильтрации независимых отрезков. Выходные параметры алгоритма: все анализируемые недопустимые состояния геометрии ГП помечаются спецсимволом для дальнейшего визуального анализа и исправления.

Задаем массив length [4] и заполняем его взаимными декартовыми расстояниями от концов одного отрезка до концов другого. Определяем минимальное (MnLength) из найденных расстояний и, сравнивая его с tolerance_length1, исключаем отрезки, лежащие далеко друг от друга. Ищем среди length[i] максимальное – MxLength. Определяем длины L_1 и L_2 обоих отрезков. Рассчитываем углы наклона первого и второго отрезков к оси Ox, и по разности этих углов находим угол между отрезками U.

Если (90 – tolerance < U < 90 + tolerance) $\vee$ (270 – tolerance < U < 270 + tolerance), то установлен факт ошибки ортогональности анализируемых отрезков; если (0 – tolerance < U < 0 + tolerance) $\vee$ (180 – tolerance < U < 180 + tolerance), то установлен факт ошибки параллельности анализируемых отрезков.

Если анализируемые отрезки лежат на одной прямой и при этом:

если (L1 = MxLength $\wedge$ L2 = MxLength), то установлен факт полного наложения отрезков, без учета их направления;

если (U = 0 $\wedge$ концы обоих отрезков совпадают покоординатно), то установлен факт полного наложения сонаправленных отрезков;

если (U = 180 $\wedge$ начало одного отрезка является концом другого), то установлен факт полного наложения противоположно направленных отрезков.

Определяем взаимное расположение отрезков, лежащих на одной прямой:

если (MxLength < $L_1 + L_2$ – tolerance), то установлен факт наложения отрезков;

если (MxLength < $(L_1 + L_2)$ $\wedge$ MxLength $\geq$ $(L_1 + L_2)$ – tolerance), то установлен факт наложения отрезков, возможна ошибка типа ошибочное примыкание (пересечение) линейных элементов**;**

если (MxLength = $(L_1 + L_2)$), то отрезки имеют общую вершину;

если (MxLength $\leq (L_1 + L_2 +$ tolerance) $\wedge$ MxLength > $(L_1 + L_2)$, то отрезки независимы, возможна ошибка отсутствия необходимого примыкания (пересечения) линейных элементов;

если (MxLength > $(L_1 + L_2 +$ tolerance), то установлен факт независимости анализируемых отрезков.

В случае если анализируемые отрезки не лежат на одной прямой, находим t_ins – точку пересечения прямых, которым принадлежат отрезки, и вычисляем Q1_min, Q1_max – наименьшее и наибольшее расстояния от концов первого отрезка до t_ins, Q2_min, Q2_max – наименьшее и наибольшее расстояния от вершин второго отрезка до t_ins. Тогда:

если ((Q1_min+Q1_max) = 0), то флаг принадлежности Flag1 = истина, т.е. t_ins $\in$ Line1;

если ((Q2_min+Q2_max) = 0), то флаг принадлежности Flag2 = истина, т.е. t_ins $\in$ Line2;

если (Flag1= истина $\wedge$ Flag2 = истина), то t_ins $\in$ Line1 $\wedge$ t_ins $\in$ Line2.

Далее возможны 4 варианта ошибок:

если (Flag1 = истина $\wedge$ Flag2 = истина $\wedge$ (Q1_min < tolerance_lenght $\vee$ Q2_min < tolerance_lenght $\vee$ MnLength < tolerance_lenght)), то возможно ошибочное примыкание (пересечение) линейных элементов;

если (Flag1= истина $\wedge$ Flag2 = ложь $\wedge$ (Q2_min < tolerance_lenght $\vee$ MnLength < tolerance_lenght)), то возможно отсутствие необходимого примыкания (пересечения) линейных элементов;

если (Flag1= ложь $\wedge$ Flag2 = истина $\wedge$ (Q1_min < tolerance_lenght $\vee$ MnLength < tolerance_lenght)), то возможно отсутствие необходимого примыкания (пересечения) линейных элементов;

если (Flag1 = ложь $\wedge$ Flag2 = ложь $\wedge$ (Q1_min < tolerance_lenght $\vee$ Q2_min < tolerance_lenght $\vee$ MnLength < tolerance_lenght)), то возможно отсутствие необходимого примыкания (пересечения) линейных элементов**.**

Алгоритм ProcessCircleLine находит и фиксирует ошибку 3.2, возникающую между ГП «отрезок» и ГП «окружность». Входные параметры алгоритма: Line, Circle – анализируемые структурные элементы; tolerance – допустимая погрешность определения точки касания отрезка и окружности. Выходные параметры алгоритма: все анализируемые недопустимые состояния геометрии ГП помечаются спецсимволом для дальнейшего визуального анализа и исправления.

Будем считать, что Point C – координаты центра ГП «окружность», R – ее радиус, а Point t1, t2 – координаты ГП «отрезок» на плоскости. Тогда уравнение прямой, проходящей через точки t1 и t2, имеет вид A*x –B*y + C = 0, где A = (t1.y – t2.y), B = (t1.x – t2.x), C = (t1.x*t2.y – t2.x*t1.y). Вектор с координатами **N**(A,B) – нормальный, тогда уравнение прямой, проходящей через центр окружности C и перпендикулярной к прямой, проходящей через ГП «отрезок»: $\frac{y - C.y}{t1.y - t2.y} = \frac{x - C.x}{t2.x - t1.x}$.

Решая систему уравнений:

$$\begin{cases} (t1.y - t2.y) * x - (t1.x - t2.x) * y + (t1.x * t2.y - t2.x * t1.y) = 0 \\ \frac{y - C.y}{t1.y - t2.y} = \frac{x - C.x}{t2.x - t1.x}, \end{cases}$$

находим точку t_ins пересечения прямой и перпендикуляра к ней, проходящего через центр окружности C.
Ищем расстояние от точки пересечения до центра окружности: $D = \sqrt{(C.x - t\_ins.x)^2 + (C.y - t\_ins.y)^2}$

Тогда, если ((D – R > 0 ∨ D – R < 0) ∧ (D – R ≤ tolerance)), то установлен факт ошибки.

Если t_ins не принадлежит отрезку, то определяем расстояния от центра окружности до вершин отрезка:

$$D1 = \sqrt{(C.x - t1.x)^2 + (C.y - t1.y)^2};$$

$$D2 = \sqrt{(C.x - t2.x)^2 + (C.y - t2.y)^2}.$$

Если (D1 < D < D2) ∨ (D2 < D < D1), то установлен факт пересечения отрезка и окружности.

Алгоритм ProcessCircleCircle находит и фиксирует ошибку 1.4, возникающую между ГП «окружность». Входные параметры алгоритма: Circle1, Circle2 – анализируемые ГП «окружность», Tolerance_center – допустимое расстояние между центрами окружностей Circle1 и Circle2, Tolerance_radius – допустимое значение, определяющее разницу радиусов окружностей Circle1 и Circle2. Выходные параметры алгоритма: все анализируемые недопустимые состояния геометрии ГП помечаются спецсимволом для дальнейшего визуального анализа и исправления.

Если (Circle1.C = Circle2.C ∧ $|\text{Circle1.R - Circle2.R}| < Tolerance\_radius$), то установлен факт концентрического вложения окружностей.

Если (Circle1.C = Circle2.C ∧ Circle1.R = Circle2.R), то установлен факт полного совпадения окружностей.

$D = \sqrt{(Circle2.C.x - Circle1.C.x)^2 + (Circle2.C.y - Circle1.C.y)^2}$ – декартово расстояние между центрами окружностей.

Если (Circle1.C ≠ Circle2.C ∧ D < Tolerance_center), то установлен факт подозрительной близости центров окружностей.

Алгоритм ProcessArcArc – находит и фиксирует ошибки 1.3, возникающие с ГП «дуга», и является модификацией алгоритма ProcessCircleCircle. Входные параметры алгоритма: Arc1, Arc2 – анализируемые ГП «дуга»; Tolerance_center – допустимое расстояние между центрами дуг; Tolerance_radius – допустимое значение, определяющее разницу радиусов дуг. Выходные параметры алгоритма: все анализируемые недопустимые состояния геометрии ГП помечаются спецсимволом для дальнейшего визуального анализа и исправления. Положим, что Arc1.Angle1 ≤ Arc1.Angle2, а Arc2.Angle1 ≤ Arc2.Angle2

Если (Arc1.C = Arc2.C ∧ $|\text{Arc1.R - Arc2.R}| < Tolerance\_radius$), то установлен факт возможной ошибки наложения окружностей.

Если (Arc1.R = Arc2.R ∧
если ((Arc2.Angle1 ≤ Arc1.Angle1 ≤ Arc2.Angle2) ∨ (Arc2.Angle1 ≤ Arc1.Angle2≤ Arc2.Angle2) ∨ (Arc1.Angle1≤Arc2.Angle1≤Arc1.Angle2) ∨ (Arc1.Angle1≤Arc2.Angle2 ≤ Arc1.Angle2))), то установлен факт возможной ошибки наложения дуг.

Пусть D – расстояние между центрами окружностей, тогда, если (Arc1.R ≠ Arc2.R ∧ D < Tolerance_center), то установлен факт подозрительной близости центров окружностей.

Алгоритм ProcessPolyline – находит и фиксирует ошибки типа 1.2, 1.7, возникающие с ГП «ломаная линия». Входные параметры алгоритма: Polyline1, Polyline2 – анализируемые ГП «ломаная линия». Выходные параметры алгоритма: Alone – счетчик наложений точек, все анализируемые недопустимые состояния геометрии ГП помечаются спецсимволом для дальнейшего визуального анализа и исправления.

Осуществляя полный перебор точек обеих ломаных, выполняем:

если (Polyline1.point[i] = Polyline2.point[j]), то установлен факт совпадении точек ломаных,

если (Polyline1.point[i-1] = Polyline2.point[j-1] ∨
Polyline1.point[i-1] = Polyline2.point[j+1] ∨
Polyline1.point[i+1]=Polyline2.point[j-1] ∨
Polyline1.point[i+1] = Polyline2.point[j+1]), то установлен факт полного совпадении обоих ГП.

Иначе увеличение счетчика Alone.

Практическая проверка указанных алгоритмов на реальных машиностроительных чертежах выявила, что при соответствующем выборе входных параметров алгоритмами выявляется от 98 до 100% ошибочных состояний.