Gleichmäßige Flächenaufteilung von Polygonen

Sebastian Loder Jan Steffen Jendrny

Januar 2022

Gummibärchen (auch Gummibären bzw. Goldbären) sind Fruchtgummis in Form von etwa zwei Zentimeter großen, stilisierten Bären. Sie werden in unterschiedlichen Farben hergestellt und bestehen im Wesentlichen aus Zucker, Zuckersirup und einer erstarrten Gelatine-Mischung, die ihnen ihre gummiartige Konsistenz verleiht.

Stichwörter: Süßigkeiten, Gelatine, Bär, Lebensmittelfarbe.

1 Einleitung

Die Arbeit basiert auf der Veröffentlichung „Polygon Area Decomposition for Multiple-Robot Workspace Division“ von Susan Hert und Vladimir Lumelsky. In dieser Veröffentlichung wird ein konkretes Problem der Polygonzerlegung, das sogenannte *Problem der verankerten Flächenaufteilung* (eng. *anchored area partition problem*) beschrieben und mittels *sweepline*- und *divide-and-conquer*-Techniken gelöst. Die Lösung erfolgt zunächst für konvexe Polygone und wird anschließend auf nicht konvexe, nicht einfache Polygone erweitert.

Die Polygonzerlegung ist eines der zentralen Probleme in der algorithmischen Geometrie und hat viele Anwendungsfälle, wie beispielsweise in der Kartographie, Bildverarbeitung oder in der Computergrafik. In vielen Fällen wird die Polygonzerlegung benötigt, um aus einem beliebigen Polygon eine Menge aus Teilpolygonen mit bestimmten Eigenschaften zu berechnen. Als Beispiel einer vielfach verwendeten Polygonzerlegung kann die Triangulation genannt werden, bei welcher ein gegebenes Polygon in eine Menge von Dreiecken zerlegt wird. Für die so berechnete Menge von Dreiecken stehen dann effiziente Algorithmen zur Lösung von Problemen zur Verfügung. Anschließend können die Lösungen der Teilpolygone zu einer Lösung für das Ausgangspolygon zusammengefasst werden.

Bei dem hier vorgestellten Problem der verankerten Flächenaufteilung ist die Anforderung an die resultierenden Teilpolygone nicht durch eine bestimmte Geometrie (z. B. ein Dreieck), sondern durch die Lage und Fläche der Teilpolygone gegeben. Bezüglich der Lage besteht die Anforderung darin, dass ein gegebener Punkt (Standort genannt) auf dem resultierenden Polygon liegen muss. Jeder Standort weist als Eigenschaft eine Flächenanforderung auf, welche durch die Größe des Teilpolygons erfüllt werden soll. Die Flächenanforderung kann je Standort den gleichen Wert aufweisen, kann aber auch unter den Standorten variieren. Somit bezieht sich das hier beschriebene Problem sowohl auf eine gleichmäßige, als auch eine ungleichmäßige Flächenaufteilung. Das beschriebene Problem ist unter anderem durch die Flächenerkundung von Robotern motiviert:

Auf einem Polygon werden n Roboter auf Standorten S_i, $i = 1, ..., n$ positioniert, welche die Aufgabe erhalten, zusammen die gesamte Fläche des Polygons zu erkunden. Hierzu muss jede Position innerhalb des Polygons von einem der n Roboter abgefahren werden. Um die Arbeit unter den Robotern aufzuteilen, ist es sinnvoll, jedem Roboter einen Polygonteil zuzuweisen, der jeweils zu bearbeiten ist. Die Teilpolygone sollen sich nicht überlappen, um ein mehrfaches Überfahren zu vermeiden. Bei der Flächenaufteilung muss berücksichtigt werden, dass der Startpunkt eines jeden Roboters auf dem zugewiesenen Teilpolygon beziehungsweise in diesem liegt. Eine unterschiedliche Leistung der Roboter kann über die Flächenanforderung je Standort berücksichtigt werden.

Zur formalen Beschreibung des Problems sind als Eingangsdaten ein Polygon P sowie eine (nicht leere) Liste von Standorten S_P gegeben. Für jeden der n Standorte S_i , $i = 1, ..., n$ ist der benötigte Flächenanteil c_i, $i = 1, ..., n$ mit $0 < c_i < 1$ gegeben, sodass $\sum_{i=1}^{n} c_i = 1,0$ gilt. Das Polygon$P$ soll in n, nicht überlappende Polygone zerlegt werden, sodass jeder Standort S_i auf dem Polygon P_i mit Fläche c_i∗Area(P)liegt. Aus der Fläche des Polygons P kann für jeden der n Standorte die benötigte Fläche mit c_i∗Area(P) bestimmt werden.

2 Aufteilung eines einfachen, konvexen Polygons

2.1 Grundidee

Bei der nachfolgend beschriebenen Lösung des Problems wird das konvexe Eingabepolygon CP mithilfe von Liniensegmenten schrittweise zerlegt. Jedes Liniensegment L ist hierbei vom Startpunkt L_s zum Endpunkt L_e orientiert, wobei beide Punkte auf dem Rand von CP liegen. Wenn für L_s und L_e eine Position gefunden wurde, erfolgt eine Zerlegung in zwei Teilpolygone. Die bei jeder Teilung entstehen Teilpolygone erhalten entsprechend ihrer Lage zum Liniensegment L die Bezeichnungen P_L^r für das rechts und P_L^l für das links des Liniensegments liegenden Polygons. Die Liniensegmente (bzw. L_s und L_e) werden so positioniert, dass die Fläche von P_L^r der benötigten Fläche der auf dem Rand von P_L^r liegenden Standorte entspricht (P_L^l analog). Die Zerlegung wird für jedes Teilpolygon P_L^r und P_L^l rekursiv aufgerufen, bis nur noch 1-Standort Polygone vorliegen. Da aus einer Zerlegung eines konvexen Polygons mit einem Liniensegment immer zwei konvexe Polygone und insbesondere kein nicht konvexes Polygon resultiert, ist dieser Ansatz möglich.

2.2 Aufteilung eines einfachen, konvexen Polygons

Aus CP entstehende, konvexe Polygone werden mit CP_i notiert. Mit den genannten Überlegungen lässt sich ein rekursiver *divide-and-conquer* - Algorithmus zur Flächenaufteilung eines konvexen Polygons - basierend auf n Standorten - nun folgendermaßen skizzieren:

Bei jedem Aufruf von *ConvexDivide* (CP) wird zunächst geprüft, ob das übergebene Polygon nur noch einen Standort enthält. Falls ja, ist der Zielzustand für das übergebene Polygon erreicht und es ist keine weitere Flächenaufteilung erforderlich. Falls das Polygon mehrere Standorte enthält, erfolgt eine weitere Aufteilung des Polygons in zwei Teil-Polygone P_L^r und P_L^l , welche dann rekursiv mit *ConvexDivide* aufgerufen werden.

2.3 Positionierung der Schnittlinie

Aus vorangegangenem Kapitel bleibt offen, wie genau die Aufteilung eines konvexen Polygons CP in die Polygone P_L^r und P_L^l erfolgt, sodass anschließend $Area(P_L^r) = AreaRequired(S_1, ..., S_i)$ und $Area(P_L^l) = AreaRequired(S_{i+1}, ..., S_n)$ gilt. Konkret ist zu klären, wie Anfangs- und Endpunkt der Schnittlinien positioniert werden (siehe Listing XX, Zeile 4).
Initialisierung von L_s und L_e beim Aufruf von *ConvexDivide* :

- Der Startpunkt L_s der Linie L wird mit den Koordinaten des ersten Punkts der Liste $W()$ initialisiert, wobei dieser nach Definition ein Polygonpunkt (und kein Standort) ist. Es gilt daher $w_1 \in V()$.
- Der Endpunkt L_e wird mit den Koordinaten des ersten Standorts in $W()$ initialisiert und mit w_k notiert, wobei k der Index in $W()$ ist, bei dem der erste Standort liegt. Da die Standorte nachihrem Vorkommen auf dem Weg von v_1 nach v_l geordnet sind, ist bei einem konvexen Polygon sichergestellt, dass die Standorte $S_2, \ldots, Sn$ alle links der Linie L liegen.

Bei einer Zerlegung mit einer so initialisierten Linie würde $S(P_L^r) = S_1$ und $S(P_L^l) = S_2, ..., S_n$ gelten, wobei S_1 in einer Ecke von P_L^r liegen würde. Je nach Fläche von P_L^r und $AreaRequired(S_1)$ werden folgende Fälle unterschieden:

Fall 1: $Area(P_L^r) > AreaRequired(S_1)$
Nach der Initialisierung der Linie L wird festgestellt, dass die Fläche von P_L^r größer ist als die benötigte Fläche von S_1. In diesem Fall erfolgt eine Verkleinerung von $Area(P_L^r)$ unter Beibehaltung

von $S(P_L^r) = S_1$. Dies geschieht, indem L_e als Drehpunkt fungiert und L_s inkrementell gegen den Uhrzeigersinn entlang des Polygons verschoben wird, bis $Area(P_L^r) = AreaRequired(S_1)$ gilt. Zur Verdeutlichung dieser Vorgehensweise sollen folgende Punkte nochmals herausgestellt werden:

- Durch die Initialisierung kann auf dem Weg von w_1 zu w_k kein weiterer Standort liegen, d. h. $AreaRequired(S(P_L^r))$ ist konstant.
- $Area(P_L^r)$ wird mit Verschiebung von L_s stetig kleiner. Bei $L_s = S_1$ gilt $Area(P_L^r) = 0$.
- L_e ist fest, d.h. S_1 ist stets Teil von P_L^r .

Wenn die Bedingung $Area(P_L^r) = AreaRequired(S_1)$ eintritt, erfolgt eine Polygonzerlegung. Für P_L^r erfolgt keine weitere Zerlegung beim Aufruf von *ConvexDivide* (siehe Listing XX, Zeile 3), da nur S_1 auf dessen Rand liegt. Falls auf dem Rand von P_L^r mehr als ein Standort verbleibt, erfolgt eine erneute Zerlegung beim Aufruf von *ConvexDivide* (P_L^l) (siehe Listing XX, Zeilen 4 und 5).

Fall 2: $Area(P_L^r) < AreaRequired(S_1)$

Nach der Initialisierung der Linie L wird festgestellt, dass die Fläche von P_L^r kleiner ist als die benötigte Fläche von S_1. In diesem Fall erfolgt eine Vergrößerung von $Area(P_L^r)$ mit dem Ziel, die Anforderung von S_1 zu erfüllen. Hierbei fungiert L_s als Drehpunkt und L_e wird auf den nächsten in $W()$ vorkommenden Polygonpunkt oder Standort w_{k+1} gesetzt. Die Anforderung wird erneut geprüft. In diesem Schritt wird L_e von Polygonpunkt zu Polygonpunkt verschoben. Eine inkrementelle Verschiebung entlang des Polygons erfolgt dann unter Fall 2.1 beziehungsweise 2.2.

Hierbei kann es nun vorkommen, dass L_e auf die Koordinaten eines Punktes w_j in $W()$ gesetzt wird, welcher ein Standort $\neq S_1$ ist. Dieser Standort wird dann beim nächsten Vorrücken (also bei w_{j+1}) zur benötigten Fläche von P_L^r hinzugenommen. Bei Fall 2 kann $AreaRequired(P_L^r)$ demnach ansteigen, sodass ein Vorücken von L_e zwar zu einer größeren Fläche von P_L^r , nicht aber unbedingt zu einem günstigeren Verhältnis aus $Area(P_L^r)/AreaRequired(S(P_L^r))$ führt.
L_e wird so oft verschoben, bis eine der folgenden Bedingungen eintritt:

- $Area(P_L^r) > AreaRequired(S(P_L^r))$
- $L_e = S_n$

Je nachdem, wie weit L_e vorrückt und wie die Fläche von P_L^r zur Flächenanforderung von S(P_L^r) ist, werden nun weiter zwei Fälle unterschieden:

Fall 2.1: $L_e = S_n$ **und** $Area(P_L^r) > AreaRequired(S(P_L^r))$

In diesem Fall wird der Endpunkt L_e inkrementell im Uhrzeigersinn entlang des Polygons bewegt, bis $Area(P_L^r) = AreaRequired(S(P_L^r))$ gilt. Hinweis: Angenommen die Ausgangsposition von L_e ist w_j, dann muss es zwischen w_j und w_{j-1} eine Position geben, bei der $Area(P_L^r) = AreaRequired(S(P_L^r))$ gilt, da beim Vorrücken $Area(P_L^r)$ bei w_{j-1} zu klein und bei w_j zu groß war. Dieser Zwischenpunkt wird durch Interpolation gefunden.

Fall 2.2: $L_e = S_n$ **und** $Area(P_L^r) < AreaRequired(S(P_L^r))$

In diesem Fall wird der Anfangspunkt L_s inkrementell im Uhrzeigersinn entlang des Polygons bewegt, bis $Area(P_L^r) = AreaRequired(S(P_L^r))$ gilt.

Diese Vorgehensweise entspricht im Wesentlichen Fall 1, wobei L_s (initialisiert mit w_1) nun nicht im Uhrzeigersinn zum ersten Standort S_1 , sondern gegen den Uhrzeigersinn zum letzten Standort S_n bewegt wird. Vergleiche auch Abbildung XX, Fall (c) und (i).