Polynomial volume point set embedding of graphs in 3D*

Farshad Barahimi† Stephen Wismath‡

Abstract

Two algorithms are presented for computing a point-set embedding of a graph in 3D on a given point set with a volume that is polynomial in the size of the graph and the size of the point set, and with at most a logarithmic number of bends per edge. This resolves the previously open *general 3D point set embedding problem* [12].

1 Introduction

A *drawing* of a graph, is a mapping of each vertex to a point in 2D or 3D Euclidean space and each edge to a simple curve between the mapped points of its endpoints. Although 2D graph drawing has been studied extensively, there has also been some significant progress on drawing graphs in 3D. One such model is a *3D Fary grid drawing*, in which each vertex is mapped to an integer grid point in 3D Cartesian coordinate system and each edge is mapped to a straight line segment, such that there is no crossing between edges or vertices.

Cohen, et al. [5] showed that it is possible to have a 3D Fary grid drawing of any graph with n vertices such that the volume does not exceed $n \times 2n \times 2n$. Although they proved that their $O(n^3)$ result is asymptotically optimal for complete graphs, other classes of graphs can be drawn in a lower volume. Calamoneri and Sterbini [4] showed that it is possible to draw every 4-colorable graph on integer coordinates and with no crossing in an $O(n^2)$ volume. Pach, et al. [13] showed that for any constant r, every r-colorable graph can be drawn crossing-free on integer coordinates in $O(n^2)$ volume. They also showed that their result is asymptotically tight by showing that a balanced complete 2-partite graph with n vertices requires $\Omega(n^2)$ volume.

Bose, et al. [2] showed that the maximum number of non-crossing edges that can be contained in an $X \times Y \times Z$ volume is exactly $(2X-1)(2Y-1)(2Z-1)-XYZ$ and as a result, $\frac{m+n}{8}$ is a lower bound for the volume of a 3D Fary grid drawing of a graph with n vertices and m edges.

Felsner, et al. [9] showed that it is possible to have a 3D Fary grid drawing of any outerplanar graph with n vertices in $O(n)$ volume, using a 3D prism. It remains an open problem to determine if all planar graphs can be drawn in linear volume.

Although in the above results each edge is a straight line segment, another model of drawing graphs in 3D introduces bends to subdivide an edge into straight line segments. Unless otherwise specified, we assume here that all such bend points occur at points with integer coordinates.

Dujmović and Wood [8] showed that it is possible to obtain a 3D crossing-free grid drawing of every graph with n vertices and m edges in a $O(n + m \ \log q)$ volume and with $O(\log q)$ bends per edge, where q is the queue number of the graph. The problem of computing the queue number of a graph is NP-Complete [10]. Di Battista, et al. [6] showed that the queue number of every planar graph is $O(\log^2 n)$ and based on these two results, every planar graph can be drawn crossing-free on integer coordinates in an $O(n \log \log n)$ volume and with $O(\log \log n)$ bends per edge; they also showed that any planar graph can thus be drawn in $O(n \log^8 n)$ volume. Dujmović [7] has recently shown that the queue number of planar graphs is $O(\log n)$, thus improving the volume bound to $O(n \; logn)$.

After acceptance of this paper, we were made aware of a result by D. Wood [15] that uses a technique similar to ours. Both these results were obtained independently.

1.1 Point set embedding

The class of point set embedding problems studies the layout of graphs when a set of fixed points are given for the location of vertices. If the mapping between the vertices and points is specified then it is called *with mapping* otherwise it is called *without mapping.* In the with mapping variant of the problem the layout is determined only by establishing the position of the bends, whereas in the without mapping variant of the problem, identifying the mapping between the vertices and the given point set, is also required.

One formulation of the two dimensional point set embedding problem (2DPSE) was suggested by Meijer and Wismath [12]:

> Given a planar graph G with n vertices, $V = \{v_1, v_2, \ldots, v_n\}$, and given a set of n distinct points $P = \{p_1, p_2, \ldots, p_n\}$ each with inte-

*Supported in part by the Natural Sciences and Engineering Research Council of Canada.

†Department of Mathematics and Computer Science, University of Lethbridge, farshad.barahimi@uleth.ca

‡Department of Mathematics and Computer Science, University of Lethbridge, wismath@uleth.ca

> ger coordinates in the plane, can G be drawn crossing-free on P with v_i at p_i and with a number of bends polynomial in n and in an area polynomial in n and the dimension of P?

Cabello [3] considered a version of the problem where bends are not allowed and proved that it is NP-Hard to determine whether a planar graph has a straight-line crossing-free drawing on a predefined set of points when the mapping between the vertices and the points is not specified. Pach and Wenger [14] proved that it is possible to draw any planar graph crossing-free on a predefined set of points with $O(n^2)$ bends per edge where the mapping between vertices and points is fixed (but bend points are not constrained to occur at integral coordinates). Kaufmann and Wiese [11] proved that it is possible to have a crossing-free drawing of every planar graph, with at most two bends per edge where each vertex can be positioned at any point of a set of predefined positions, but the area of the drawing may be exponential.

In 3D similar issues can be considered. Meijer and Wismath [12] formulated the three dimensional point set embedding problem (3DPSE) as follows:

> Given a graph G with n vertices, $V = \{v_1, v_2, \ldots, v_n\}$, and a set of n distinct points $P = \{p_1, p_2, \ldots, p_n\}$ each with integer coordinates in three dimensions, can G be drawn crossing-free on P with v_i at p_i and with a number of bends polynomial in n and in a volume polynomial in n and the dimension of P?

In this paper without loss of generality, the bounding box of P is assumed to range from $(1, 1, 1)$ to (w, l, h).

In [12], this general problem is stated as an open problem and solutions to modified versions of the problem are given. Barahimi [1], in his master's dissertation provided two algorithms for the general problem, one of which is presented in section 2, and the second one is replaced by another algorithm discussed in section 3.

The first modification to 3DPSE that is considered in [12] is to remove the polynomial volume constraint from the problem definition. They prove that K_n can be drawn crossing-free on any predefined set of integer points in 3D with at most 3 bends per edge, but the volume is unbounded. The proof incrementally adds edges to the graph. For each endpoint of each edge, a visible bend point outside the bounding box of the current drawing is found and the endpoint is connected to that bend. The bends found for each edge can be connected by finding a third visible bend point and connecting both to it. The idea of finding visible bend points is used in the proposed algorithms in sections 2 and 3 but the visible bend points are found in a bounded volume.

The second modification to 3DPSE that is considered in [12] is to restrict P to the XY plane and the problem is called $3DPSE_p$. They proved that a graph with n vertices and m edges can be drawn crossing-free in 3D with vertices on a predefined set of integer points in a $W \times H$ rectangular area of the XY plane using $O(\log m)$ bends per edge and within a bounding box of $\max(W, m) \times (H + 3) \times (2 + \log m)$. To create such a drawing, they first introduce a method to draw a perfect matching of two sets of m points in 2D on $O(\log m)$ tracks with $O(\log m)$ bends per edge and no X-Crossings. An X-Crossing occurs if there are two edges (u, v) and (w, z) such that u and w are on the same track and v and z are both on a different track, and u appears before w in their track but v appears after z in their track. This track layout can be converted to 3D without any edge crossings in a box of volume of $m \times 3 \times (1 + \log m)$ and with $O(\log m)$ bends per edge. This technique is also used in the first proposed algorithm of this paper described in section 2. To draw an arbitrary graph in 3D, two lines are considered and for each edge two bend points are added, one on the first line and one on the second line. In the first line the order of the bends representing edges is lexicographic meaning that edges of the vertex v_i appear before the edges of the vertex v_{i+1}. For the second line the order of the bends representing edges is opposite of the first line meaning that edges of the vertex v_i appear after the edges of the vertex v_{i+1}. The two corresponding bends of each edge on these two lines are connected on $O(\log n)$ tracks with $O(\log n)$ bends using the perfect matching technique. Next another line is added for vertices of the graph and vertices are connected to the corresponding bend of their incident edges without creating any crossings. For the $3DPSE_p$ problem, without loss of generality it is assumed that the vertices are ordered by X coordinate and then by Y coordinate in case of a tie. The vertices are placed in the $Z = 0$ plane. The first line for the matching is placed at the $Z = -1$ plane and the second line of the matching is put on the $Z = 1 + \log m$ plane.

Barahimi [1], in his master's dissertation proposed two algorithms for the general $3DPSE$ problem.The first algorithm which is also presented here creates a drawing of volume $O(m + n + w) \times O(m + n + l) \times O(\log n + h)$, with at most $O(\log n)$ bends per edge. A second algorithm is proposed in this paper which fits the drawing in a $O(m + n + w) \times O(m + n + l) \times O(h)$ volume and uses only one bend per edge.

2 The algorithm with a logarithmic number of bends per edge

In this section an algorithm is given which will produce a drawing of size
$O(m + n + w) \times O(m + n + l) \times O(\log n + h)$, with at most $O(\log n)$ bends per edge.

2.1 General idea

The algorithm has three phases and the general ideas are outlined below while details follow later:

- Phase one: Consider two rectangles R_A and R_B that lie in planes parallel to the XY plane. R_A is one unit in front of the bounding box of the points in the direction of the Z axis and R_B is one unit from the back of the bounding box of the points in the direction of the Z axis. For each edge find two visible integer bend points, one in R_A and one in R_B. Connect the first vertex of the edge to the bend point in R_A and connect the second vertex of the edge to the bend point in R_B.

- Phase two: Consider two lines L_A and L_B parallel to the Y axis. L_A is at least one unit in front of R_A in the direction of the Z axis and two units to the left of R_A in the direction of the X axis. L_B is one unit from the back of R_B in the direction of the Z axis and two units to the left of R_A in the direction of the X axis. Connect each bend point in R_A to a corresponding integer bend point in L_A and connect each bend point in R_B to a corresponding integer bend point in L_A.

- Phase three: Each edge has two corresponding bend points in L_A and L_B. If the corresponding bend points of each edge in L_A and L_B are connected then they form a matching. This matching can be drawn crossing free using the perfect matching technique of [12] in a bounding box of $3 \times m \times (1 + \log m)$.

Figure 1 shows a conceptual picture of R_A, R_B, L_A, L_B and the bounding box of the points.

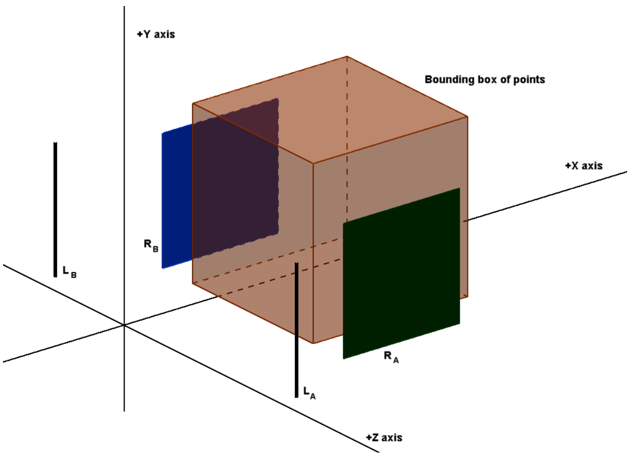

Figure 1: The conceptual picture of R_A, R_B, L_A, L_B and the bounding box of the points.

2.2 Phase one

Let $k = \max(n, m)$. Let P_A denote the plane $z = h + 1$ and R_A denote the rectangle going from $(1, 1, h+1)$ to $(2k, 2k, h+1)$ in the plane P_A. Let P_B denote the plane $z = 0$ and R_B denote the rectangle going from $(1, 1, 0)$ to $(2k, 2k, 0)$ in the plane P_B. A point s is visible from point t if the line segment connecting s to t does not intersect any vertex of G or any line segment that is previously drawn. The edges are considered one by one in m steps. At the i^{th} step $(1 \leq i \leq m)$, the i^{th} edge e_i, connecting vertices u_i and w_i is considered. Now a visible integer bend point a_i from u_i is found in R_A and a line segment α_i is drawn between u_i and a_i. Next a visible integer bend point b_i from w_i is found in R_B and a line segment β_i is drawn between w_i and b_i. At the end of this phase each edge has one corresponding bend point in R_A and one corresponding bend point in R_B.

To prove that there is always a visible integer bend point from u_i in R_A, or from w_i in R_B, at the i^{th} step of this phase of the algorithm, consider that there are only two ways that an integer bend point in R_A or R_B becomes invisible from u_i or w_i:

1. A previously drawn line segment is between R_A and u_i, or R_B and w_i. The previously drawn line segment can be any of α_j or β_j for $1 \leq j < i$, or α_i for w_i. There are at most $2k - 1$ such line segments and each line segment can make at most $2k$ integer points of R_A or R_B invisible. So this case will make at most $(2k-1)2k$ integer points of R_A or R_B invisible. To prove that each such line segment connecting vertices or bend points q and t, will make at most $2k$ integer points in R_A or R_B invisible from a vertex v which can be u_i or w_i, consider the plane P_{vqt} containing v, q and t. If the plane P_{vqt} intersects with the plane P_A or P_B the intersection will be a line. This line can contain at most $2k$ integer points of R_A or R_B. If v, q, and t are collinear, at most one integer point of R_A or R_B is made invisible.

2. A vertex is between R_A and u_i, or R_B and w_i: This can be any vertex other than u_i or w_i. Each such vertex can make at most one integer point of R_A or R_B invisible. There are at most $k-1$ such vertices. So this case can make at most $k - 1$ integer points of R_A or R_B invisible.

Subtracting the maximum number of invisible points of both cases from the number of integer points of R_A or R_B, leaves at least $k + 1$ visible points as shown in equation 1.

$$4k^2 - (2k-1)2k - (k-1) = k + 1 \qquad (1)$$

2.3 Phase two

Let $\lambda = \max(h+2, \log m)$. Let L_A denote the line segment going from $(-1,1,\lambda)$ to $(-1,m,\lambda)$ and let L_B denote the line segment going from $(-1,1,-1)$ to $(-1,m,-1)$.

For each bend point a_i in R_A, find a corresponding integer bend point in L_A called $\bar{a}_i$ and draw a line segment between a_i and $\bar{a}_i$. To find such corresponding bend points, consider the integer bend points of L_A in the order of increasing Y coordinate and consider a_i integer bend points in the order of X coordinate and in case of a tie in the order of Y coordinate, and match them one by one. This ordering will avoid any crossings.

Similarly, for each bend point b_i in R_B, find a corresponding integer bend point in L_B called $\bar{b}_i$ and draw a line segment between b_i and $\bar{b}_i$. To find such corresponding bend points, consider the integer bend points of L_B in the order of increasing Y coordinate and consider b_i integer bend points in the order of X coordinate and in case of a tie in the order of Y coordinate, and match them one by one. This ordering will avoid any crossings. At the end of this phase each edge has four corresponding bend points, one in R_A, one in L_A, one in L_R and one in L_B.

2.4 Phase three

Each edge e_i has a corresponding bend point $\bar{a}_i$ in L_A and a corresponding bend point $\bar{b}_i$ in L_B. If each $\bar{a}_i$ is connected directly to each $\bar{b}_i$ they form a perfect matching but it may introduce crossings. To avoid crossings the perfect matching technique of [12] is used to draw this perfect matching in 3D. Such a 3D perfect matching drawing can be drawn in a bounding box of size $3 \times m \times (1 + \log m)$ using the [-2,0] range of X coordinates and at most $O(\log n)$ bends per edge. Also it is notable that this phase does not use any bend point on the two lines $X = 0, Z = \lambda$ and $X = 0, Z = -1$, otherwise it may introduce crossings with the line segments of the previous phase. This phase will add at most $O(\log n)$ bends per edge, and at most three units to the dimension of drawing in X direction.

2.5 Summary

Each phase of the algorithm does not create any crossings. Each of the three phases uses different partitions of space which will avoid crossings between the three phases.

To find the visible points, for each vertex v, the algorithm maintains a set of integer points in R_A or R_B that are visible from v. The set is implemented using a balanced binary search tree. After adding each line segment at each step of the algorithm, for each vertex v, the algorithm removes the integer points blocked by that line segment from the set of visible points of v. The algorithm has $O(m \cdot n \cdot k \cdot \log n)$ time complexity and $O(nk^2)$ memory complexity. The algorithm is summarized in Theorem 1 and Algorithm 2.1. Also figure 2 shows the drawing of K_5 on a given point set produced by the proposed algorithm using software We3Graph [1].

Theorem 1 *Given a graph G with m edges, and n vertices, $V = \{v_1, v_2, \ldots, v_n\}$, and a given set of n distinct points $P = \{p_1, p_2, \ldots, p_n\}$ each with integer coordinates in three dimensions, G can be drawn crossing-free on P with v_i at p_i and with at most $O(\log n)$ bends per edge and in a $O(m+n+w) \times O(m+n+l) \times O(\log n + h)$ volume such that each bend has three dimensional integer coordinates. The drawing can be produced in $O(m \cdot n \cdot k \cdot \log n)$ time and $O(nk^2)$ memory.*

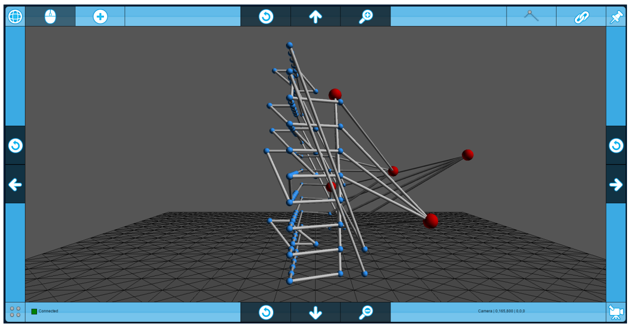

Figure 2: 3D drawing of K_5 on a given point set using the first proposed algorithm. Y axis upward and camera looking toward the negative side of Z axis direction.

3 The algorithm with one bend per edge

In this section an algorithm is given which will produce a drawing of size $O(m+n+w) \times O(m+n+l) \times O(h)$, with exacly one bend per edge. The algorithm considers a rectangle parallel to the XY plane in front of the bounding box of the points in the direction of the Z axis and for each edge, finds an integer bend point in that rectangle that is visible from both endpoints of the edge and connects the endpoints of the edge directly to the bend point. Here is a detailed explanation of the algorithm.

Let $k = \max(n, m)$. Let P_C denote the plane $z = h+1$ and R_C denote the rectangle going from $(1, 1, h+1)$ to $(4k, 4k, h+1)$ in the plane P_C. A point s is visible from point t if the line segment connecting s to t does not intersect any vertex of G or any line segment that is previously drawn. At the i^{th} step $(1 \leq i \leq m)$, the i^{th} edge e_i, connecting vertices u_i and w_i is considered. Now an integer bend point a_i is found in R_C that is visible both from u_i and w_i. A line segment α_i is drawn between u_i and a_i and a line segment β_i is drawn between w_i and a_i. Figure 3 shows a conceptual picture of R_C and the bounding box of the points.

Algorithm 2.1 The algorithm with logarithmic number of bends per edge

R_A denotes the rectangle going from $(1, 1, h+1)$ to $(2k, 2k, h+1)$ in the plane $z = h+1$
R_B denotes the rectangle going from $(1, 1, 0)$ to $(2k, 2k, 0)$ in the plane $z = 0$

```
λ = max(h + 2, log m)
Let S_v be the set of all visible integer points from the vertex v, in R_A.
Let Ŝ_v be the set of all visible integer points from the vertex v, in R_B.
for all vertex v in V do
    for all vertex v_2 in V - v do
        Remove the point in S_v or the point in Ŝ_v that is blocked by v_2 from v (if it exists).
for all edge e_i = (u_i, w_i) in E do
    Let a_i be a point in S_{u_i}
    Draw a line segment α_i from u_i to a_i.
    for all vertex v in V do
        Remove every point in S_v and Ŝ_v that is blocked by α_i from v.
    Let b_i be a point in Ŝ_{w_i}
    Draw a line segment β_i from w_i to b_i.
    for all vertex v in V do
        Remove every point in S_v and Ŝ_v that is blocked by β_i from v.
counter=1
for all α_i ordered by x coordinate and in case of a tie by y coordinate do
    Draw a line segment between α_i and the point ā_i = (−1, counter, λ).
    counter++
counter=1
for all β_i ordered by x coordinate and in case of a tie by y coordinate do
    Draw a line segment between β_i and the bend point b̄_i = (−1, counter, −1).
    counter++
Use the technique of [12] for drawing a perfect matching in 3D to connect each ā_i to b̄_i.
```

To prove that there is always an integer bend point visible from both of u_i and w_i in R_C at the i^{th} step of the algorithm, consider that there are only two ways that an integer bend point in R_C becomes invisible from u_i or w_i:

1. A previously drawn line segment is between R_C and, u_i or w_i. The previously drawn line segment can be any of α_j or β_j for $1 \leq j < i$. There are at most $2k-2$ such line segments and each line segment can make at most $4k$ integer points of R_C invisible from u_i and at most $4k$ integer points of R_C invisible from w_i. So this case will make at most $2(2k-2)4k$ integer points of R_C invisible from either of u_i or w_i. To prove that each such line segment connecting vertices or bend points q and t, will make at most $4k$ integer points in R_C invisible from a vertex v which can be u_i or w_i, consider the plane P_{vqt} containing v, q and t. If the plane P_{vqt} intersects with the plane P_C the intersection will be a line. This line can contain at most $4k$ integer points of R_C. If v, q, and t are collinear, at most one integer point of R_C is made invisible.

2. A vertex is between R_C and, u_i or w_i: This can be any vertex other than u_i or w_i. Each such vertex can make at most one integer point of R_C invisible from u_i and at most one integer point of R_C invisible from w_i. There are at most $k-1$ such vertices. So this case can make at most $2(k-1)$ integer points of R_C invisible from either of u_i or w_i.

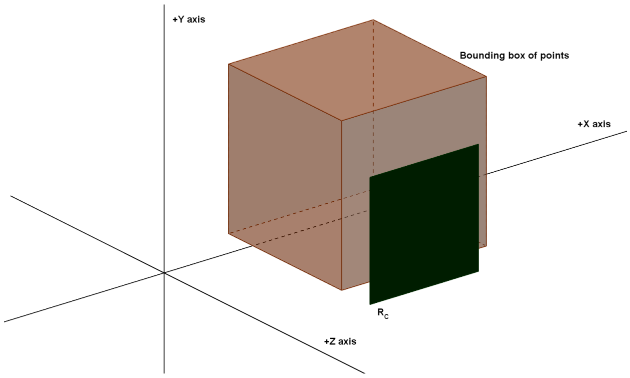

Figure 3: The conceptual picture of R_C and the bounding box of the points.

Subtracting the maximum number of integer points invisible from both u_i and w_i of both cases from the total number of integer points of R_C, leaves at least $14k+2$ visible points as shown in equation 2.

$$16k^2 - 2(2k-2)4k - 2(k-1) = 14k + 2 \qquad (2)$$

To find the visible points, for each vertex v, the algorithm maintains a set of integer points in R_C that are visible from v. The set is implemented using a 2D array. After adding each line segment at each step of the algorithm, for each vertex v, the algorithm removes the integer points blocked by that line segment from the set of visible points of v. The algorithm has $O(mk^2)$ time complexity and $O(nk^2)$ memory complexity. The algorithm is summarized in Theorem 2 and Algorithm 3.1. Also figure 4 shows the drawing of K_5 on a given point set using the proposed algorithm.

Theorem 2 *Given a graph G with m edges, and n vertices, $V = \{v_1, v_2, \ldots, v_n\}$, and a given set of n distinct* *points $P = \{p_1, p_2, \ldots, p_n\}$ each with integer coordinates in three dimensions, G can be drawn crossing-free on P with v_i at p_i and with exactly one bend per edge and in a $O(m+n+w) \times O(m+n+l) \times O(h)$ volume such that each bend has three dimensional integer coordinates. The drawing can be produced in $O(mk^2)$ time and $O(nk^2)$ memory.*

Algorithm 3.1 The algorithm with one bend per edge

R_C denotes the rectangle going from $(1, 1, h+1)$ to $(4k, 4k, h+1)$ in the plane $z = h+1$

Let S_v be the set of all visible integer points from the vertex v, in R_C.
for all vertex v in V **do**
  for all vertex v_2 in V - v **do**
    Remove the point in S_v that is blocked by v_2 from v (if it exists).
for all edge $e_i = (u_i, w_i)$ in E **do**
  Let a_i be a point in both S_{u_i} and S_{w_i}
  Draw a line segment α_i from u_i to a_i.
  Draw a line segment β_i from w_i to a_i.
  for all vertex v in V **do**
    Remove every point in S_v that is blocked by α_i or β_i from v.

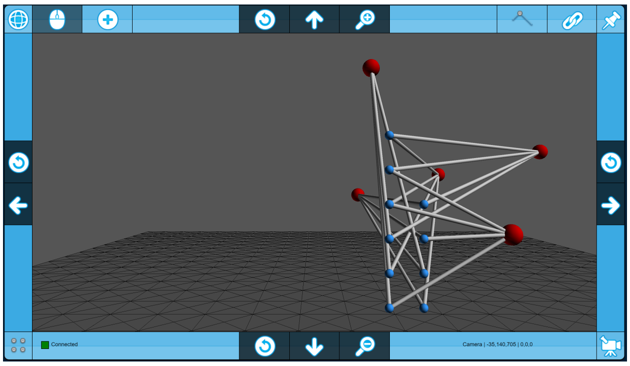

Figure 4: 3D drawing of K_5 on a given point set using the second algorithm. Y axis upward and camera looking toward the negative side of Z axis direction.

4 Comparison of the two algorithms

While the second algorithm, with lower number of bends per edge provides an equal or better asymptotic volume, the first algorithm with a better asymptotic running time for dense graphs, might result in lower exact volume since it uses two $2k \times 2k$ rectangles instead of one $4k \times 4k$ rectangle to find visible integer bend points.

5 Conclusions and open problems

Two algorithms were presented to answer a previously raised question in the 3D graph drawing literature. Although the algorithms run in polynomial time, improving the practical and asymptotic runtime performance should be considered. Also, although the algorithms produce drawings in 3D without crossings, edges can be very close, thus finding an algorithm which can guarantee a particular minimum distance between edges or vertices is another area which can be investigated.

References

[1] F. Barahimi. Web-based drawing software for graphs in 3d and two layout algorithms. Master's thesis, U. of Lethbridge, Dept. of Math. and Comp. Sci., 2015.

[2] P. Bose, J. Czyzowicz, P. Morin, and D. R. Wood. The maximum number of edges in a three-dimensional grid-drawing. *J. Graph Alg. & App.*, 8(1):21–26, 2004.

[3] S. Cabello. Planar embeddability of the vertices of a graph using a fixed point set is np-hard. *J. Graph Alg. & App.*, 10(2):353–363, 2006.

[4] T. Calamoneri and A. Sterbini. 3d straight-line grid drawing of 4-colorable graphs. *Information Processing Letters*, 63(2):97–102, 1997.

[5] R. Cohen, P. Eades, T. Lin, and F. Ruskey. Three-dimensional graph drawing. *Algorithmica*, 17(2):199–208, 1997.

[6] G. Di Battista, F. Frati, and J. Pach. On the queue number of planar graphs. *SIAM J. Computing*, 42(6):2243–2285, 2013.

[7] V. Dujmović. Graph layouts via layered separators. *J. of Combinatorial Theory, Series B*, 110:79–89, 2015.

[8] V. Dujmović and D. Wood. Stacks, queues and tracks: Layouts of graph subdivisions. *Discrete Math. & Theoretical Computer Science*, 7(1), 2006.

[9] S. Felsner, G. Liotta, and S. Wismath. Straight-line drawings on restricted integer grids in two and three dimensions. *J. Graph Alg. & App.*, 7(4):363–398, 2003.

[10] L. Heath and A. Rosenberg. Laying out graphs using queues. *SIAM J. Computing*, 21(5):927–958, 1992.

[11] M. Kaufmann and R. Wiese. Embedding vertices at points: Few bends suffice for planar graphs. *J. Graph Alg. & App.*, 6(1):115–129, 2002.

[12] H. Meijer and S. Wismath. Point set embedding in 3d. *J. Graph Alg. & App.*, 19(1):243–257, 2015.

[13] J. Pach, T. Thiele, and G. Toth. Three-dimensional grid drawings of graphs. In G. Di Battista, editor, *Graph Drawing*, volume 1353 of *LNCS*, pages 47–51. Springer Berlin Heidelberg, 1997.

[14] J. Pach and R. Wenger. Embedding planar graphs at fixed vertex locations. *Graphs and Combinatorics*, 17(4):717–728, 2001.

[15] D. Wood. Three dimensional graph drawing with fixed vertices and one bend per edge. *arXiv: http://arxiv.org/abs/1606.09188*, 2016.