

映像合成処理に適したノードグラフのレイアウト手法

Node graph layout technique for digital compositing process

居出 貴之† 今井 敏行‡ 床井 浩平‡

Takayuki Ide Toshiyuki Imai Kohei Tokoi

1. はじめに

近年ノードベースのビジュアルプログラミングが普及してきており、映像制作から自動車の組み込みシステム開発まで幅広く用いられている。

ノードベースのビジュアルプログラミングでは、ノードエディタ上で処理を表すノードどうしを線でつなぎ、処理の全体像を表すノードグラフを作成する。ノードグラフの可読性を高めるためには、処理の順番にノードを並べるなどのノードの配置を工夫する手間と時間が必要となる。そこで作業の効率化のために、ノードグラフを自動でレイアウトするためのグラフ描画アルゴリズムが用途に応じていくつか開発されている。例えばゲームエンジン Unreal Engine のサードパーティ製アドオン Blueprint Assist などがある[1]。また、これらのアルゴリズムではグラフドローイングの分野の技術が応用されている。

本研究の目的は、映像制作の効率化のために、映像合成処理に適したノードグラフを自動でレイアウトするグラフ描画アルゴリズムを考案することである。

提案手法を開発するにあたって、まずは映像合成処理に適したノードグラフの可読性を向上させるための美的基準をいくつか定めた。また、その美的基準を満たすように提案手法を考案した。

また映像制作の効率化のためには、映像合成処理のノードグラフを記述するのにかかる時間のうち、ノードグラフのレイアウトを整える時間を節約したい。そのため、提案手法の実行時間は数秒程度にすることを目標とした。

最後に、映像制作ソフト「Davinci Resolve」のノードエディタを模したプロトタイプのシステムを作成した。プロトタイプのシステムを用いて、提案手法で美的基準に沿ったレイアウトができるかどうか、また実行時間が目標通りであるかを検証した。

以下、2 章では映像合成処理に適したノードグラフの特徴、3 章では美的基準について説明する。

4 章では提案手法の参考にした既存のグラフ描画アルゴリズムについて説明し、5 章では提案手法について説明する。6 章では評価実験の結果と考察を示し、7 章で本研究の結果をまとめ、今後の課題について述べる。

2. 映像合成処理に適したノードグラフの特徴

映像合成処理に適したノードグラフは有向グラフの一種で、映像や画像データの処理の流れを表す。ノードは映像データや処理（合成など）を表し、エッジはデータや処理の流れを表している。また、映像合成処理では、2 つ以上のデータの流れが合流して 1 つになる、ということがある。

例えば「合成」を表すノードでは、前の方に来る景色を表すデータと、後ろに来る景色を表すデータが合流する。

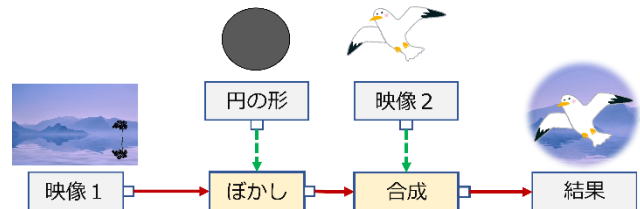


図 1. ノードエディタ上の映像合成のイメージ
(文献[2]より引用し一部改変)

3. 映像合成処理に適したノードグラフにおけるグラフの美的基準

グラフドローイングの分野においては、グラフのレイアウトの可読性を向上させるためのノードやエッジの配置のルールのことを美的基準と呼ぶ[3][4][5]。なお、本研究における可読性とは、ノードやエッジ同士の視覚的な区別のしやすさを考慮した可読性と、映像合成処理そのものの意味を考慮した可読性のことである。

映像合成処理に適したノードグラフに要求される美的基準は以下の 7 つであると考えた。

- (1) エッジが特定の方向に従う
- (2) ノードをグリッドに沿って配置する
- (3) エッジの長さが初期配置と比べ変化しない
- (4) エッジの交差数を最小にする
- (5) ノードとエッジが交差しないようにする
- (6) ノード同士の重なりを禁止する
- (7) エッジ間の角度は小さくしない

以上のうち(1)~(3)は映像合成処理に適したノードグラフの特徴を考慮した基準であり、(4)~(7)はどのグラフでも広く採用されている基準である[4][5]。

(1)~(3)の美的基準を採用した根拠について以下に記す。

(1) エッジが特定の方向に従う

小町[6]によると、映像合成処理に適したノードグラフでは、処理の順番が分かりやすい方がよい。そのため基本はエッジの向きを一方向にした方がよいとしている。また、どの処理のタイミングでどのデータが合流するかを分かりやすくするために、データの合流を表すエッジの向きをそれとは別に決めた方がよいとしている[3]。例えば、図 2 ではエッジの向きは基本右向きだが、映像 2 の合流を表すエッジは下向きにしている。

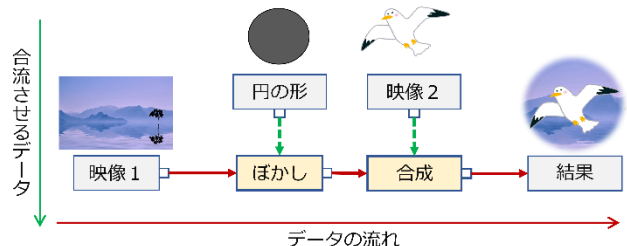


図 2. エッジが特定の方向に従う

†和歌山大学大学院システム工学研究科,
Wakayama University, Graduate School of Systems Engineering

‡和歌山大学システム工学部,

Wakayama University, Faculty of Systems Engineering

(2) ノードをグリッドに沿って配置する

ノードグラフに限らず、要素をグリッドに配置した方が情報を整理しやすいと言われている[7].

また、ノードベースの映像合成を採用している代表的なソフトである *Davinci Resolve* や *Nuke* では、ノードエディタにグリッドを表示できる。また、ノードを動かす際にノードがグリッドにスナップするように設定することも可能になっている[8][9]。さらに、小町[6]も映像合成処理のノードグラフにおいては、ノードをグリッドに沿って配置することを推奨している。

(3) エッジの長さが初期配置と比べ変化しない

グラフドローイングの分野では、ここで挙げた基準とは異なる「エッジの長さはすべて一定にする」という基準の方が広く採用されている。

しかし、映像合成処理に適したノードグラフの場合、エッジの長さをすべて一定にすると無理なレイアウトが生じる場合があると自分で判断した。またユーザの考えた初期配置も大きく崩し、ユーザに混乱をもたらす可能性もあると考え、この基準を考えた。

例えば図3では、位置の遠いノード同士を接続するために他より極端に長いエッジがある。このエッジを短くしようとするとノードの配置を大きく変える必要がある。また基準の1つである「エッジが特定の方向に従う」を大きく崩す可能性もある。

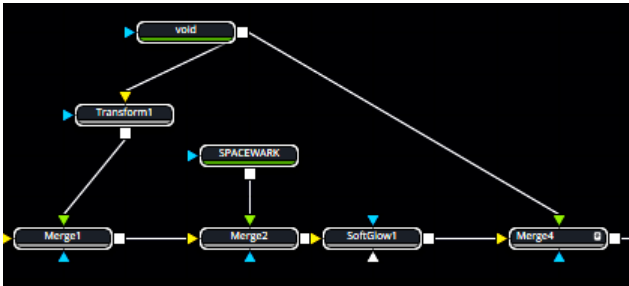


図3. 文献[10]をもとに作成したノードグラフ
(文献[2]より引用し抜粋)

4. 既存のグラフ描画アルゴリズム

有向グラフをレイアウトする手法のうち、提案手法ではマグネティック・スプリング・モデル[11]をもとに手法を考案した。この手法ではエッジの役割に応じエッジを異なる方向に整えることが可能である。

有向グラフをレイアウトするアルゴリズムとしては、杉山フレームワークが広く知られている[12]。しかし杉山フレームワークはすべてのエッジの向きを一方方向にする手法であるため、映像合成処理に適したノードグラフには適さない。

マグネティック・スプリング・モデルは力指向アルゴリズムの一種である。力指向アルゴリズムではノードに仮想的な力を一定回数働かせグラフをレイアウトする。この手法では次の3つの力が設定されている。

- (1) ノードを回転させエッジの方向を調節する力
- (2) エッジの長さを一定にするための引力・斥力
- (3) ノード同士を反発させノード間隔を整える力

(1)~(3)の力について、以下に詳細に記す。また、4.1節でマグネティック・スプリング・モデルの処理の手順、4.2節でマグネティック・スプリング・モデルの欠点について詳細に記す。

(1) ノードを回転させエッジの方向を調節する力

エッジを特定の向きにそろえるための力である。

この力では、いくつかの、あるいはすべてのエッジは磁針とみなされグラフの置かれた磁場から回転力を受ける。

有向エッジの場合、エッジが定義された磁場における北を向くようにノードに回転力がかかる。無向エッジの場合、エッジに指定された向きがないので、方向だけをそろえる。そのため、磁場の北向き、あるいは南向きに近いノードの方がその向きを向くように回転力が働く(図7)。

ノードにかかる回転力を f_m とすると、 f_m の大きさは次のように表される。

$$|f_m| = c_m b d^\alpha |\Delta t|^\beta$$

b は基準点(実現上は辺の midpoint)における磁場の強さ、 d は現在のエッジの長さである。 Δt はエッジの基準点における磁場の北からのエッジの向きのずれの角度である。定数 α はエッジの長さの、定数 β は Δt の、それぞれの回転力への影響を制御する。また、 c_m は他の力とのバランスを制御するための係数である。以下に現れる c_s, c_r も同様である(図7)。

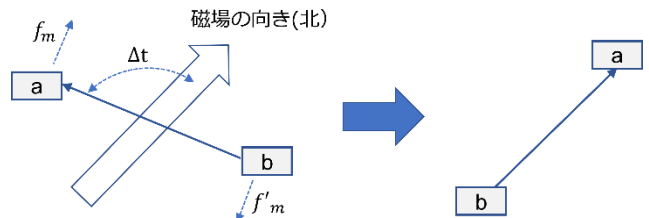


図4. ノードを回転させる力(文献[2]より引用し抜粋)

(2) エッジの長さを一定にするための引力・斥力

エッジの長さをすべて均一にするための力である。エッジをばねとみなし、エッジの両端のノードには、ばねによって引力あるいは斥力が働くものとして考える。また、エッジの理想の長さをばねの自然長とする。エッジの両端の頂点のうち1つの頂点がそのエッジから受ける力を f_s とすると、 f_s の大きさは次のように表される。

$$|f_s| = c_s \log(d/d_0)$$

d は現在のエッジの長さ、 d_0 はばねの自然長、つまりエッジの理想的な長さである。 $d > d_0$ の時は引力がかかり、 $d < d_0$ の時は斥力がかかる。 $d = d_0$ の時は力が働かない。

(3) ノード同士を反発させノード間隔を整える力

エッジで接続されていないノード間の重なりや接近しすぎを防ぐために、ノード同士を反発させる力である。

ノードにかかる斥力を f_r とすると、 f_r の大きさは次のように表される。

$$|f_r| = c_r(1/d^2)$$

d は現在のノード間の距離である。

4.2. マグネティック・スプリング・モデルの手順

3つの力の安定状態は、逐次的な方法に従って求める。すなわち、各ノード v にかかる3つの力の合計 f_v を求め、その力によってノードの位置を $\delta * f_v$ だけ移動させることを繰り返して安定状態を求める。繰り返しはあらかじめ与えられた回数定数 M だけ行う。 δ はノードにかかる力を小さくし、ノードが大幅に動くことを防ぐ(図5)。

図5の疑似コードは、野口らの論文[14]を参考に記述した。

各ノードを初期配置する
(以下の処理を回数定数 M だけ繰り返す)

```
{
  for (v = 0 to ノード数) {
    for (w = 0 to ノード数) {
      if (ノードvとwがエッジeで接続) {
        then
          エッジeに対してvが受ける $f_m$ を計算;
          エッジeに対してvが受ける $f_s$ を計算;
           $f_v := f_v + f_m + f_s$ ;
        else
          ノードwに対してvが受ける $f_r$ を計算;
           $f_v := f_v + f_r$ ;
      }
    }
  }
  ノードvを $\delta * f_v$ だけ移動する.
}
```

図 5. マグネティック・スプリング・モデルの手順

4.2. マグネティック・スプリング・モデルの欠点

提案手法では、4.2.1 節、4.2.2 節で詳細に記すマグネティック・スプリング・モデルの欠点を補う。

4.2.1. エッジ間の角度が狭まり、交差が発生する

Leu[13]によると、マグネティック・スプリング・モデルでは、同一の起点から向きが同じエッジが複数出ていた場合、エッジ間の角度が過度に狭まることもある。また、そのせいでノードとエッジの重なりが起こる場合がある (図 6)。

野口らの手法[14]では、ノードとノードの中心間を結んだ線分のうち、ノードと重なっていない部分をエッジとしているため、角度の狭まりが見られない (図 7)。しかし実際のノードグラフでは、エッジの起点がノードの外にある場合が多いため、野口らの手法はノードグラフに適用できない。また、野口らの手法でも、ノードとエッジの重なりが見られた。

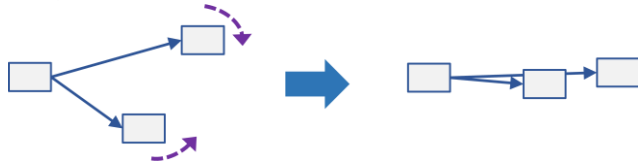


図 6. エッジ間の角度が過度に狭まる場合

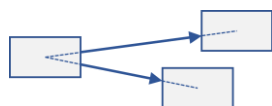


図 7. 野口らの手法[14]におけるエッジ

4.2.2. 異なる方向のエッジで回転力が対立する

自分でマグネティック・スプリング・モデルの実装を試したところ、同じノードから異なる方向を向くエッジが出力されていた場合 (図 8)、それぞれの回転力が対立し常にかかり続け、最終的に崩れたレイアウトになることがあった。

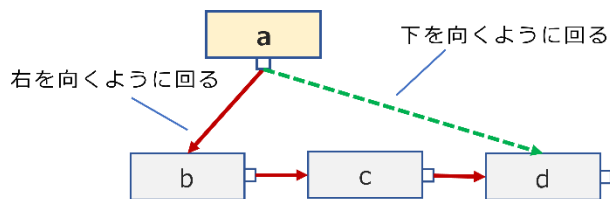


図 8. 異なる方向のエッジが出力されている場合
(文献[2]より引用し一部改変)

5. 提案手法

提案手法では力指向アルゴリズムによりレイアウトを整える。その後ユーザの任意でグリッドレイアウトアルゴリズムによりノードをグリッドに沿わせる。Davinci Resolve や Nuke などの映像合成ソフトでは、グリッドにノードをスナップさせる機能を使用するかどうか選択できるようになっている [8][9] ため、それに倣いグリッドレイアウトアルゴリズムの使用はユーザの任意とした。

また、マグネティック・スプリング・モデルでは、ノードはすべて大きさを持たない点として扱われる。しかし、ノードグラフではノードに大きさがあるため、ノード間の距離はノードの大きさを考慮して決める必要がある。

5.1 節で力指向アルゴリズムについて述べ、5.2 節でグリッドレイアウトアルゴリズムについて述べる。最後に、5.3 節でノードの大きさの考慮について述べる。

5.1. 力指向アルゴリズム

力指向アルゴリズムは次の 5 つの仮想的な力を働かせる。

- (1) ノードを回転させエッジの方向を調節する力
- (2) エッジの長さを調節する引力・斥力
- (3) ノード同士を反発させノード間隔を整える力
- (4) エッジ間の角度の狭まりを防ぐための斥力
- (5) ノードとエッジの交差を防ぐための斥力

以上の力のうち、(1)~(3)の力はマグネティック・スプリング・モデルを、3 章の映像合成処理のノードグラフの美的基準や 4.2 節のマグネティック・スプリング・モデルの欠点を踏まえ改良したものである。(1)の力では 4.2.1 節で述べた異なる方向のエッジで回転力が対立する欠点を補った。(2)の力は、マグネティック・スプリング・モデルではエッジの長さを一定にしていたが、提案手法ではエッジの長さを初期配置の時の長さとして一定にするように働く。

(4),(5)の力は、4.2.1 節で述べたエッジ間の角度が狭まり、ノードやエッジ同士の交差が発生する欠点を補うための力である。

以下に、(1)~(5)の力の詳細について記述する。また 5.1.1 節で処理の手順について記す。

(1) ノードを回転させエッジの方向を調節する力

ノードにかかる回転力を f_m とすると、 f_m の大きさは同じノードから異なる方向のエッジが出力されていない場合とされている場合で分ける。

(1-1) 異なる方向のエッジが出力されていない場合

f_m の大きさは、マグネティック・スプリング・モデルの式と同様である。

エッジは基本右方向を向くようにする。また、データの合流などを表すエッジは下方向を向くようにする。

そのため、基本はエッジで結ばれているノード同士には右向きの磁場を与える（図 9）．また、データの合流を表すエッジで接続されているノード同士には、下向きの磁場を与える（図 10）．

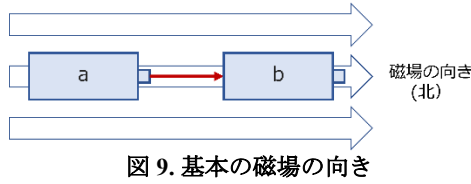


図 9. 基本の磁場の向き

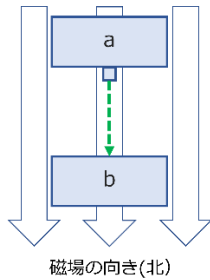


図 10. データの合流を表すエッジの磁場の向き

(1-2) 異なる方向のエッジが出力されている場合

この場合は回転力ではなく、Tim Dwyer らの手法[15]を参考にノードに上下方向、左右方向に力を与えることでエッジの方向を調節する。

ノードを回転させる方法では、ノードを垂直あるいは水平にそろえる。それに対しこの場合は、「ノード a がノード b より一定以上の距離だけ上にあれば良い」という形式でノード間の相対位置のみを制約するため、ノードの位置の制約が回転による制約よりも厳しくなく、力の対立が起こりにくい。

エッジで接続されているノード同士は基本、出力元となっているノードが、出力先のノードよりも一定以上の距離だけ左上に行くようにする（図 11）．

f_m の大きさは、次のように表される。

$$|f_m| = |f_{mx}| + |f_{my}|$$

$$|f_{mx}| = \begin{cases} c_m * (x_b - x_a - d_{x0})^2 / (d_{x0})^2, & x_b - x_a \leq d_{x0} \\ 0, & x_b - x_a > d_{x0} \end{cases}$$

$$|f_{my}| = \begin{cases} c_m * (y_b - y_a - d_{y0})^2 / (d_{y0})^2, & y_b - y_a \leq d_{y0} \\ 0, & y_b - y_a > d_{y0} \end{cases}$$

x_a はノードの出力元のノードの中心の x 座標、 x_b はノードの出力先のノードの中心の x 座標である。 y_a はノードの出力元のノードの下辺の y 座標、 y_b はノードの出力先のノードの上辺の y 座標である。 d_{x0} は x_a から x_b までの理想的な距離、 d_{y0} は y_a から y_b までの理想的な距離である。また、 c_m は他の力とのバランスを制御するための係数である（図 11）．

基本のエッジ（映像の処理の順序を表す）

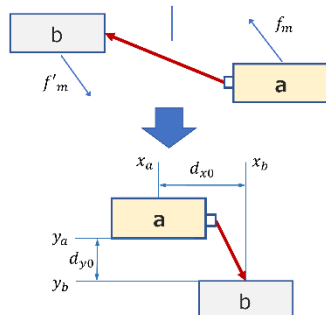


図 11. ノードを左上に移動させる力

データの合流を表すエッジの場合、出力元となっているノードが、出力先のノードよりも一定以上の距離だけ上に行くように力を働かせる（図 12）．

f_m の大きさは、次のように表される。

$$|f_m| = \begin{cases} c_m * (y_d - y_a - d_0)^2 / (d_0)^2, & y_d - y_a \leq d_0 \\ 0, & y_d - y_a > d_0 \end{cases}$$

ここで、 y_a はノードの出力元のノードの下辺の y 座標、 y_d はノードの出力先のノードの上辺の y 座標である。 d_0 は y_a から y_d までの理想的な距離とする。また、 c_m は他の力とのバランスを制御するための係数である（図 12）．

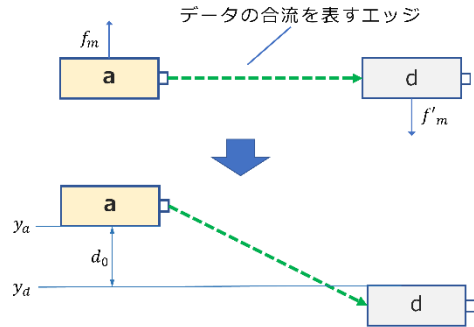


図 12. ノードを上を移動させる力

(2) エッジの長さを調節する引力・斥力

エッジの長さが初期長以上の場合、エッジで接続されたノード間に引力がかかる。初期長未満の場合斥力がかかる。

また、初期配置のときのエッジの長さが指定した最低の長さよりも短かった場合や、エッジで接続されたノード同士が重なっていた場合は、エッジの長さを最低の長さにする。

この力を f_s とすると、 f_s の大きさは次のように表される。

$$|f_s| = \begin{cases} c_s \log(d/d_{min}), & d < d_{min} \\ c_s \log(d/d_f), & d \geq d_{min} \end{cases}$$

d は現在のエッジの長さ、 d_f はエッジの初期長、 d_{min} はエッジの最低の長さである。また、 c_s は他の力とのバランスを制御するための係数である。

(3) ノード同士を反発させノード間隔を整える力

ノード間に斥力を与える。この力を f_r とすると、 f_r の大きさは以下ようになる。

$$|f_r| = \begin{cases} c_r(1/\alpha^2), & d < d_{min} \\ 0, & d \geq d_{min} \end{cases}$$

α は $|f_r|$ を大きくしすぎないための定数である。 d は現在のノード間の距離、 d_{min} はノード間の最低距離である。また、 c_r は他の力とのバランスを制御するための係数である。

本来のマグネティック・スプリング・モデルではノード間の最低距離が定められていなかった。しかし、提案手法ではノード間の最低距離を決め、その最低距離以上ノードが離れていたら斥力をかけないようにする。これは、ノード間の距離が過度に離れるのを防ぐためである。

また、本来のマグネティック・スプリング・モデルではノード間の距離によって引力・斥力の大きさが決まっていたが、提案手法では大きさを一定にする。提案手法の予備実験の結果、大きさを一定にした方が、エッジの交差やノード間の距離が過度に離れるのを防ぐことができたためである。しかし、すべての場合において大きさを一定にする方法が有効であるかは検証できていない。

(4) エッジ間の角度の狭まりを防ぐための斥力

BIGANGLE アルゴリズム[4]をもとに、エッジの角度を一定以上に保つようにノード間に斥力をかける (図 13) . この力を f_{sin} とすると、 f_{sin} の大きさは次のようになる.

$$|f_{sin}| = \begin{cases} c_{sin} * (\alpha/d^\beta) * \sin((\phi - \theta)/2), & \theta \leq \phi \\ 0, & \theta > \phi \end{cases}$$

d はエッジで接続されているノード同士の中心間の距離である. θ はエッジ間の角度, ϕ はエッジ間の最小角度である. α と β はいずれとも d への力の影響を制御する定数である. また, c_{sin} は他の力とのバランスを制御するための係数である (図 13) .

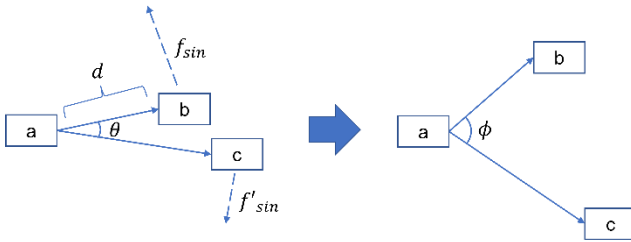


図 13. エッジ間の狭まりを防ぐための斥力
(文献[2]より引用し一部改変)

(5) ノードとエッジの交差を防ぐための斥力

Extended モデル[5]をもとに、エッジに対してノードに斥力をかける (図 14) . この力を f_{er} とすると f_{er} の大きさは以下ようになる.

$$|f_{er}| = \begin{cases} c_{er}(1/d^2), & d < d_{min} \\ 0, & d \geq d_{min} \end{cases}$$

d はエッジと、ノードの四隅のうち一番エッジに近い隅との距離である. また, d_{min} はエッジとノード間の最低距離である. また, c_{er} は他の力とのバランスを制御するための係数である. (図 14) .

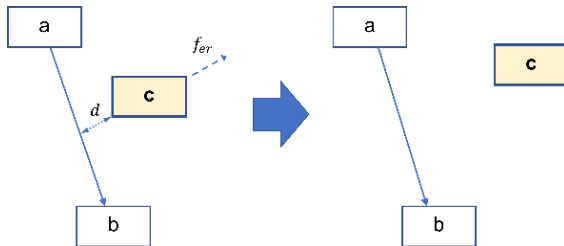


図 14. ノードとエッジの交差を防ぐための斥力

また, 図 15 の場合は位置的にノードがエッジと重なるといことが起こらないため, f_{er} を働かせないことにした.

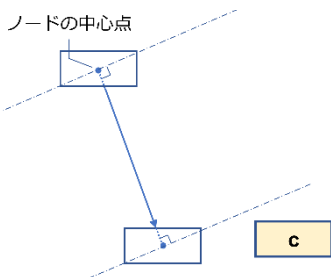


図 15. ノードに斥力がかからない場合

5.2. 力指向アルゴリズムの処理の手順

力指向アルゴリズムの処理の手順を (図 16) で記す.

力指向アルゴリズムではマグネティック・スプリング・モデルと同様, ノードにかかる力を求めた後, ノードを各手法で求めた力の合計の分だけ移動する. 繰り返しはあらかじめ与えられた回数定数 M だけ行う.

「エッジ間の角度の狭まりを防ぐための斥力」と「ノードとエッジの交差を防ぐための斥力」は, 一定回数後から適用する. これら 2 つの手法で求める力は, いずれともエッジに関わる力であるため, 美的基準の一つである「ノードの位置が特定の方向に従う」に悪影響を及ぼす可能性がある. そのため 2 つの手法はある程度ノードの方向が整った後で適用したい, というのが理由である. 本研究では, 「エッジ間の角度の狭まりを防ぐための斥力」は $M/2$ 回目から, 「ノードとエッジの交差を防ぐための斥力」は $3M/4$ 回目から適応させることにしたが, それが最適かどうかまでは検証できていない.

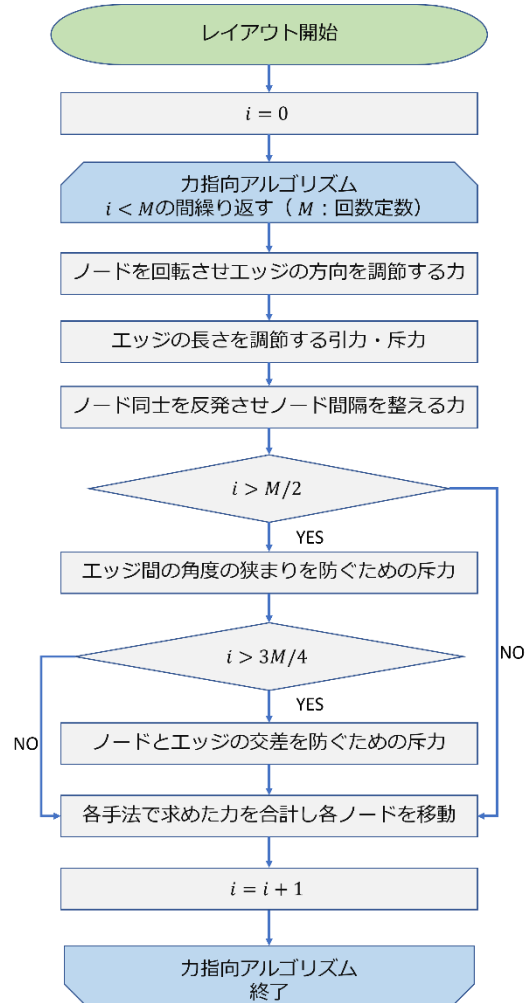


図 16. 提案手法のフローチャート

5.2. グリッドレイアウトアルゴリズム

グリッドレイアウトアルゴリズムは, ハイブリッド・グリッド・レイアウトアルゴリズム[16]を元に考えた. この手法では x 座標が小さい, つまり左にあるノードから順に, ノードに一番近いグリッドの空き格子点にノードを移動させる. x 座標が同じノードがある場合は, y 座標が小さい, つまり上にあるノードから順に移動させる. 一番近いグリッドの格子点にすでに別のノードがあった場合は, ノードに最も近い空き格子点を探索する. ハイブリッド・グリッド・レイアウトアルゴリズム[16]ではどのノードからグリ

ッドに沿わせれば全体のノードの移動量が小さくなるのかを計算し、その順にノードに沿わせていたが、その仕組みは実装できなかった。

また、ノードをグリッドの格子点に移動させるときは、その移動の分だけ、まだ格子点に移動させていないノードを平行移動させる。平行移動させない場合は、ノードの密度が大きい部分ではノードに近い格子点が埋まりやすくなり、ノードに一番近い空き格子点を見つけるのに長い時間がかかる。そこで、平行移動をすることにより、まだ格子点へ移動されていないノードが密度の大きい部分から離れるため、空き格子点を探す時間を節約することができる。

また、ノードから格子点までの距離はマンハッタン距離とする。

5.3. ノードの大きさについての考慮

マグネティック・スプリング・モデルでは、ノードはすべて大きさを持たない点として扱われる。しかし、ノードグラフではノードに大きさがあるため、ノード間の距離はノードの大きさを考慮して決める必要がある。

そこで野口らの手法[14]を踏襲し、提案手法ではノード間の距離をノードの端から端までの実際の距離として考える(図17)。この手法であれば、ノードの形や大きさ、相対位置によらず常にノード間の距離を決定できる。

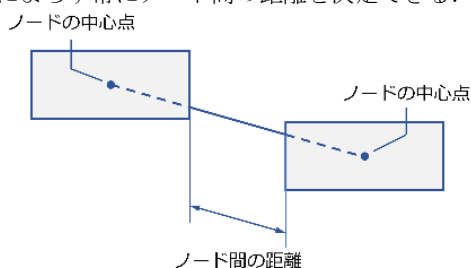


図17. ノード間の距離

この方法では、ノードが重なっていた場合に距離が算出できない。また、この場合のノード間の距離を0とすると、ノードにかかる力が無限大となり計算できない。そこで提案手法では、ノードが重なっていた時はノード間の距離に小さい値を設定することで、この問題を解決する。

野口らの手法[14]では、ノード間の距離が最低距離よりも近づかないようにすることでこの問題を解決していたが、実際のノードエディタでは初期配置でノード同士の重なりのあるノードグラフが作成されることもあるため、野口らの手法そのままでは対応できない。

また、提案手法のアルゴリズムの実行中は、エッジで接続されたノード間の距離をエッジの長さとして考える。提案手法のアルゴリズムの実行中は、実行時間の節約のために実際のエッジの長さや位置を計算していない。エッジの長さや位置は、レイアウトアルゴリズムの適応後に決まる。そのため、提案手法のアルゴリズムの実行中のエッジの長さ、レイアウト後のエッジの長さや位置はわずかに異なっている(図18)

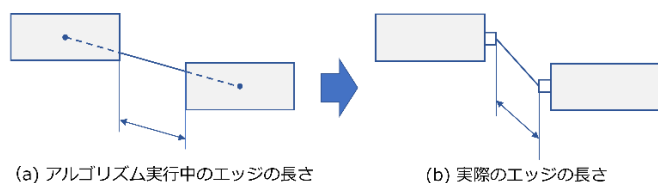


図18. 提案手法におけるエッジの長さ

6. 評価実験

評価実験では、提案手法によるレイアウト結果が美的基準に沿っているかを目視で確認した。また、映像合成処理のノードグラフを記述するのにかかる時間のうち、ノードグラフのレイアウトを整える時間を節約したいため、実行時間が数秒程度であるかどうかを確認した。

C++とSiv3Dにより映像制作ソフトDavinci Resolveのノードエディタを模したシステムを作成し、そのシステム上で評価実験を行った。ノード数10~35、エッジ数11~36のノードグラフを以下の文献やWebサイトから16個集め評価を行った。

- Fusion VFX 入門 Davinci Resolve15[17] (7個)
- NsFarm@クリエイター兼エンジニアのページ[18] (8個)
- DaVinci Resolve カンタン! クロマキー合成のススメ | Vook(ブック)[19] (1個)

6.1. レイアウト結果と考察

実験の結果、提案手法ではノードグラフを映像合成処理の流れを意識したレイアウトにすることができると分かった。しかし、ノードグラフのノードの初期配置によっては、提案手法でのレイアウト後にむしろレイアウトが悪化する場合があることが分かった。

図19はグリーンバック素材の合成を表すノードグラフ、[19]、図20は図19を力指向アルゴリズムでレイアウトした結果である。また、図21は図20をグリッドレイアウトアルゴリズムでレイアウトした結果である。図19、図20では、基本のエッジ(映像の処理の順序を表す)を赤枠、データの合流を表すエッジを緑の点線の枠で囲っている。

図19では、エッジの向きがエッジの役割ごとに決まっておらず、処理の順序や、どの時点で何のデータが合流するのかが分かりにくくなっている。

それに対し図20では、映像の処理の順序を表すエッジは左から右へ、またデータの合流を表すエッジは上から下を向くように整理できている。また従来のマグネティック・スプリング・モデルでは、一つのノードから同じ向きのエッジが出ていた場合にエッジ間が狭まりノードとエッジが重なることがあったが、図20ではそのようなことが起きていない。

また図21では、ノードグラフの構造をほぼ保ったままグリッドに揃えることができています。

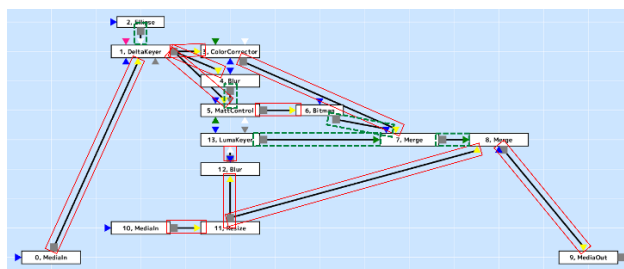


図19. グリーンバック素材の合成を表すノードグラフ[19]

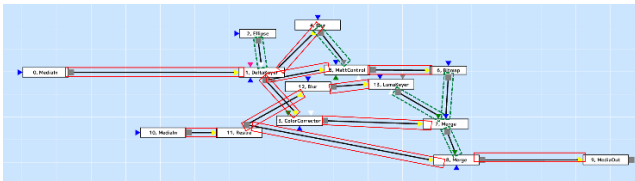


図 20. 図 19 に力指向アルゴリズムを適用した結果

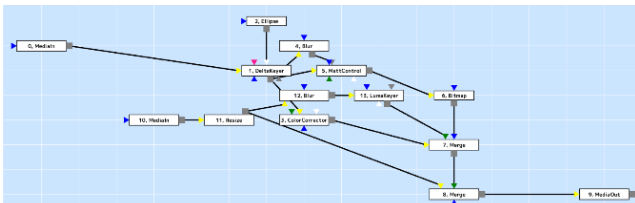


図 21. 図 20 にグリッドレイアウトアルゴリズムを適用した結果

図 22 は Fusion VFX 入門のレッスン 6 に掲載されていたノードグラフの配置を意図的に崩したものであり、図 23 はそれを力指向アルゴリズムでレイアウトした結果である。図 22 で配置を崩した理由は、レッスン 6 に掲載されていたノードグラフではすでに大半のノードの並ぶ向きが揃っていたため、レイアウト前と後で提案手法の効果を測れないからである。

図 22、図 23 では同じノードから異なる方向を向くエッジが出力されているが、この場合も対応できている。図 22、図 23 では、基本のエッジ（映像の処理の順序を表す）を赤枠、データの合流を表すエッジを緑の点線の枠で囲っている。

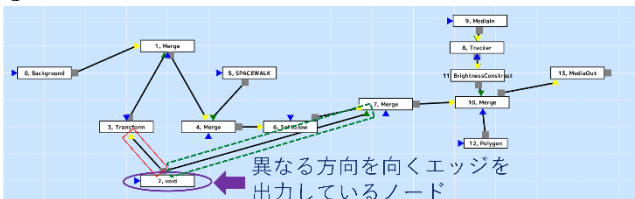


図 22. Fusion VFX 入門のレッスン 6 の配置を崩したノードグラフ[17]

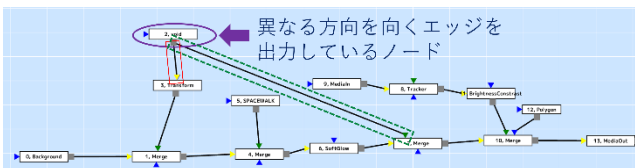


図 23. 図 22 を力指向アルゴリズムでレイアウトした結果
しかし力指向アルゴリズムでは、エッジの初期長が長かった場合にエッジ同士やノードとエッジ同士の交差がある結果になることが 16 例中 7 例あり、また 6 例中 4 例は著しくレイアウトが崩れていた。

例えば、図 25 のノードグラフは図 24 を力指向アルゴリズムでレイアウトした結果である。図 25 ではエッジの役割ごとにエッジの向きが整理されているが、図 24 には存在しなかったノードやエッジ同士の交差が発生している。

これは、エッジの初期長が長かった場合や、1 つのノードに多数のノードが接続されている場合に発生する。提案手法ではノードの向きを「基本のエッジの向き」と「データの合流を表すエッジの向き」の 2 種類に限定したために、エッジの数が増えた場合に何本かのエッジを別方向に向けてエッジの密集を避けることができないのが原因である。また、提案手法ではエッジの初期長を尊重するため、エッジが長かった場合に短くできないのも原因である。

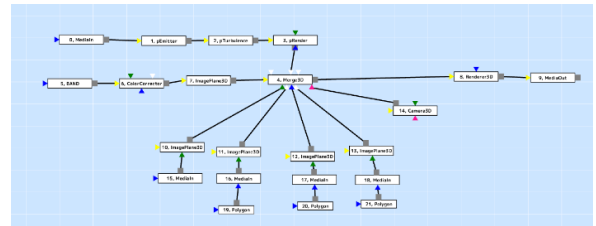


図 24. Fusion VFX 入門のレッスン 11 のノードグラフ[17]

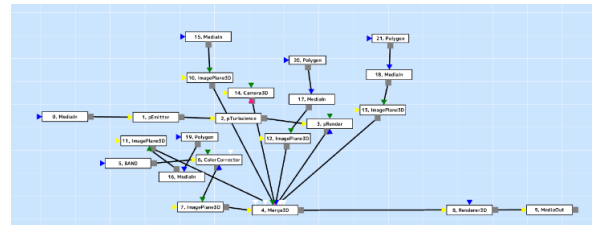


図 25. 図 24 を力指向アルゴリズムでレイアウトした結果

また、グリッドレイアウトアルゴリズムでは実行後に新たにエッジとノード同士の交差が発生することが 16 例中 7 例、実行後にノードグラフの形が崩れることが 16 例中 4 例あった。

図 26 はテキストアニメーションの処理を表すノードグラフ[20]を力指向アルゴリズムでレイアウトしたものである。また、図 27 は図 26 をグリッドレイアウトアルゴリズムでレイアウトしたものである。図 27 では、図 26 でのレイアウトを大幅に崩している。ハイブリッド・グリッド・レイアウトアルゴリズム [16]とは違い提案手法では、どのノードからグリッドに沿わせれば全体のノードの移動量が小さくなるのかを計算せず、左にあるノードから順にグリッドに沿わせている。それが原因で、ノードをグリッドに沿わせる際に全体のノードの移動量が非常に大きくなり、レイアウトが崩れる場合が起きている可能性がある。

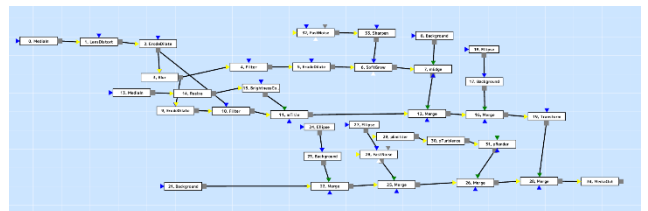


図 26. テキストアニメーションを表すノードグラフ[20]

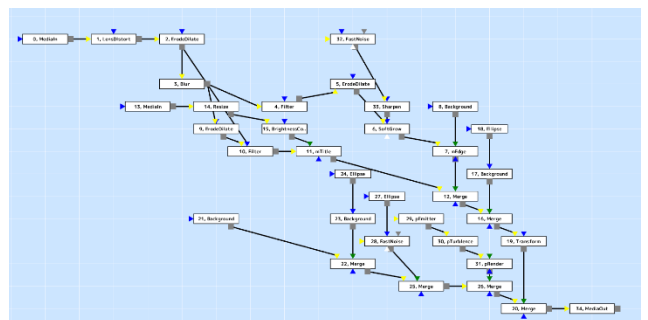


図 27. 図 26 をグリッドレイアウトアルゴリズムでレイアウトした結果

6.2. 実行時間

提案手法を各ノードグラフで10回実行し実行時間を測り、その平均値をとった。実行環境は以下の通りである。

- CPU : Intel(R) Core(TM) i7-1260P
- メモリ : DDR5 (16GB)
- OS : Windows11 Home

実験の結果、力指向アルゴリズムでは図28のノードグラフの時に16例中最も時間がかかり、その時間は約3秒であった(ノード数35, エッジ数36)。またグリッドレイアウトアルゴリズムでは、どのノードグラフでも1秒未満であった。そのため、映像合成処理のノードグラフを記述するのにかかる時間のうち、ノードグラフのレイアウトを整える時間を節約できる。

7. まとめ

本研究では、映像制作の効率化のために、映像合成処理に適したノードグラフを自動でレイアウトするグラフ描画アルゴリズムを開発した。また、そのために映像合成処理に適したノードグラフにおける美的基準を定めた。

提案手法では、ノードグラフを映像合成処理の流れを意識したレイアウトにすることができると分かった。しかし、ノードグラフの初期配置や構造によっては、定めた美的基準を崩すレイアウト結果になる場合があることが分かった。とくに、「エッジの交差数を最小にする」「ノードとエッジが交差しないようにする」の基準が崩れた。

今後の課題は3つある。

1 つ目は、今回定めた美的基準である「エッジが特定の方向に従う」と「エッジの長さが初期配置と比べ変化しない」が原因でレイアウトが崩れたことがあったので、これらの基準を見直す必要がある。

2 つ目は、文献などで収集したノードグラフは配置が整っている場合が多いため、提案手法を適切に評価するためには、ユーザへのアンケート調査などが必要である。

3 つ目は「エッジを整える向きをどの方向にするか」などの、美的基準のうち細かい部分はユーザが決めることができるようにすることである。今回は過去の文献を参考に美的基準を決めたが、研究科内では、見やすいノードグラフの配置はユーザごとに個人差があるという指摘も受けている。

参 考 文 献

- [1] “Blueprint Assist : コードプラグイン - UE マーケットプレイス”, Epic Games, Inc, 2019 年 6 月 19 日 公開, <https://www.unrealengine.com/marketplace/ja/product/blueprint-assist>, 2023 年 7 月 16 日にアクセス
- [2] 居出貴之, 今井敏行, 床井浩平, “映像合成処理に適したノードグラフの描画アルゴリズムの構築と実装”, 映像情報メディア学会技術報告= ITE technical report, 47 (9), 69-72, (2023)
- [3] 大宮拓馬, 小形真平, 岡野浩三, “可読性の高いクラス図レイアウトを作成するための美的基準の調査”, 38, 4, pp.4_33-4_39, (2021)
- [4] W. Huang, P. Eades, S.-H. Hong and C.-C. Lin, "Improving multiple aesthetics produces better graph drawings," Journal of Visual Languages & Computing, 24, .4, pp. 262-272, (2013)
- [5] 鈴木和彦, 鎌田富久, 榎本彦衛, “単純無向グラフ自動描画アルゴリズム”, コンピュータ ソフトウェア, 12, 4, pp.4_335-4_345, (1995)
- [6] 小町直, “Fusion 挫折する理由はレイヤー構造じゃないから！？ ~挫折してからが本当のスタートです~ | Vook(ヴック)”, Vook, 2023 年 3 月 20 日 更新, <https://vook.vc/n/4135>, 2023 年 7 月 17 日にアクセス
- [7] M. Soegaard, “グリッドシステムで強固なデザインレイアウトを作る | UX MILK””, ポップインサイトカンパニー (社名: 株式会社メンバーズ)), 2020 年 7 月 31 日 更新, <https://uxmilk.jp/90133>, 2023 年 7 月 17 日にアクセス
- [8] “ノードをグリッドにスナップ|ノードツリーの整頓”, DaVinci Resolve 18 リファレンスマニュアル, Blackmagic Design, p. 1220
- [9] “Nuke-NodeGraph にグリッドを表示させる - Born Digital サポート”, Born Digital, 2017 年 07 月 12 日 更新, <https://support.borndigital.co.jp/hc/ja/articles/115010373448-Nuke-NodeGraphにグリッドを表示させる>, 2023 年 7 月 17 日にアクセス
- [10] ダミアン・アレントニー・ギャルド, ディオン・スコペトゥオロ, Fusion VFX 入門 DaVinci Resolve 15, Blackmagic Design, (2019)
- [11] 三末和男, 杉山公造, “マグネティック・スプリング・モデルによるグラフ描画法について”, 情報処理学会研究報告ヒューマンコンピュータインタラクション (HCI), 94, 60(1994-HI-055), pp.17-24, (1994)
- [12] 尾上洋介, “階層グラフの可視化”, オペレーションズ・リサーチ, 63, 1, pp.20-26, ii, (2018)
- [13] Leu Thomas, “Scalable Drawing of Nested Directed Acyclic Graphs With Gates and Ports”, Master's thesis, ETH Zurich, Department of Computer Science, (2021)
- [14] 野口隆佳, 西川博昭, “インタラクティブ性を考慮したオブジェクト図の自動レイアウト手法”, 修士論文, 筑波大学大学院, 2000.
- [15] T. Dwyer, Y. Koren, “Dig-CoLa: directed graph layout through constrained energy minimization”, IEEE Symposium on Information Visualization, 2005. INFOVIS 2005, pp. 65-72, IEEE, 2005
- [16] Kentaro Inoue, Shinichi Shimozono, Hideaki Yoshida, Hiroyuki Kurata, Application of Approximate Pattern Matching in Two Dimensional Spaces to Grid Layout for Biochemical Network Maps, Plos one, 7, 6, e37739, (2012)
- [17] ダミアン・アレントニー・ギャルド, ディオン・スコペトゥオロ, Fusion VFX 入門 DaVinci Resolve 15, Blackmagic Design, 2019
- [18] NsFarm, “NsFarm@クリエイター兼エンジニアのページ”, <https://www.nsfarm.life/index.html>, 2023 年 7 月 16 日にアクセス
- [19] 小町直, “DaVinci Resolve でクロマキー合成する2つの方法【認定トレーナーが解説！】 | Vook(ヴック)”, Vook, 2023 年 3 月 20 日 更新, <https://vook.vc/n/4102>, 2023 年 7 月 16 日にアクセス
- [20] NsFarm, “【後編】Cinematic に音と映像を合わせるコツとスモーク、パーティクル、ライティングを追加する方法 in Fusion | NsFarm”, <https://www.nsfarm.life/page2019/acl2019BlogDr16CinematicTitleDesney2.html>, 2023 年 7 月 16 日にアクセス