生活支援ロボットによる物体配置タスクにおける Transformer PonNet に基づく危険性予測および可視化

Collision Risk Prediction and Visualization Based on Transformer PonNet in Object Placement Tasks by Domestic Service Robots

植田 有咲 *1 Arisa Ueda
Aly Magassouba *2 Aly Magassouba
平川 翼 *3 Tubasa Hirakawa
山下 隆義 *3 Takayoshi Yamashita
藤吉 弘亘 *3 Hironobu Fujiyoshi
杉浦 孔明 *1 Komei Sugiura

*1 慶應義塾大学 Keio University
*2 国立研究開発法人情報通信研究機構 National Institute of Information and Communications Technology
*3 中部大学 Chubu University

Placing everyday objects in designated areas, such as placing a glass on a table, is a crucial task for Domestic service robots (DSRs). In this paper, we propose a physical reasoning method about collisions in placement tasks. The proposed method, Transformer PonNet, predicts the probability of a possible collision and visualizes areas involved in the collision. Unlike existing methods, Transformer PonNet can be applied to objects whose models are unavailable. We propose a novel Transformer Perception Branch that handles relationships among features more complex than simple self-attention. We built simulation and physical datasets using a DSR, and validated our method on the datasets. We obtained an accuracy of 82.5% for the physical dataset.

1. はじめに

高齢化社会では, 人手不足が深刻な問題となっており, 生活支援ロボットは高齢者や障害者を支える上で有望視されている. 机の上にコップを置くなど, 日常の物を指定された場所に置くことは, 生活支援ロボットにとって必要不可欠である. このような背景から, 本論文では物体配置タスクでの衝突に関する physical reasoning に焦点を当てている. 本タスクは対象物体を雑然とした領域に配置した場合に起こり得る物体間の一連の物理的相互作用の予測を行う必要があり, 難しいタスクである.

本論文では, 静止画像から衝突可能性を推定し, 衝突に関連する部分の可視化を行う physical reasoning 手法, Transformer PonNet の提案を行う. Transformer PonNet は, Feature Extractor, Target Embedder, Attention Branch および Transformer Perception Branch の 4 つのモジュールで構成されている.

既存手法 [Magassouba 21] では, 入力として, 配置される対象物体の詳細な大きさが必要であり, 汎用性が低かった. 対照的に, Transformer PonNet では, Target Embedder を導入することで, 正確な大きさが分からない対象物体に対しても適用可能である.

本論文の貢献は以下の通りである.

- 対象物体の特徴量を抽出するための Target Embedder を導入することで, 対象物体の 3 次元モデルを考慮する必要がなくなった.
- 既存手法と違い, 配置場所を中央だけでなく, 9 領域を対象とした.
- 特徴量間の複雑な関係を扱うことができる Transformer Perception Branch を導入した.
- シミュレーション環境, 実機環境の両方で検証を行なった.

連絡先: 植田有咲, 慶應義塾大学, 神奈川県横浜市港北区日吉 3-14-1, arinko31@keio.jp

2. 問題設定

本論文では, physical reasoning in object placing (PROP) タスクに焦点を当てている. このタスクでは, 生活支援ロボットが日常物体を指定された領域に配置する際の衝突可能性を予測する必要がある. 画像から衝突可能性を予測し, 画像内で衝突に関連する部分を可視化することを目標としている. PROP タスクの入出力は次のように定義する.

入力: 対象物体と配置場所の RGBD 画像
出力: 衝突可能性

本論文で使用する用語を次のように定義する.

対象物体: 配置予定の物体
配置場所: ロボットが対象物体を置く予定の場所
障害物: 配置場所に既に置いてある物体
衝突: 相対速度が閾値よりも大きい危険な接触.
接触: 相対速度が閾値よりも小さい軽微な接触.

生活支援ロボットが対象物体を配置する際, 必ず対象物体と机の間で「接触」が起こる. このように, PROP タスクには常に少なくとも 1 つの衝突または接触が存在することに注意する必要がある.

本論文では, ロボットの配置動作などは固定されることを前提とする. したがって、ロボットの制御および計画タスクは扱わず, physical reasoning のみに焦点を当てる. また, ロボットは配置場所の前にあることを想定している.

実機ロボットによるデータ収集は手作業で行う必要があり, 非常に時間がかかる. そのため, 我々は効率的にデータの収集ができるシミュレーションデータを用いてモデルの学習を行なった. それらの学習モデルを用いて実機データセットの評価を行なった.

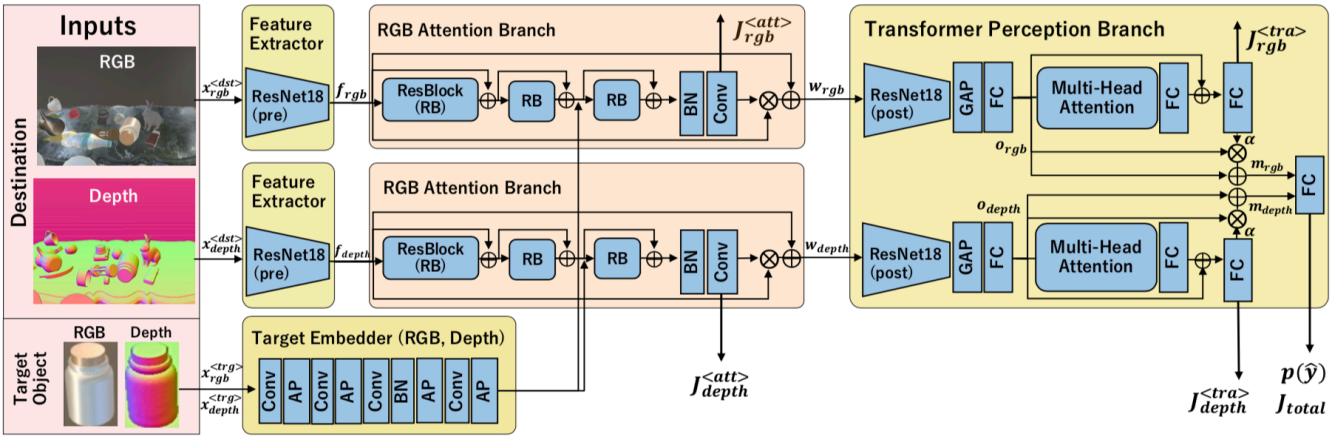


図 1: Transformer PonNet のネットワーク図

3. 提案手法

3.1 Transformer PonNet

図 1 に提案手法のネットワーク構造を示す．提案手法は大きく分けて 4 つ (Feature Extractor, Target Embedder, Attention Branch および Transformer Perception Branch) に分かれる．図 1 の Conv, BN, GAP, AP および FC はそれぞれ畳み込み層，バッチ正規化層，global average pooling 層，average pooling 層および全結合層を表す．Transformer PonNet の入力は次に示す通りである．

$$\boldsymbol{x}(i) = \{\boldsymbol{x}_{depth}^{<dst>}(i), \boldsymbol{x}_{rgb}^{<dst>}(i), \boldsymbol{x}_{depth}^{<trg>}(i), \boldsymbol{x}_{rgb}^{<trg>}(i)\} \quad (1)$$

$\boldsymbol{x}_{depth}^{<dst>}(i)$ および $\boldsymbol{x}_{rgb}^{<dst>}(i)$ はそれぞれ配置場所の depth 画像および RGB 画像を表す．$\boldsymbol{x}_{depth}^{<trg>}(i)$ および $\boldsymbol{x}_{rgb}^{<trg>}(i)$ は対象物体の depth 画像および RGB 画像を表す．生活支援ロボットが対象物体を把持するときに，既にこれらの入力は得られているものとする．入力画像は標準化された後，224×224 の大きさに処理される．depth 画像については面法線 [Aakerberg 17] に基づいてカラー画像へと変換される．

3.1.1 Feature Extractor および Target Embedder

Feature Extractor は ResNet18 の stage4-unit1-bn1 層までで構成される．入力は $\boldsymbol{x}_{depth}^{<dst>}(i)$ および $\boldsymbol{x}_{rgb}^{<dst>}(i)$ である．

Target Embedder は $\boldsymbol{x}_{depth}^{<trg>}(i)$ および $\boldsymbol{x}_{rgb}^{<trg>}(i)$ を入力とし，対象物体画像の特徴量を抽出する．複数の畳み込み層，average pooling 層 およびバッチ正規化層から構成されている．

3.1.2 Attention Branch

我々のモデルは RGB と depth の 2 つの Attention Branch を持ち，Attention Branch はバッチ正規化層，畳み込み層およびシグモイド活性化関数で構成されている．Attention Branch の入力 $\boldsymbol{f}_{rgb}$ および $\boldsymbol{f}_{depth}$ は Feature Extractor からの出力として得られる．これらの特徴量 map は ResBlock およびシグモイド関数に入力される．その後，attention map $\boldsymbol{a}_{rgb}$ および $\boldsymbol{a}_{depth}$ が得られる．重み付き特徴量 map は次のように定義する．

$$\boldsymbol{w}_{rgb} = (1 + \boldsymbol{a}_{rgb}) \odot \boldsymbol{f}_{rgb} \quad (2)$$

$$\boldsymbol{w}_{depth} = (1 + \boldsymbol{a}_{depth}) \odot \boldsymbol{f}_{depth}, \quad (3)$$

ここに $\odot$ はアダマール積を表す．Attention 機構は，衝突可能性の予測に関連する部分を可視化することが期待できる．Attention Branch からの出力は、Transformer Perception Branch の ResNet18 の後半部分に入力される．

3.1.3 Transformer Perception Branch

Transformer Perception Branch は ResNet18 の後半と Transformer Encoder から構成され，RGB と depth の Attention Branch からの出力を結合し，最終出力として衝突可能性を予測する．$\boldsymbol{w}_{rgb}$ および $\boldsymbol{w}_{depth}$ は ResNet18 の後半に入力され，その後，全結合層から特徴量 $\boldsymbol{o}_{rgb}$ および $\boldsymbol{o}_{depth}$ が得られる．これらは Transformer Branch に入力される．query $\boldsymbol{Q}^{(i)}$, key $\boldsymbol{K}^{(i)}$ および value $\boldsymbol{V}^{(i)}$ は次のように定義される．

$$\boldsymbol{Q}^{(i)} = W_q^{(i)} \boldsymbol{o}_k^{(i)} \quad (4)$$

$$\boldsymbol{K}^{(i)} = W_k^{(i)} \boldsymbol{o}_k^{(i)} \quad (5)$$

$$\boldsymbol{V}^{(i)} = W_v^{(i)} \boldsymbol{o}_k^{(i)} \quad (6)$$

ここに，$k \in \{rgb, depth\}$ であり，attention Ω は次のように定義される．

$$\boldsymbol{\omega}_k^{(i)} = \boldsymbol{V}^{(i)} \mathrm{softmax}\left(\frac{\boldsymbol{Q}^{(i)} \boldsymbol{K}^{(i)\top}}{\sqrt{d_k}}\right) \quad (7)$$

$$\sqrt{d_k} = \frac{H}{A} \quad (8)$$

$$\Omega = \{\boldsymbol{\omega}_k^{(1)}, ..., \boldsymbol{\omega}_k^{(A)}\}, \quad (9)$$

H は入力 $\boldsymbol{o}_k$ の次元数，A は attention のヘッド数を表す．Attention Ω は 2 つの全結合層に入力され，それぞれ線形変換され，出力として $\boldsymbol{\alpha}$ が得られる．Transformer Branch の出力 $\boldsymbol{m}_k$ を次のように定義する．

$$\boldsymbol{m}_k = (1 + \boldsymbol{\alpha}) \odot \boldsymbol{Q}^{(i)}, \quad (10)$$

$\boldsymbol{m}_{rgb}$ および $\boldsymbol{m}_{depth}$ をチャネル方向に結合し，全結合層に入力し，最終出力として $p(\hat{y})$ が得られる．

3.1.4 損失関数

我々は次に示す損失関数 $L(\hat{y})$ を用いる．

$$L(\hat{y}) = \lambda_{rgb}^{<att>} J_{rgb}^{<att>} + \lambda_{depth}^{<att>} J_{depth}^{<att>} + \lambda_{rgb}^{<tra>} J_{rgb}^{<tra>} + \lambda_{depth}^{<tra>} J_{depth}^{<tra>} + \lambda_{total} J_{total}, \quad (11)$$

ここに，$\lambda_{rgb}^{<att>}$, $\lambda_{depth}^{<att>}$, $\lambda_{rgb}^{<tra>}$, $\lambda_{depth}^{<tra>}$ および λ_{total} はそれぞれ RGB Attention Branch からの出力，depth Attention Branch からの出力，RGB Transformer Branch からの出力，depth Transformer Branch からの出力および 最終出力の損失関数の重みを表す．$J_{rgb}^{<att>}$, $J_{depth}^{<att>}$, $J_{rgb}^{<tra>}$, $J_{depth}^{<tra>}$ および J_{total} は交差エントロピー誤差である．

4. 実験

4.1 データセット

既存の標準データセットのほとんどはロボットや日常環境を対象としていないため, 独自のデータセットを作成した. 次に示す 3 つのデータセットを実験に用いた.

4.1.1 PonNet-A-Sim dataset

[Magassouba 21] で使用されているシミュレーションデータセットである. 生活支援ロボットは, 障害物が無作為に配置された配置場所に対象物体を配置した. 各サンプルには「衝突」または「接触」のラベルが付けられている.

4.1.2 PonNet-B-Sim dataset

シミュレーション環境で新たに収集したデータセットである. 生活支援ロボットは, 無作為に選択された対象物体を中央および周辺領域に配置した. その他の条件は, PonNet-A-dataset の条件と同じである. 高さや形の異なる 5 種類の家具, 明るさや背景の異なる 5 種類の場面を使用した.

4.1.3 PonNet-A-Real dataset

実機環境で新たに収集したデータセットである. 図 2 は, このデータセットで使用した環境と物体を示している. 生活支援ロボットはトヨタ自動車株式会社の Human Support Robot (HSR) [Yamamoto 19] を用いた. まず, 選択した障害物を無作為に机の上に配置した. 生活支援ロボットは配置場所の RGBD 画像を撮影し, 対象物体を配置した. その後, 手動で衝突の有無を記録した. このデータセットには 200 サンプルが含まれており, テストセットとしてのみ使用される. 実験環境の広さは 1.5 [m]×1.5 [m] とした. 配置場所として, World Robot Summit （WRS）2020 [WRS 20] で標準家具に指定されている机を使用した. 机は長さ 0.4 [m], 幅 0.4 [m], 高さ 0.6 [m] とした. HSR とテーブルの間の距離は約 0.3 [m] とした.

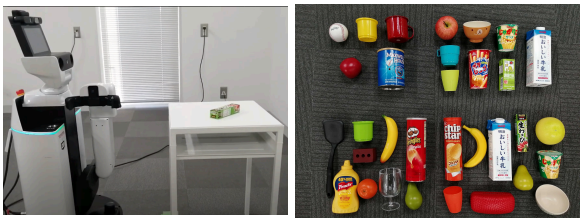
図 2(b) にこのデータセットで使用した対象物体と障害物を示す. これらの物体は訓練集合に含まれていない未知物体である. 左右の物体群はそれぞれ 18 個の標準的な物体と 14 個の一般的な物体である. 標準物体は, WRS2020 で標準物体として指定された YCB オブジェクト [Calli 15] から選択した. 上部と下部の物体群は, それぞれ対象物体と障害物である. 生活支援ロボットが扱う可能性がある日常物体に加え, 透明な物体など難しい物体を選択した. HSR が把持できない物体は対象物体に含めなかった. 以下, PonNet-A-Sim データセット, PonNet-B-Sim データセット, および PonNet-A-Real データセットは, 簡略化のため, それぞれ A-Sim, B-Sim, および Real データセットと呼ぶこととする.

4.1.4 データセットのラベル付けと統計情報

A-Sim と B-Sim では自動的にラベル付けを行なった. これらのシミュレーションデータセットでは, 各サンプルに「衝突」または「接触」のラベルが付けられている. 衝突判定の閾値は

表 1: 各データセットの統計情報

Dataset	Label	Train	Valid	Test
PonNet A-Sim	Collision	4807	492	448
	Contact	5074	496	490
PonNet B-Sim	Collision	4652	612	596
	Contact	6148	738	754
PonNet A-Real	Collision	-	-	126
	Contact	-	-	74



(a) (b)

図 2: 実機実験環境: (a) HSR と実験環境, (b) 使用した障害物と対象物体

表 2: ハイパパラメータ設定

Optimizer	Adam (learning rate = 0.0003)
Network input size	[224 × 224 × 3]
Backbone CNN	ResNet18
Batch size	64
Attention Branch	Input : [14 × 14 × 512]
	Conv.layer : [14 × 14 × 2]
	Att.layer : [14 × 14 × 1]
	Output : [14 × 14 × 256]
Transformer Perception Branch	Input : [14 × 14 × 1]
	FC: 256, 2
loss weights	$\lambda_{rgb}^{<att>}$= 1, $\lambda_{depth}^{<att>}$= 1, λ_{total}= 1, $\lambda_{rgb}^{<tra>}$= 0.5, $\lambda_{depth}^{<tra>}$= 0.5
epochs	50

$V_c = 0.1$ [m/s] とした. 相対速度 $|v|$ がこの閾値 V_c を超えた場合, 「衝突」とし, それ以外の場合は「接触」とした. Real でのラベル付けは手動で行なった. 表 1 は, A-Sim, B-Sim, および Real データセットでの統計情報を示している.

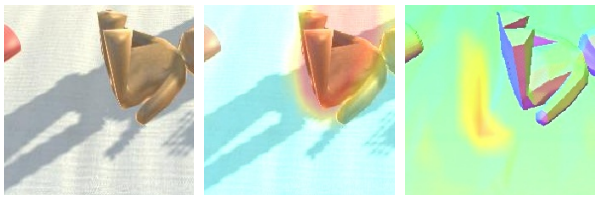
4.2 実験設定

表 2 に実験設定を示す. モデルの入力次元は 224 × 224 × 3 である. 最適化手法は Adam を使用した. 我々の手法は約 2600 万のパラメータがある. 学習にはメモリ 11GB 搭載の GeForce RTX 2080 および Intel Core i9-9900K を使用した. 学習に要した時間は, 1 時間であった. 検証集合において最も高い精度を記録したときのテスト集合における精度を, 最終的な学習の精度とした.

表 3: 各データセットでの定量的結果

	Accuracy [%]		
Method	Train:A-Sim Test:A-Sim	Train:B-Sim Test:B-Sim	Train:A-Sim Test:Real
Plane detection	82.5	-	-
PonNet (baseline)	90.94±0.22	-	-
Ours (type1)	86.57±3.34	72.52±2.97	72.70±13.90
Ours (type2)	90.64±0.21	79.94±1.75	81.70±1.29
Ours (type3)	91.13±0.76	**83.64±1.39**	80.80±3.20
Ours (full)	**91.19±0.35**	80.47±0.48	**82.50±2.85**

(a)



(b)

図 3: 定性的結果: 左から元の RGB 画像, RGB の attention による可視化画像, depth の attention による可視化画像

4.3 定量的結果

Transformer PonNet を, PonNet [Magassouba 21] と従来の RGBD 平面検出アルゴリズム [Wang 18][*1] の 2 つのベースライン手法と比較した. PonNet をベースラインとした理由は, 提案手法で新しく導入した部分を同じデータセットで比較および評価することができるからである. 評価尺度としては精度を用いた. 表 3 に定量的結果を示す. 表の 2 列目, 3 列目, および 4 列目は, それぞれ A-Sim, B-Sim, および Real の精度を示している. A-Sim の評価では, 最も良い精度は提案手法で得られ, 平面検出法と PonNet と比較してそれぞれ約 8.6 %および 0.2 %精度が向上し, 平均 91.19 %という精度が得られた. PonNet は, 対象物体の正確な大きさが与えられるため, 我々の手法に比べて有利であるが, 提案手法が上回る結果を得たことから, Target Embedder を導入することは精度向上に寄与すると考えられる. Real での評価でも, Transformer PonNet で最も良い精度が得られ, 82.50 %という結果だった. この結果から, シミュレーションデータセットで学習したモデルが実機環境に対しても十分適応可能であると言える.

4.4 定性的結果

図 3 は Transformer PonNet から得られた attention map の可視化の例である. 図 3(a) は A-sim での True Positive の例を表している. RGB では配置場所にある障害物のおもちゃに正しく注目できている. depth 画像では, 対象物体を安全に配置できる領域に注目できている. 図 3(b) は Real での True Negative の例である. 実機データセットでも, RGB では障害物に正しく注目し, depth では安全に配置できる領域に注目していることがわかる.

4.5 Ablation Studies

Ablation Study は以下の 3 つの条件で行なった.

1. Ours (type1): ResNet18 の前半を Target Embedder とした提案モデル. Transformer Perception Branch の代わりに単純な self-attention の Perception Branch を使用した.
2. Ours (type2): Transformer Perception Branch を使用した提案モデル. 式 (4), (5), (6) において, RGB Transformer Branch では $\boldsymbol{Q} = W_q \boldsymbol{o}_{rgb}$, $\boldsymbol{K} = W_k \boldsymbol{o}_{depth}$, $\boldsymbol{V} = W_v \boldsymbol{o}_{depth}$ とした. 同様に, depth Transformer Branch では $\boldsymbol{Q} = W_q \boldsymbol{o}_{depth}$, $\boldsymbol{K} = W_k \boldsymbol{o}_{rgb}$, $\boldsymbol{V} = W_v \boldsymbol{o}_{rgb}$ とした.
3. Ours(type3): Transformer Perception Branch の代わりに Perception Branch で単純な self-attention を使用した提案モデル.

A-Sim では, 提案手法 (type3) と Transformer PonNet で, ベースラインをわずかに上回る結果が得られた. この結果から, Target Embedder の導入は精度向上に寄与していると考えられる. Target Embedder の構造について, 提案手法 (type1) と Transformer PonNet の比較から ResNet18 などのネットワークの代わりに畳み込みニューラルネットワークを使用した場合の方が精度が良いことが分かった. また, 提案手法 (type2) と Transformer PonNet の結果を比較すると, Transformer を Perception Branch に導入すると精度が向上することが分かった. 対象物体情報を画像として入力できないため, ベースライン手法は B-Sim および Real データセット評価しなかった. B-Sim の評価では, 提案手法 (type3) は Transformer PonNet よりも良い精度が得られた. 単純な構造の Perception Branch の方が 9 領域を対象としたデータセットでは, 精度が良くなることが分かった.

*1 https://github.com/chaowang15/RGBDPlaneDetection

5. おわりに

本論文では, PROP タスクのための Transformer PonNet を提案した. 我々の貢献は以下の通りである.

- 対象物体の特徴量抽出のために, Target Embedder を導入した. これにより, 対象物体の 3D モデルを想定する必要がなくなった.
- 提案手法は, 複数の配置領域を対象としている. PonNet とは異なり, 配置領域は画像の中心に限定されていない.
- 単純な self-attention よりも複雑な特徴量間の関係を処理可能な新たな Transformer Perception Branch を提案した.
- 生活支援ロボットと未知物体を用いて実機データセットを構築し, 82.5 %の精度が得られた.

参考文献

[Aakerberg 17] Aakerberg, A., Nasrollahi, K., Rasmussen, C. B., and Moeslund, T. B.: Depth Value Pre-Processing for Accurate Transfer Learning based RGB-D Object Recognition., in *IJCCI*, pp. 121–128 (2017)

[Calli 15] Calli, B., Walsman, A., Singh, A., Srinivasa, S., Abbeel, P., and Dollar, A. M.: Benchmarking in manipulation research: The YCB object and model set and benchmarking protocols, *arXiv preprint arXiv:1502.03143* (2015)

[Magassouba 21] Magassouba, A., Sugiura, K., Nakayama, A., Hirakawa, T., Yamashita, T., Fujiyoshi, H., and Kawai, H.: Predicting and Attending to Damaging Collisions for Placing Everyday Objects in Photo-Realistic Simulations, *arXiv preprint arXiv:2102.06507* (2021)

[Wang 18] Wang, C. and Guo, X.: Plane-based optimization of geometry and texture for RGB-D reconstruction of indoor scenes, in *2018 International Conference on 3D Vision (3DV)*, pp. 533–541IEEE (2018)

[WRS 20] World Robot Summit 2020 Partner robot challenge Real Space Rules Regulations, https://worldrobotsummit.org/wrs2020/challenge/download/Rules/DetailedRules_Partner_EN.pdf (2020), [Online; accessed 23-Jan-2021]

[Yamamoto 19] Yamamoto, T., Terada, K., Ochiai, A., Saito, F., Asahara, Y., and Murase, K.: Development of Human Support Robot as the research platform of a domestic mobile manipulator, *ROBOMECH journal*, Vol. 6, No. 1, p. 4 (2019)