動的センサフュージョンを実現するソフトウェアアーキテクチャに関する研究

M2023SE004 近藤友樹
指導教員：野呂昌満

1　はじめに

自動運転システムにおいて，物体検知の精度を高めるためにセンサフュージョン技術が必要である．実際の走行環境では，悪天候や夜間に特定のセンサの性能が低下する問題がある．センサフュージョン技術は異なるセンシングデータを統合し，多様な環境に対応可能な物体検知を実現する有効な手法である．センサフュージョンの研究 [2][3] は，単一の物体検知方式のみを利用した静的な物体検知を行っている．用いる方式が苦手な環境に陥ったとき，求められる精度や実行時間の性能を保証できないという問題がある．物体検知の精度向上のために多くの研究 [1][2][3] が行われている．その結果，新たな物体検知器とセンサが開発されている．今後このことは加速度的に増大すると考えられる．このことから本研究の目的は，変化する環境に適応し，技術進化を考慮した物体検知システムのアーキテクチャ提案である．

2　関連研究

2.1　センサフュージョン技術

現在提案されているセンサフュージョンの方法は Early-fusion[2]，Late-fusion[2]，Joint-fusion[3] である．Early-fusion とは取得したデータのまま統合する方式である．実行速度が優れているが，統合するときに多くのデータを損失してしまう．Late-fusion は，別々に物体検知を行い生成されたモデル同士を統合する方式である．高精度だが，物体検知までの処理量が多いことから，センサ同士の時間同期が難しいという問題がある．Joint-fusion は，センシングデータからニューラルネットワークを用いて特徴量を抽出し，抽出した特徴量同士を統合する方式である．他の方式と比べて，センサの特性を効率よく物体検知結果に反映させることができる．

2.2　動的な振る舞い

水谷ら [5] は偽造画像を検出するためのアーキテクチャを提案している．この提案は造画像を検出するニューラルネットワークを動的に選択する構造を持つ．偽造検知結果をコンテキストとしてその情報をもとに NN_selector がニューラルネットを選択する構造を持つ．この構造を物体検知方式の切り替えへ応用する．

3　研究課題

研究目的と関連研究から以下の研究課題を定める．

1. センサフュージョンを動的に行うための要件定義
   システムの概要と特性を定義する．
2. 動的センサフュージョンを行うアーキテクチャの設計
   要件定義をもとに設計を行う．
3. アーキテクチャ妥当性の議論
   定性的な議論から考察と評価をする．

4　課題解決へのアプローチ

複数の物体検知器を組み合わせることで、それぞれの検知器が得意とする環境や状況によって動的に切り替え，様々な環境に適応可能な物体検知を実現する．本研究で提案する動的センサフュージョンとは，多層型センサの組み合わせを静的に定義することで，物体検知器を動的に変化させる物体検知方式である．動的センサフュージョンを実現するために物体検知器を動的に切り変える構造を持ち，適切な選択を行う必要がある．本研究では，水谷ら [4] の提案を参考に，コンテキスト指向に基づきアーキテクチャで動的な振る舞いを実現する．これに加えて，振る舞いに関するデザインパターンを活用したアーキテクチャを定義する．動的センサフュージョンは，複数の物体検知器を利用することを前提にしている．利用する物体検知器によって必要なデータが異なることから，センシングデバイスと物体検知器の関係を効率的に管理することが必要である．この要求に対応するために、構造に関するデザインパターンを活用したアーキテクチャを定義する．提案したアーキテクチャが物体検知システムの開発における課題をどのように解決するかを明らかにするために，定性的な議論を行う．

5　設計

5.1　動的センサフュージョンの概要と要件定義

動的センサフュージョンの概念図を図 1 に示す．

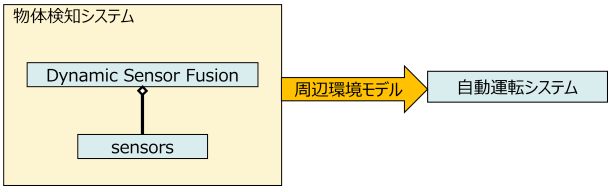

図 1　動的センサフュージョンの概念図

物体検知システムは，自動運転システムが走行判断を行うための周辺環境データを作成する．動的センサフュージョンは複数の物体検知器によって周辺環境モデルを作成する．動的センサフュージョンのシステムに必要な要件を

以下に挙げる．

・ 物体検知の動的な切り替え

・ センシングデバイスの適切な配置

・ 拡張性

5.2 パターンの適用

前節で定義した要件を満たす設計を行うためにデザインパターンを利用する．センシングデバイスの効率的な配置のために構造に関するデザインパターンを利用して設計を行った．動的な振る舞いを行うために，振る舞いに関するパターンを利用した．各パターンについて特性を分析するとともに，システムの拡張性について議論を行った．

5.2.1 Bridge パターン

Bridge パターンは抽象部分と実装部分を分離して，独立に拡張ができる構造である．このパターンを物体検知システムに適用し，センサと物体検知器それぞれに抽象部分を作ることによって，拡張性を保証できると考えた．Bridge パターンを利用した動的センサフュージョンのアーキテクチャを図 2 に示す．

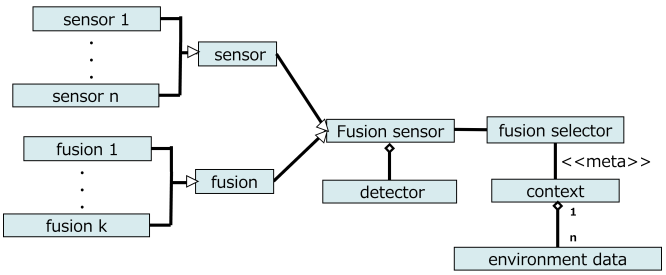

図 2 Bridge pattern

各センシングデバイスと物体検知器に抽象化クラス sensor と fusion を定義した．それぞれのサブクラスとして，カメラなどのセンサ，物体検知器が実装される．この構造によって，それぞれの変更を独立して行うことができる．物体検知器の協調を行うための Fusion sensor クラスは，どの変更を行ったときでも必ず影響を受ける．このクラスは，他のクラスに比べて変更が頻繁に起こることが予想される．

5.2.2 Composite パターン

Composite パターンは単一のオブジェクトと複数のオブジェクトの集合を同一視できるようにする構造である．この構造をセンシングデバイスの適切な配置に利用できると考えた．Composite パターンを利用した動的センサフュージョンのアーキテクチャを図 3 に示す．

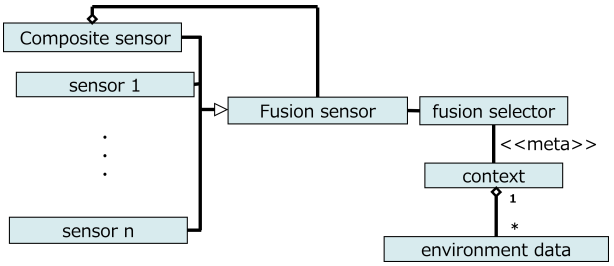

図 3 Composite pattern

各センシングデバイスが Leaf となり, Composite sensor は複数のセンシングデータの組み合わせを管理するクラスである．必要なセンシングデータの組を定義しておくことで動的に変化する必要なデータの組に対応する．この構造により，物体検知器とセンサの追加いずれの場合でも，Composite sensor クラスの変更で対応することができる．

5.2.3 Strategy パターン

Strategy パターンは，特定のアルゴリズムをカプセル化し，動的に切り替える構造を実現する設計パターンである．動的な振る舞いの実現，拡張性を持つ設計ができると考えた．Strategy パターンを利用した動的センサフュージョンのアーキテクチャを図 4 に示す．

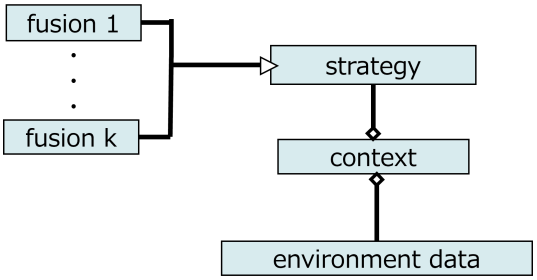

図 4 Strategy pattern

この構造はセンシングデバイスと物体検知器の組をカプセル化することから，その組み合わせが固定化される．同一センシングデータを利用する物体検知器の利用が制限されていることから，動的センサフュージョンに適さないと考えた．

5.2.4 Visitor パターン

Visitor パターンはオブジェクトの各要素に対する操作を分離することで，操作の追加を容易にするデザインパターンである．この構造は，動的な振る舞い，センサの適切な配置のために利用できると考えた．Visitor パターンを利用した動的センサフュージョンのアーキテクチャを図 5 に示す．

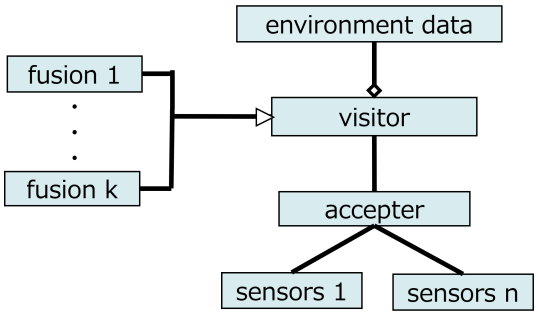

図 5 Visitor pattern

Visitor パターンを適用すると，Visitor に物体検知器，Accepter にセンサが配置される．この設計によって，物体検知器が動的に切り替わり，必要なセンシングデータをセンサ群から取得する仕組みを実現できる．物体検知器を追加する場合は，新しい Visitor クラスを定義するだけで拡張を行うことができる．センサを追加する場合，accepter の構造が変化するので，すべての Visitor を変更する必要がある．センサの種類が変化すると予想される物体検知システムには不向きであるといえる．

5.2.5 Command パターン

Command パターンは，カプセル化した処理の実行を遅らせたり，取り消ししたりするデザインパターンである．動的に切り替える構造を実現する設計パターンである．動的な振る舞いの実現，拡張性を持つ設計ができると考えた．Command パターンを利用した動的センサフュージョンのアーキテクチャを図 8 に示す．

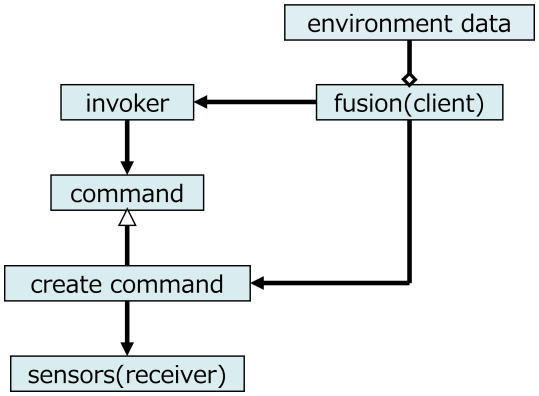

図 6 Command pattern

この設計では，各物体検知器を起動するためのコマンドが作成され，必要なタイミングで Invoker によって実行される．Invoker が起動を遅延させることができるので，物体検知の時間同期に有効であると考える．変更が起こるたびに複雑な Invoker の再定義が必要になる問題が挙げられる．

5.3 提案アーキテクチャ

議論の結果から，Bridge パターンと Composite パターンを組み合わせたアーキテクチャ提案を提案する．デザインパターン統合によって定義した提案アーキテクチャを図 7 に示す．

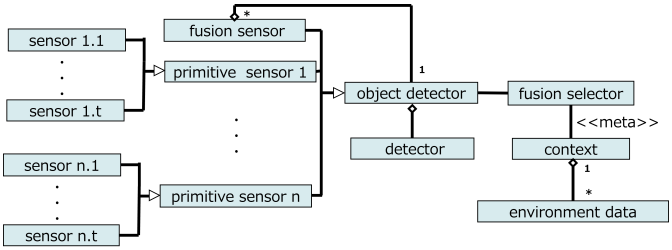

図 7 提案アーキテクチャ

Bridge パターンはセンサの種類ごとに抽象センサを定義するために利用する．広範囲の検知を行うために，同一種類のセンサが複数搭載される．このことからセンサを効率的に管理するために，Bridge パターンを利用して，primitive sensor のサブクラスとして各センサを配置した．この多層型センサによって，既存センサの追加を実現する．Composite パターンは複雑化が予想されるセンシングデータの管理を行うために利用する．複数の物体検知器を利用することから，センシングデータを再利用できる構造が必要である．Conposite パターンによって静的に多層型センサの組を定義することで，同一センサを複数の物体検知器で利用できる構造になっている．

5.3.1 提案アーキテクチャの動的振る舞い

図 8 に提案アーキテクチャの動的振る舞いを示したコミュニケーション図を示す．

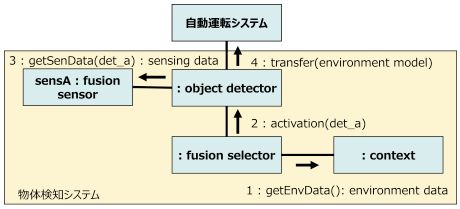

図 8 動的な振る舞い

1. context から，天候や道路情報などの走行環境データが送信される．
2. fusion selector が適切な物体検知器を選択し，選ばれた object detector を起動する．
3. fusion sensor から必要なセンシングデータを取得する．
4. object detector が物体検知を行い，周辺環境モデルを作成する．
5. 自動運転システムへ周辺環境モデルを送信する．

6　考察

6.1　拡張性

統合したアーキテクチャの拡張性について想定される変更を挙げながら考察，評価を行う．想定される拡張例を以下に示す．

例 1. センサの追加
　既存センサと新規センサ追加の 2 種類ユースケースが考えられる．

例 2. 物体検知器の追加，削除
　利用センサの組が変わることも考慮する．

例 3. コンテキストの拡張と選択ポリシーの変化

6.2　センサの追加

既存センサ，新規センサの拡張を示したものを図 9 に示す．

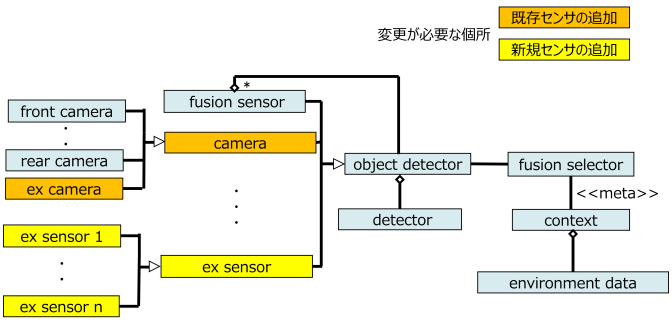

図 9　センサの追加

図で示すように新たなカメラを追加するときは，camera という抽象センサがあることで，容易にシステムへ追加することができる．新しいセンサ追加する場合は，新しい抽象センサである ex sensor を定義し，そのサブクラスに個々のセンサを配置する必要がある．抽象センサをあらかじめ定義することで，その先で起こる進化へ適応できる．いずれの変更も，システムにある他のクラスに影響を与えることなく行うことができることから，センサの追加に対して拡張性を持っていると考えられる．

6.3　物体検知器の追加，削除 (利用センサの組が変わる)

この進化を示したものを図 10 に示す．

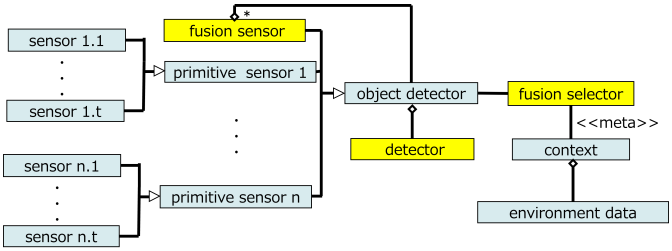

図 10　物体検知器の追加

物体検知器の追加には，利用するセンサの組みの変化と，切り替えポリシーの変更が必要である．利用するセンサの組み合わせ変更には Compositte パターンの複合センサ定義によって対応することができる．走行環境に応じて最適な物体検知器を選択するために，適切な選択ポリシーを定義する必要がある．削除を行う際も追加と同様に変更を行うことで実現できる．これらの変更は，センサ部分の構造に影響を与えることなく行われる．

6.4　コンテキスト拡張と選択ポリシーの変化

様々な物体検知方式には，それぞれ適した走行環境がある．センサ特性に合わせて天気や路面情報を取得することで，より正確な選択を行うことができる．コンテキストの変更が行われた際には，適切な選択ポリシーを再定義する必要がある．

6.5　利用時，実行時の評価

提案するアーキテクチャは，既存のセンサフュージョン技術と比べてより多くの走行環境に適応できると考えている．現在多くの物体検知器で利用されているカメラと LiDAR はいずれも悪天候での性能低下が課題になっている．environment data として，天気情報を取り入れることができれば，その変化に適応できると考えている．

7　おわりに

本研究では、動的センサフュージョンを実現するための複数アーキテクチャの定義と、それらが解決する物体検知システムにおける課題や特性について定性的な議論を行った．複合アーキテクチャを定義することで物体検知システム開発に適切な設計方針を示すことができた．今後の課題は，提案アーキテクチャのプロトタイプ作成である．提案するアーキテクチャに，実際の物体検知器やセンサを適用することで選択機構による実行時間のオーバーヘッドを測定し，実走行に与える影響を検証できると考える．プロトタイプへの変更を行い，設計で示した拡張性について検証を行う．この検証によって，設計では考えられていなかった問題点を見つけることができると考えている．

8　参考文献

[1] Y.Kondo, et al, “動的に物体検知方法を切り替えるソフトウェアアーキテクチャに関する研究,” 南山大学卒業論文要旨集，南山大学, 2023.

[2] Richard Zhang,et al, ” Sensor Fusion for Semantic Segmen-tation of Urban Scenes,” ICRA, 2015.

[3] Zhijian Liu, et al,”BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird’s-Eye View Representation,” IEEE 2023.

[4] A.Mizutani, et al,”Design of Software Architecture for Neural Network Cooperation: Case of Forgery Detection,” APSEC , 2021.