博士論文

自動運転システムの画像認識処理を対象とした
形式仕様記述言語 BBSL と機能テストの提案

田中 健人

主指導教員 青木 利晃

北陸先端科学技術大学院大学
先端科学技術専攻
[情報科学]

令和７年６月

Abstract

Automated driving systems (ADSs) are complex entities comprising numerous components, and traditional testing methods often struggle to ensure their safety, primarily due to the diversity driving environments. Among the core components of ADSs, object detection using deep neural networks (DNNs) has shown remarkable effectiveness in recognizing surrounding vehicles, pedestrians, and obstacles. However, ensuring the safety of object detection in such systems requires more than just high detection performance—it requires the ability to verify whether the positions of detected objects comply with safety-relevant specifications that govern system responses. Unfortunately, current testing practices are often grounded in informal specifications, which lack the precision and rigor needed to validate object-detection behaviors, particularly in safety-critical contexts. These informal descriptions make it difficult to define expected behavior and judge whether a system has passed or failed a given test. To address this issue, this paper first propose the bounding box specification language (BBSL), a framework capable of mathematically articulating the specifications for object and event detection and response (OEDR) tasks in ADSs. BBSL enables stakeholders such as developers and testers to mathematically express requirements concerning object positions and their relationships in a highly interpretable and verifiable manner. Building on this language, we further propose a functional testing approach for object-detection modules in ADSs. This approach leverages BBSL to define the expected behaviors in a testable form and allows us to determine whether the system ' s object recognition results satisfy these function. Notably, our approach can identify safety-critical defects that may remain undetected by conventional testing methods that rely solely on performance metrics such as accuracy or Intersection over Union (IoU). Furthermore, our proposed approach can identify safety-critical defects that conventional tests, which focus solely on performance evaluation, might overlook. Furthermore, we propose two sets of test criteria. The first set reflects the diversity of object positions and sizes within an image, while the second set includes coverage metrics that determine whether the test cases cover all conditions outlined by the BBSL specifications. Overall, our contributions facilitate the implementation of functional testing for object-detection systems using DNNs, a challenge previously considered formidable.

Keywords— Automated driving, coverage, formal specification, object detection, testing

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　背景

1.1.1　自動運転システム

自動運転システムは，交通事故の削減，渋滞の緩和，高齢者や障害者の移動支援など，多くの利点を提供する可能性がある．こうした利点が注目される中で，近年では多くの自動車メーカーや研究機関が，自動運転技術の研究開発を加速させている [8]．このシステムは，SAE（Society of Automotive Engineers）J3016 が提案する自動運転のレベル分類に基づいて定義づけされることが一般的である [1]．SAE の分類では，自動運転の度合いをレベル 0 からレベル 5 までの 6 段階に分けており，レベル 0 は完全にドライバーが運転操作を行う手動運転を指す．一方，レベル 5 は，あらゆる条件下でシステムが運転を完全に担う完全自動運転を指している．この間には，特定のタスクをシステムが支援するレベル 1 や 2，条件付きでシステムが運転を完全に実行するレベル 3，そして特定条件下でシステムがすべての運転タスクを担うが運転者が不要となるレベル 4 が含まれる．現在の研究開発は主にレベル 3 以上を対象としており，これらのレベルではシステムが運転を担う範囲が拡大し，それに伴って高度なセンサー技術やアルゴリズムの開発が求められている．一方で，自動運転システムの安全性を保証するためには，多様で不確定な条件下でのテストが必要であり，これが非常に困難な課題となっている．自動運転システムは，動的かつ複雑な道路環境で動作するため，あらゆるシナリオを網羅的にテストすることは現実的ではない．さらに，他の車両や歩行者，自転車などとの相互作用がシステムの挙動に大きな影響を与えるが，これらを正確に再現することも容易ではない．また，自動運転システムの中核をなす認識モジュールでは，ディープニューラルネットワーク（DNN）が広く採用されているが，この技術のブラックボックス的性質がテストのさらなる障壁となっている．システムの判断がどのように行われたのかを完全に解釈することが難しく，適切なテスト基準の設定に課題をもたらしている．自動運転システムは，センサーで周囲環境を認識するセンシングモジュール，認識結果を基に車両の走行計画を立案する計画モジュール，計画に従い車両を制御する制御モジュールなど，複数のモジュールで構成されている．特に，認識モジュールは，センサーからのデータを処理し，障害物や交通標識などを検出・認識する役割を担っており，他のモジュールに情報を提供する中心的な役割を果たす．その性能や安全性との乖離は，システム全体の自動運転システムの事故に直結するため，特に重要である．本論文では，この認識モジュールに焦点を当て，安全性向上のために新たなテスト手法を提案する．次節では，認識モジュールの詳細について説明する．

1.1.2　認識モジュール

認識モジュールは，自動運転システムにおいて周囲環境を正確に理解するための中心的な役割を担っている．このモジュールでは，カメラ，LiDAR（Light Detection and Ranging），

レーダーといった各種センサーが取得したデータを処理し，障害物，交通標識，歩行者などの対象物を検出・認識する．具体的には，対象物の位置や種類を特定し，それをラベル付きのバウンディングボックスと呼ばれる矩形の形式として出力する．これらの認識結果は，計画モジュールや制御モジュールが運転意思決定を行うための基盤となり，その精度と信頼性が自動運転システム全体の安全性に直結している．認識モジュールの中核技術として，DNN が広く採用されている [9]．DNN は，カメラや LiDAR，レーダーから得られる膨大なデータを基に学習し，高精度な物体認識を実現している．特に自動運転システムでは，You only look once(YOLO)[3] や Faster R-CNN[10] などのモデルが広く採用され，リアルタイムでの物体検出や分類を可能にしている．しかし，DNN ベースの認識モジュールには，従来のソフトウェアにはない特有のテスト困難性が存在する．まず，DNN はデータ駆動型のモデルであるため，トレーニングデータの品質と多様性が認識精度に直接影響を与える．トレーニングデータに含まれない未知のシナリオや，外れ値に対しては不安定な挙動を示すことが多い．例えば，夜間や霧など特定の環境条件や違法駐車された車両などの対象物の認識において，モデルの性能が急激に低下することが報告されている．このようなケースでは，従来のテストではカバーしきれない状況が頻発する．さらに，DNN はブラックボックス的な特性を持ち，その内部構造や意思決定プロセスを完全に解釈することが難しい．一般的なソフトウェアでは，コードカバレッジや分岐カバレッジといった指標を用いてテストの網羅性を評価するが，DNN ではこれらの指標をそのまま適用することができない．また，DNN の性能はモデルのサイズやアーキテクチャにも依存する．たとえば，大規模なモデルほど表現能力が高まるが，その分過学習や計算負荷のリスクも増加する．一方で，小規模モデルでは計算効率が高いものの，学習データに含まれる複雑なパターンを十分に捉えられない場合がある．

この研究では，このような認識モジュール内の DNN の中でも画像認識システムに焦点を当てる．ここでの画像認識システムとは，画像を入力として受け取り，検出されたオブジェクトの位置表すバウンティングボックスと呼ばれるラベル付きの矩形を出力として生成するものを想定している．画像認識システムの評価には，現状，主に Intersection over Union（IoU）が用いられている [11]．IoU は，認識モジュールによって推定されたバウンディングボックスと正解のバウンディングボックスの重なりの割合を基準とする指標であり，物体検出の精度を測定するための標準的な方法である．しかし，IoU には安全性評価におけるいくつかの限界が存在する．例えば，同じ IoU スコアを持つ 2 つのケースがあったとしても，物体が画像上のどこに位置しているかによって，安全性に与える影響は大きく異なる．IoU はこうした位置関係の違いを考慮しないため，システムが適切な判断を下すかどうかを評価するには不十分な場合がある．このような場合の具体例として，IoU スコアは同じになるが，DNN の推論結果のバウンディングボックスの位置が異なるようなケースを考える．図 1.1 は，上部の 2 枚も下部の 2 枚も同じ環境における 1 枚の入力画像に対して，推論結果が異なる 2 種類の自動運転システムの出力を表現した図であり，車両のグラウンドトゥルースラベルは赤色のバウンディングボックスで示しており，DNN の推論結果のラベルは緑色のバウンディングボックスで示している．この時，全ての画像において IoU は 1/3 であるため，IoU 上ではどの場合も同じ性能であると評価される．しかし，自動運転システムの機能として，進行方向に車両がある場合に停止するという仕様があった場合, 上部の右の出力は，進行方向にある車両を進行方向上に予測できているが，左の出力は進行方向にある車両を進行方向にない位置に誤認しているため，仕様に従っておらず衝突の危険性がある．また，別の例として，自動運転システムの機能として，適切な車間距離内に車両がある場合に停止するという仕様があった場合, 下部の右の出力は，適切な車間距離内の車両を機能に影響のない範囲で予測できているが，左の出力は，適切な車間距離内の車両を，適切な車間距離外にある位置に誤認しているため，仕様に従っておらず衝突の危険性がある．このように，既存の DNN の性能評価では，対象の物体がどの

位置にあるかについて考慮することが困難であり，自動運転システム全体の要件や仕様が反映されていないため，安全性の面で不充分である可能性を示唆している．これらの課題に対処するため，本論文では認識モジュールの既存のテスト手法では考慮することが困難だった要件を補うために，新たな機能テストのアプローチを提案する．提案手法は，画像認識処理の出力が機能仕様に準拠しているかどうかを評価することで，IoU の限界を補い，自動運転システム全体の安全性を向上させることを目指している．

図 1.1: IoU を用いた評価では安全性が不充分になる例

1.1.3 自動運転システムの仕様

自動運転システムの設計と開発において，仕様はシステムの信頼性や安全性を保証するための基盤である．仕様は，システムがどのように振る舞うべきか，あるいは振る舞うべきでないかを定義するものであり，設計，実装，テスト，そして運用に至るまでの全プロセスを通じて重要な役割を果たす．特に，自動運転システムのように多様な環境条件と複雑な相互作用が要求されるシステムでは，安全性を損なわないために仕様に要件を正確に記述し，一貫性を保つことは欠かせない．多様な走行環境において，様々なタスクを実行する自動運転システムの仕様策定に関する研究は活発に行われており，近年，自動運転システムにおける仕様策定の重要性は，国際的な規制機関や研究機関によっても強調されている．例えば，The National Highway Traffic Safety Administration（NHTSA）[2] は，自動運転システムの安全性を評価するためのガイドラインを複数発表しており，システムが満たすべき安全要件を定義している．その中には，障害物検出や車両の動的挙動に関する具体的な性能要件が含まれており，これらは自動運転車両の認証や運用の基準として機能している．また，ISO 26262（自動車における機能安全）[12] や ISO 21448（意図した機能の安全性）[13] といった国際規格も，仕様策定の一部として参照されることが多い．これらの規格は，特に安全性の観点から，設計プロセスやテスト基準を仕様に基づいて定義する必要性を強調している．このような標準やフレームワークにおける設計プロセスは，初めに開発する自動運転システムの自動化レベルを特定し，次に，ODD(operational design domain) を確立し，OEDR(object and event detection and response) を明確にするという手順になっている．具体的には，道路の種類，速度制限，照明条件，天候設定，その他の運用上の制約など，自動運転システム，またはその機能が動作するように設計されている特定の条件を確立し，多様な運転環境を「車線合流」などのやや抽象的なシナリオに分類し，これらのシナリオに基づいて，特定の運転環境を監視し，物体やイベントに適切に対応するための OEDR の設計などが含まれる．ここでの，自動化レベル，ODD，OEDR は SAE J3016 標準で定義されているものと同義である [1]．1.1.2 節で示した認識モジュールは，この中でも特に OEDR のタスクの実行において重要な役割を果たす．このような

プロセスににによって，安全性と意図した機能の適正性を保証するための基盤として利用されている．しかし，これらのフレームワークで使用される自然言語，グラフ，非形式的な表現は，システムの挙動を正確に記述するには不十分である．特に，認識モジュールの動作や物体間の位置関係を正確に記述する能力に欠けており，このような曖昧さをもって機能仕様を定義すると，その仕様に基づいたテストの実行において多くの悪影響をもたらす可能性がある．例えば，テストケースの生成において，曖昧な記述は必要なシナリオを漏らす原因となり，特定の環境下での認識モジュールの性能を正確に評価することが難しくなる．さらに，非形式的な仕様に基づくテストは，テスト網羅性を客観的に評価する基準を欠いているため，どの程度システムが仕様に適合しているかを定量的に把握することが困難である．

このような問題に対処するために，ソフトウェア工学の分野では形式仕様記述言語の研究が進められている．形式仕様は，システムの動作や制約を数学的に記述する手法であり，その特徴として曖昧さの排除と一貫性の確保が挙げられる．形式仕様を利用することで，システムが期待される条件を正確にモデル化でき，設計から実装に至るプロセス全体で一貫性を持った検証が可能となる．これにより，仕様に基づくテストケースの生成が体系的に行われ，テストの網羅性を客観的に評価できる環境が構築され，複雑なシナリオや相互作用を含むテストケースの漏れを防ぎ，システムの信頼性を向上させることが可能となる．

本研究では，自動運転システムの走行環境におけるオブジェクト間の位置関係を正確に記述するために特別に設計された BBSL(Bounding box specification language) と呼ばれる形式仕様記述言語を提案する．BBSL を使用すると，厳密で明瞭な仕様を作成できるため，認識モジュールの機能テストが容易になり，仕様に密接に関係のある重要な空間関係が適切に評価されることが期待できる．

1.1.4 安全性テストにおけるカバレッジ

テストに対して充分性を確保することは，システムの信頼性を確保するために重要である．テストの充分性とは，一連のテストケースがシステムの重要な側面をどの程度包括的にカバーしているかを評価する概念であり，カバレッジとは，その充分性を示す指標である．従来のテストカバレッジでは，コードまたは関数レベルで充分性を測定することが多い [14]．カバレッジが 100%であることは実装した機能が適切であることを裏付けるものではないが，カバレッジが高いことはテスト結果に対する信頼を高める．しかし，一般的なソースコードと異なり，DNN ベースのソフトウェアはコードロジックが明示的に表現されているわけではないため，コードカバレッジのような基準の確立が複雑になる．一応，DNN ベースのソフトウェアにおいても，ニューロンカバレッジという基準が提案されている [15][16]．ニューロンカバレッジは，特定の入力データに基づいて DNN 内のニューロンがどの程度活性化されるかを測定するものである．しかしながら，いつくかの研究によって，このようなカバレッジ基準は DNN テストの充分性を評価するために汎用的に使用できるものではないことを主張している [17][18]．実際に，ニューロンカバレッジを用いた研究では，敵対的攻撃と呼ばれる DNN ベースのソフトウェアに対する攻撃の 1 種に焦点がおかれていることが強調されており，人工的に生成された入力である敵対的攻撃に対して堅牢であることと，実際の自然な入力セットに対して充分に障害を検出したかは必ずしも相関していないことが既に確認されている．また，前述したように形式仕様記述言語を用いことで，どの程度システムが仕様に適合しているかを定量的に把握しやすくなることが期待できるが，現状，テストケースを仕様と比較するための形式的な基準が欠如しており，テストの完全性を保証することが困難となっている．そのため，本研究では提案した形式仕様記述言語 BBSL を活用したカバレッジの評価手法を提案することで，各テス

トケースが仕様のどの部分を満たしているかを形式的に評価し，カバレッジの完全性を客観的に保証することを試みる．

さらに，別の重要な問題として，テストをどのタイミングで終えるかを決定する必要がある．テストケースはソフトウェアプログラムの 1 回の実行のための入力のコレクションだが，実際の運用時に想定される全ての入力をテストケースとして完全に準備することは非常に困難である．この問題は，動的な交通環境や多様な運用条件を考慮する必要のある自動運転システムにおいてより顕著となる．そこで，本研究では多様性を測定するパラメータとしてオブジェクトの位置や大小に着目し，テスト実施者のテストしたい範囲に関して充分にテストが実施されているかに関して飽和度を測定する方法を提案し，これらの基準を用いたテストの終了条件を提案する．

1.2 研究の目的

前述した通り，自動運転システムの普及に伴い，認識モジュールの安全性を評価するためのテスト手法の重要性が年々高まっている．しかしながら，従来の評価手法や仕様記述方法には以下のような課題が存在している．まず，従来の評価手法は IoU やニューロンカバレッジのような指標に依存しており，これらの指標は自動運転システムの安全要件を反映するためには不十分である．たとえば，IoU は主にバウンディングボックス間の重なり具合を評価するものであり，位置関係や方向性を含むより詳細かつ，重要な安全要件を含む画像認識の挙動を十分に評価できない．このような制限のため，従来の評価手法は自動運転システムの重大な安全リスクを見逃す可能性がある．次に，既存の仕様には非形式的で曖昧性を含む記述があり，具体的には上述した物体間の位置関係や動作条件のような空間的な要素を網羅的かつ正確に記述することが難しい．この曖昧さにより，テストケースの生成が不完全になり，システムの安全性を保証することが困難となる．この問題を解決するために，形式仕様記述を用いた厳密かつ一貫性のある仕様記述が必要である．さらに，画像認識システムのテストに対する網羅性やテスト終了基準の不明確さも課題として挙げられる．動的な交通環境や多様な運用条件を考慮したテスト基準が欠如しており，これがシステムの信頼性評価を妨げている一因となっている．

本研究の目的は，これらの課題に対応し，自動運転システムの認識モジュールの安全性評価を向上させることにある．そのために，形式仕様記述言語を設計し，機能テストのフレームワークを構築し，仕様に基づく新しいカバレッジ基準を確立する．特に，認識モジュールの中でも物体検出のためのカメラを入力とする画像認識を対象にする．前提として，自車両の正面を撮影するカメラを入力とする画像認識を想定している．提案する仕様と機能テスト手法には以下が含まれる．

- 自動運転システムの OEDR の仕様を画像情報で記述することができる形式仕様記述言語
- OEDR の仕様を用いた画像認識処理の機能テスト手法
- 仕様に記述された要件と相関のあるテストカバレッジ
- 画像上のオブジェクトの多様さと相関があり，充分なタイミングでテストを停止するためのカバレッジ

そして，これらの目的のために大きく次の 4 点の提案を行う．

1. 自動運転システムの OEDR の仕様を記述するための形式仕様記述言語 BBSL

2. BBSL で記述した機能仕様と機能テスト

3. BBSL で書かれた仕様ベースのカバレッジ

4. テストに用いた画像上のオブジェクトの多様さと相関のあるカバレッジ

本研究の貢献は，形式仕様記述言語 BBSL の開発と，それを用いた機能テスト手法およびカバレッジ基準の提案により，自動運転システムの全体のコンテキストにおける画像認識モジュールの仕様を記述することを可能とし，従来の手法では検出が難しかった最終的な運転タスクに影響を与える画像認識の不具合の早期発見を実現した点にある．そのために，まず BBSL を提案する．BBSL は，物体間の位置関係を含む自動運転システムの動作条件を厳密に記述することを目的としており，従来の仕様では扱えなかった位置関係や方向性などの要素を正確に定義する．この形式仕様記述言語を活用することで．認識モジュールが機能要件をどの程度満たしているかを定量的に評価することを可能にする．次に，BBSL を基盤とした機能テスト手法を構築する．この機能テストは，一般的な機能テストと異なり，画像入力に対して形式仕様により定義された機能（停止する，停止しないなど）を期待値とし，画像認識の出力との整合性を確認するものである．このときの期待値は，単なるグラウンドトゥルースの正解ラベルではなく，BBSL で形式的に定義された「状況に応じた適切な動作」を指す．したがって，本テストは，従来の画像認識の精度評価とは異なり，自動運転全体の要件を満たしているかどうかを判断するための機能テストであり，機能仕様に照らして不適切な出力を検出する役割を担っている．さらに，BBSL を用いて，テストカバレッジ基準を定量化する新しい手法を提案する．この基準により各テストケースが仕様をどの程度網羅しているかを正確に把握し，重要なシナリオの漏れを防ぐことができる．また，別のカバレッジ基準として，テスト終了条件の判断を正当化し，テスト進捗を効率的に管理するための基準を提供する．それぞれが自動運転システムの安全性評価における現在の課題に直接対応し，システム全体の信頼性を向上させるための具体的な手段を提供するものである．

また，本研究では提案する機能テスト手法について，以下の観点から自動運転システムの安全性テストとしての有用性を評価する．

- OEDR の仕様として書くべき機能の条件が記述可能であること
- 既存のシナリオに対して，全ての仕様が手作業でも現実的なコストで記述可能なこと
- 従来の性能評価では検出ができなかった安全性を損なう不具合を検出可能であること
- 不足したテストケースの条件を特定するのに各カバレッジが効率的であること

1.3 本論文の構成

本論文の構成は以下である．第 1 章に，本論文の背景を示した．第 2 章に，本論文の背景となる既存用語，概念を概説する．第 3 章に，本論文の提案の一つである形式仕様記述言語 BBSL を提案する．第 5.3 章に，本論文の二つ目の提案である BBSL を用いた機能テスト手法を提案し，そのカバレッジを提案する．第 5 章で，提案手法を評価し，第 6 章で考察を述べる．第 7 章に，本論文に示す研究に関連する研究を示す．第 8 章で本論文をまとめ，今後の課題を示す．

第2章　背景技術と準備

2.1　自動運転システムに関する基本用語と概念

ここでは，自動運転システムに関する基本用語と概念について本論文で使用するものに焦点を当てて説明する．

まず，自動運転システムの能力を示す基準として，自動運転レベルという概念を説明する．自動運転レベルは，モビリティ専門家を会員とする米国の非営利団体である Society of Automotive Engineers(SAE)International によって，SAE J3016[1] に定義されており，国際的に使用されている．図 2.1 に示すのが，SAE にてまとめられた自動運転のレベルと基準を表した図であり，この基準では，レベル 0 からレベル 5 までの 6 段階が定義されており，各レベルはシステムの自動化範囲に基づいて分類される．画像上の DDT,ODD,OEDR という用語に関しては，後ほど説明をするが，これらは自動運転システムにおける各タスクに対して，どの程度システムによって自動化されているかで定義されている．簡潔に述べると，レベル 0 では，運転操作のすべてが人間によって行われ，システムは警告や情報提供に限定される．レベル 1 では，システムが運転操作の一部，具体的にはステアリングまたは加減速のどちらか一方を支援する．レベル 2 では，ステアリングと加減速の両方をシステムが制御するが，周囲の状況の監視や運転の最終責任は人間が担う．一方で，レベル 3 ではシステムが特定の限定的な状況ですべての運転操作を担当するが，システムの介入要求等を受けて，人間は必要に応じて介入する必要がある．レベル 4 では，システムが特定の限定的な状況で完全に運転操作を担当し，人間の介入が不要となる．最も高度なレベル 5 では，いかなる条件下でもシステムが運転操作を完全に担い，人間の操作や監視を必要としない．例えば，本論文の 5.1 節で使用する自動運転システム「Level 3 Conditional Automated Traffic Jam Drive」とは，交通渋滞時の特定の条件下で自動運転システムが運転操作をすべて担当する自動運転システムを指す．このシステムでは，交通渋滞時に限りシステムが車両のステアリング，加減速，停止を完全に制御し，ドライバーはこれらの操作から解放される．しかし，システムが動作可能な条件を逸脱する場合や，システムの介入要求が発生した場合には，ドライバーが速やかに運転操作を引き継ぐ必要がある．このように，レベル 3 ではシステムが運転を担当する範囲が明確に定義されており，人間がバックアップとして存在することが求められる．特に，人間主体のレベル 2 の場合を自動運転支援システム，そして，レベル 3 以降のシステム主体の自動運転システムを狭義の自動運転システムと呼ぶ場合もあり，本論文でも，このようなレベル 3 以降の自動運転システムを中心に議論を展開する．

次に，自動運転システムを理解する上で重要な概念となる Dynamic Driving Task(DDT)，Operational Design Domain(ODD)，Object and Event Detection and Response(OEDR) を説明する．これらはそれぞれ異なる役割を持ちながら，自動運転システムの全体的な機能を構成する要素として相互に関連しており，自動運転システムの仕様を含む設計段階から用いられる概念であり，自動運転システムレベルと同様に，SAE J3016 に定義されており，国際的に使用されている．まず，DDT は，日本語では，動的運転タスクと呼ばれ，自動車の運転操作に関するすべてのタスクを指し，大きく戦術的タスクと運転操作タスクの

Level	Name	Narrative definition	DDT		DDT fallback	ODD
			Sustained lateral and longitudinal vehicle motion control	OEDR		
Driver* performs part or all of the *DDT						
0	**No Driving Automation**	The performance by the *driver* of the entire *DDT*, even when enhanced by *active safety systems*.	*Driver*	*Driver*	*Driver*	n/a
1	**Driver Assistance**	The *sustained* and *ODD*-specific execution by a *driving automation system* of either the *lateral* or the *longitudinal vehicle motion control* subtask of the DDT (but not both simultaneously) with the expectation that the *driver* performs the remainder of the *DDT*.	*Driver* and *System*	*Driver*	*Driver*	Limited
2	**Partial Driving Automation**	The *sustained* and *ODD*-specific execution by a *driving automation system* of both the *lateral* and *longitudinal vehicle motion control* subtasks of the *DDT* with the expectation that the *driver* completes the *OEDR* subtask and *supervises* the *driving automation system*.	***System***	*Driver*	*Driver*	Limited
***ADS* ("*System*") performs the entire *DDT* (while engaged)**			*System*	***System***	*Fallback-ready user (becomes the driver during fallback)*	Limited
3	**Conditional Driving Automation**	The *sustained* and *ODD*-specific performance by an *ADS* of the entire DDT with the expectation that the *DDT fallback-ready user* is *receptive* to *ADS*-issued *requests to intervene*, as well as to *DDT performance-relevant system failures* in other *vehicle* systems, and will respond appropriately.				
4	**High Driving Automation**	The *sustained* and *ODD*-specific performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	***System***	Limited
5	**Full Driving Automation**	The *sustained* and unconditional (i.e., not *ODD*-specific) performance by an *ADS* of the entire *DDT* and *DDT fallback* without any expectation that a *user* will respond to a *request to intervene*.	*System*	*System*	*System*	**Unlimited**

図 2.1: SAE における自動運転のレベル定義 [1]

2 つに分類される．戦術的タスクは，特定の交通状況に応じて車両の挙動を調整するものである．例えば，前方に障害物がある場合に減速したり，車線変更を行ったりする判断が含まれる．一方，運転操作タスクは，ステアリング操作や加減速の制御など，車両の直接的な操作に関するものである．例えば，渋滞時の車両制御では，DDT の戦術的タスクとして前方車両との車間距離を適切に保つ判断が含まれ，運転操作タスクとして，その距離を保つための減速や停止の操作が実行される．次に，ODD は，日本語では，運用設計領域と呼ばれ，ある自動運転システム又はその機能が作動するように設計されている特定の条件を指すものである．具体的には，都市部，高速道路，駐車場などの地理的条件や， 天候や照明条件（昼間，夜間）などの環境条件，そして，速度制限や交通密度などの運転条件が含まれる．例えば，本論文の 5.1 節で使用する自動運転システム「Level 3 Conditional Automated Traffic Jam Drive」の設計例では，ODD が「高速道路上の渋滞状態，時速 60km 以下，晴天時」であることが定義されており，これを超える状況では自動運転システムが動作しないよう設計されている．次に，OEDR は，日本語では,「物体と事象の検知と反応」と呼ばれ，自動運転システムが周囲環境を認識し，物体やイベントに適切に応答するタスクを指す．具体的に，物体とは，車両，歩行者，障害物などであり，イベントとは信号が赤になった，他車が割り込んできた場合などがある．例えば，交差点で歩行者が横断歩道を渡っているときに，歩行者を検出し，停止または減速することで適切な応答をすることなどが OEDR として考えられる．DDT，ODD，OEDR はそれぞれ独立した概念であるが，実際の自動運転システムでは密接に関連している．例えば，ODD で定義された運転条件下で DDT を実行する際，OEDR が環境の変化や予期しないイベントに対する適切な応答を提供することが必要となる．特に，OEDR 及び，その仕様は，自動運転システムの画像認識の機能に重要な要素であり，定められた OEDR をどのように機能仕様で表現してテストすれば自動運転システムの安全性向上の一助になるかは本論文の主な焦点である．

2.2 自動運転システムの安全性フレームワーク

自動運転システムの安全性を確保するため，各国の機関や団体はさまざまなフレームワークやガイドラインを策定している．これらの多くは，自動運転システムが直面する多様な運転状況やリスクに対応するために，シナリオベースのアプローチを採用し，具体的な運転状況や環境に応じた設計と評価を行うことを推奨している．例えば，米国の国家道路交通安全局（NHTSA）は「A Framework for Automated Driving System Testable Cases and Scenarios」を公表し，自動運転システムのテストケースとシナリオの構築に関する指針を提供している [2]．このフレームワークでは，自動運転システムの機能を評価するために，運転シナリオを体系的に分類し，テスト可能なケースを設定する方法が示している．また，ドイツの PEGASUS プロジェクトでは，自動運転車のテストと認証のための包括的なフレームワークを提供することを目的としており，シナリオベースのテスト手法を中心に据えている [19]．このプロジェクトでは，実際の交通状況を反映したシナリオを収集・分類し，それに基づくテストケースを生成することで，自動運転システムの性能と安全性を評価する．例えば，高速道路での車線変更や都市部での交差点通過など，多様な運転シナリオに対応するテストが行われている．さらに，日本では，日本自動車工業会（JAMA）が,「自動運転の安全性評価フレームワーク Ver3.0」を公表し，物理原則に基づくアプローチ（Physical Principal Approach Process）を提唱している．このフレームワークでは，運転に必要な「認知」「判断」「操作」の各プロセスを詳細に分析し，それぞれのプロセスにおける潜在的なリスクを特定・評価する．さらに，具体的な運転シナリオを設定し，各シナリオにおける自動運転システムの挙動を評価することで，安全性の確保を図っ

ている．これらのフレームワークに共通するのは，シナリオベースのアプローチを採用し，具体的な運転状況に応じた自動運転システムの性能評価を行う点というである．シナリオごとにシステムの挙動を詳細に分析・評価することで，より現実的で包括的な安全性評価を可能としている．本論文でも，これらのフレームワークに則った上で，画像認識システムの仕様に用いるために，現状，充分に決定されていないシナリオごとの物体間の位置関係に基づく動作条件を特定し明確に記述する方法を提案する．実際に，節 5.1 では，既存の自動運転システムの安全性フレームワークの 1 つとして，NHTSA の安全性フレームワーク「A Framework for Automated Driving System Testable Cases and Scenarios」を参照し，仕様を作成している．そこで，ここでは，NHTSA の「A Framework for Automated Driving System Testable Cases and Scenarios」について，本論文で使用する要素に焦点を当てて説明する．

まず，NHTSA の研究では，自動運転システムの運転制御に関連する戦術的および運用上の操作にはどのようなものがあるかを機能として整理している．そして，SAE レベル 3 以降の既存の研究中・開発中の自動運転システムを調査することで，各機能がどういう自動運転システムで備わっているかをまとめている．このように既存の自動運転システムを調査することで，機能を抽出し，自動運転システムに実装する機能を Generic ADS Feature として定義している．次に，開発する自動運転システムが前提とする ODD を決定している．まず，分類するための枠組みとして，想定すべきとされている様々な ODD 要素をカテゴリとサブカテゴリからなる階層的な形式により，整理してる．例えば，最上位のカテゴリは，道路やトンネル，下水道などに該当する物理的なインフラストラクチャ，法的な制約や，交通状況などに該当する運行上の制約，信号機や，歩行者など検出し応答すべき対象に該当するオブジェクト，他の車両との通信リンクや信号機から送られる電波の形態などのテクノロジーに該当する接続性，天候や太陽光などに該当する環境条件，そして，児童通学路などに該当する空間的に制限がかかるゾーンの 6 種類からなる．このように，カテゴリーごとに大別した上でさらに各子カテゴリから次の子カテゴリーへ階層的に ODD を決定していく．例えば，最上位カテゴリの環境条件に対して，その子カテゴリの天候を決定し，さらに子カテゴリの雪や雨などが定義され，雨は子カテゴリーに少ない，中間，多いなど定性的に選べるよう項目がふられている．このように，階層的に ODD を整理し定義した上で，開発する自動運転システムの対象となる ODD を順番に決定していく．カテゴリによって，yes，no の 2 値で記すものから，speed limits のように値で記すものなどがある．一般に，自動運転システムをテストする場合に，どの程度多様な環境に対してテストできているかのカバレッジは，この ODD で定めたカテゴリの組み合わせで定義される場合が多い．

次に，OEDR を決定する方法について説明する．まず，自車に対する相対的な位置の領域を定義する．図 2.2 と図 2.3 は A Framework for Automated Driving System Testable Cases and Scenarios」の OEDR の仕様の策定について掲載されているのであり，前者は，自動運転システムに対して相対的に定義される領域を説明する図であり，後者は，OEDR を決定する際のオブジェクトと，イベントの例として掲載されている図である．NHTSA は，図 2.2 のように，Frontal Zone，Side Zone，Rear Zone という 3 つの領域を定義し，自動運転システムが対象とした ODD のうち，OEDR のイベントとなるオブジェクトを抜き出している．そして，図 2.3 のように，それらのオブジェクトがどの領域で，どのイベントを引き起こすことが想定されるか決定していく．このような順序で整理し，最終的にどのようなオブジェクト，イベントに対して自車がどう対応するかの OEDR を決定する．

以上によって，決定された自動運転システムの種類からの機能，対象となる ODD，整理した OEDR を組み合わせ，該当するテストシナリオの構成を定める．例えば，図 2.4 はテストシナリオの一例であり，Tactical Maneuver Behaviors は左折により隣接する左車線への車線変更や，ODD Elements の Daylight はテストが行われる時刻や太陽の位置な

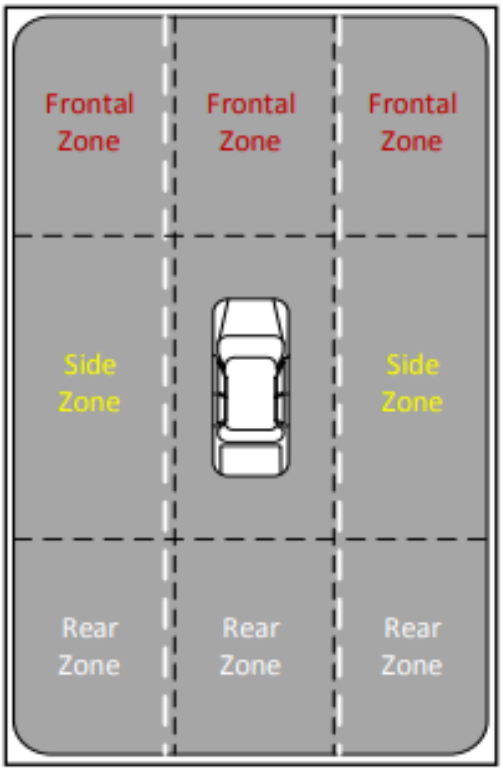

図 2.2: 自車に対して対応が必要な領域分類 [2]

Objects	Events/Interactions
Vehicles (e.g., cars, light trucks, heavy trucks, buses, motorcycles)	**Lead vehicle decelerating (frontal), lead vehicle stopped (frontal), lead vehicle accelerating (frontal), changing lanes (frontal/side), cutting in (adjacent), turning (frontal), encroaching opposing vehicle (frontal/side), encroaching adjacent vehicle (frontal/side), entering roadway (frontal/side), cutting out (frontal)**
Pedestrians	**Crossing road – inside crosswalk (frontal), crossing road – outside crosswalk (frontal),** walking on sidewalk/shoulder
Pedalcyclists	**Riding in lane (frontal)**, riding in adjacent lane (frontal/side), riding in dedicated lane (frontal/side), riding on sidewalk/shoulder, **crossing road – inside crosswalk (frontal/side), crossing road – outside crosswalk (frontal/side)**

図 2.3: 対象オブジェクトの位置とイベント対応例 [2]

どを記している．そして，OEDR Behaviours には，テスト中の他車両の数や，その初期条件（位置，速度，方向など）および軌道などを記述する．このように，これまで決定した要素からシナリオを定義し，段階的に詳細化しながらテストシナリオを作成していくという流れが，NHTSA を含む既存の安全性フレームワークの流れとなっている．

Scenario Elements	Example
Tactical Maneuver Behaviors	Perform lane change/low-speed merge
ODD Elements	Arterial roadway type
	Asphalt roadway surface
	Lane markers
	Straight, flat
	72 kph (45 mph) speed limit
	Nominal traffic
	Clear, dry weather
	Daylight
	...
OEDR Behaviors	Detect and respond to relevant adjacent vehicles (frontal, side, rear)
Failure Mode Behaviors	N/A

図 2.4: テストシナリオの一例 [2]

このように，既存の自動運転システムの安全性フレームワークでは，ODD，OEDR を明確にしながら，具体的なテストシナリオを定める手法がとられる．しかしながら，OEDR に関して，オブジェクトの種類とそのオブジェクトに対するイベントと，自車のとるべき反応に関しては整理されているが，オブジェクト間の位置関係に基づく動作条件を特定し明確にするのは困難であるのが現状である．

2.3 自動運転システムのオープンデータセット

自動運転システムの開発において，実際の走行環境を反映した大規模かつ多様なデータは，学習・評価およびアルゴリズムの改良に不可欠な要素となる．一般的な方法としては，これらのデータは実データに対して，分類やタグ付けなどのアノテーションをつけることで作成される．特に，カメラ入力を基盤として 2 次元のバウンディングボックスを返す画像認識分野では，自動運転システムが注目される以前から，よく研究されている分野であるため，多くのオープンデータセットが作成され，企業の開発プロセスや，異なる研究チームにおいて効率的に作業ができるように利用されてきた．例えば，2005 年から 2012 年にかけて成長した PASCAL VOC データセットは，自動運転システムに関連するクラスも含むデーターセットの一つとして公開されている [20]．Pascal VOC は，vehicle や person を含む 20 種類のオブジェクトクラスに対する 2 次元のバウンディングボックスアノテーションが付与された画像データセットで，画像認識アルゴリズムの基礎的な評価に用いられてきた．シンプルながらも多様なオブジェクトと背景の組み合わせを含み，物体検出技術の初期の評価基準として確立されているが，運転シナリオで遭遇するものを代表しないシーンやスケールで単一のオブジェクトが含まれているため，走行環境を反映できてはおらず，自動運転システムにおける学習・評価としてのデータセットとしては不充分であった．そこで，2012 年に公開された KITTI Vision Benchmark では，自動運転システム向けの実走行データを利用して，比較的大量のラベル付き運転シーンが提供された [21].

このデータセットでは，車載カメラで撮影された実際の走行環境の画像や動画,Lidar データを含み，手作業によって車両，歩行者，自転車などのオブジェクトに対して，2 次元のバウンディングボックスのみではなく，3 次元のバウンディングボックスなどのアノテーションが付与されているため，自動運転に関連するシステムで広く使用されているデータセットの 1 つである．しかし，データ量や取り扱っているクラスの数という点では，従来の ImageNet[22] や COCO[23] などの汎用的な画像認識用のデータセットよりも少ないものとなっている．また，自動運転タスクにおけるベンチマークのためのデータセットとしては Cityscapes がある [24]．Cityscapes も自動運転に関連するシステムで広く使用されているデータセットの 1 つであり，特に都市部の走行シーンを対象とした大規模なデータセットである．高解像度の画像と，詳細なセマンティックセグメンテーションやインスタンスセグメンテーションのラベルが提供されている点が特徴であり，2 次元のバウンディングボックスの情報も含まれている．都市環境では，交通量の多いシーン，複雑な背景，そして多数の小さなオブジェクトが存在するため，Cityscapes はこれらの要素を正確にアノテーションすることで，都市部における自動運転システムの認識アルゴリズムの評価に極めて有用なデータセットとなっている．また，Cityscapes は，道路のレイアウト，歩行者の動き，車両の相互作用など，都市特有のシナリオを豊富に含むため，実際の走行環境を模擬した評価が可能となっている．比較的に新しいデータセットとしては，Waymo Open Dataset があげられる [25]．Waymo Open Dataset は，Alphabet 傘下の Waymo が公開したデータセットで，これまでに公開された自動運転システムに関するデータセットの中でも最大級なものとして，最も多様でリッチなものとされており，近年自動運転システムの研究開発において注目されている。このデータセットは，車載カメラだけをとっても 360 度をカバーする 5 つのカメラから高解像度で撮影された映像と，他車や歩行者，自転車に対する，精度の高い 2 次元のバウンディングボックス，及び，3 次元のバウンディングボックスのアノテーションが付与されている．また，Lidar に関しても複数台に対応したデータを含んでいる．Waymo Open Dataset は，都市部や郊外，さらには多様な天候条件下での走行シナリオを豊富に提供しており，実運用に近い環境での物体検出アルゴリズムのより厳密な安全性検証や性能改善のための基盤として重要視されている．本論文においては，データ量や取り扱っているクラスの数が多く，画像認識の学習用としての多くの実績がある ImageNet と COCO のデータセットで学習された学習済みの画像認識を使用する．また，実走行環境のシーンをで提供されており，自動運転システムの適切なコンテキストの評価が比較的しやすいことが期待できるため，その画像認識のテストのためのデータセットとしては，KITTI のデータセットと Waymo のデータセットを使用する．

2.4 画像認識システム

自動運転システムの認識モジュールは，周囲環境を正確に理解し，適切な応答を行うための中核的な役割を担っている．画像認識技術は，物体検出，分類，位置推定などの複数のタスクを通じ，走行環境における車両，歩行者，信号機などの対象物を抽出する．これらのタスクの実現には，DNN(Deep Neural Netwaorks) が用いられる．DNN は，複数の層を持つニューラルネットワークであり，入力データから低次の特徴を抽出し，層を重ねるごとに高次の抽象的な特徴を学習する仕組みである [26]．この多層構造により，従来の機械学習手法では捉えにくかった複雑なパターンや関係性を学習できる点が大きな強みとなっている．この DNN の中でも，特に画像認識タスクにおいては，Convolutional Neural Networks（CNN）が非常に効果的であるとされている [27]．CNN は，入力画像の空間的構造に着目し，まず，畳み込み層を用いて局所的な特徴（エッジ，テクスチャ，形状など）を抽出する [28]．そして，プーリング層を用いて，入力された特徴マップ内の隣接する領域

ごとに，最大値や平均値などの代表値を算出する．この処理により，特徴マップの解像度が低下し，データ量が減る，つまり，次元削減が行われるとともに，画像内の微小な位置ズレやスケールの変動に対しても，同じ特徴が抽出されるようになる．従来の全結合型のDNNが，画像全体の各ピクセルを独立した入力として扱うため，非常に多くのパラメータが必要となり，計算資源が多く必要ではったことに対し，CNNでは，パラメータ数を大幅に削減しながらも高い認識精度を実現できる．

本論文では，自動運転システムにおける画像認識の代表的な手法として，YOLO（You Only Look Once）を使用している [29]．YOLOは，CNNを基盤としたエンドツーエンドの物体検出モデルであり，画像全体を一度に処理して対象物の位置情報とクラス確率を同時に予測する [3]．図 2.5 はYoloの画像認識の仕組みの概要を示した図である．入力画像をあらかじめS × Sのグリッドと呼ばれる正方形に分割し，各グリッドセルは，そのセル内に対象物の中心が存在する場合に物体検出の責任を担い，固定個数 B 個のバウンディングボックスを予測する．予測するバウンディングボックスは，x軸及び，y軸の中心座標と幅w，高さhによる位置情報 (x, y, w, h) に加え，そのバウンディングボックスがオブジェクトである信頼度スコア C の5つの予測で構成される．信頼度スコア C はオブジェクトが存在する確率 $P(Object)$ とその位置の正確さを評価する指標であるIntersection over Union（IoU) を用いて以下で定義される．

$$C = P(Object) \times IoU(b, b^*).$$

ここで，b は予測されたバウンディングボックス，b^* はグラウンドトゥルースのバウンディングボックスを表し，IoU は以下で計算される指標である．

$$\text{IoU(A,B)} = \frac{|A \cap B|}{|A \cup B|}.$$

一方で，グリッドセル内がオブジェクトである場合に，各クラス $Class_i$ に属する確率の推定として各クラスの条件付き確率 $P(Class_i|Object)$ を計測する．以上の値により，各グリッドセル内で予測された各バウンディングボックスに対して，物体が存在する可能性とその位置精度を統合的に評価するスコアとして，最終的に各クラスごとに以下で定義される S_i を算出する手順となっている．つまり，グリッドの数 $S \times S$ 毎に，そして，各バウンディングボックスの数B毎に算出される．

$$S_i = P(Class_i|Object) \times C = P(Class_i) \times IoU(b, b^*).$$

YOLOの信頼度スコアや検出スコアに限らず，画像認識全般においての性能評価でも，先程のIoUという指標は広く使用されている [20]．この場合，IoUは一般的に0.5以上を，高い精度を求める場合で0.8以上を設定し，それを満たす場合にそのオブジェクトは正しく検出されたとみなす．一方で，IoUが閾値を下回る場合は，そのオブジェクトは正しく検出されなかったものとして扱われる．これに基づいて，物体検出の性能が評価される．この評価基準に基づき，検出結果はTrue Positive（TP），False Positive（FP），False Negative（FN），およびTrue Negative（TN）の4つに分類される．TPは実際に存在する物体を正確に検出した場合を指し，FPは存在しない物体を誤って検出した場合を表す．一方，FNは実際に存在する物体を検出できなかった場合を指し，TNは検出対象外の領域を検出しなかった場合を意味する．IoUに基づく評価は，画像認識の精度を測定する上で一定の有効性を持つが，いくつかの課題も存在する．特に，IoUはバウンディングボックスの重なり具合を評価する指標であり，物体間の位置関係や動作条件を考慮できないという限界がある．また，特にFalse Negativeの発生は安全上重大なリスクを引き起こす可

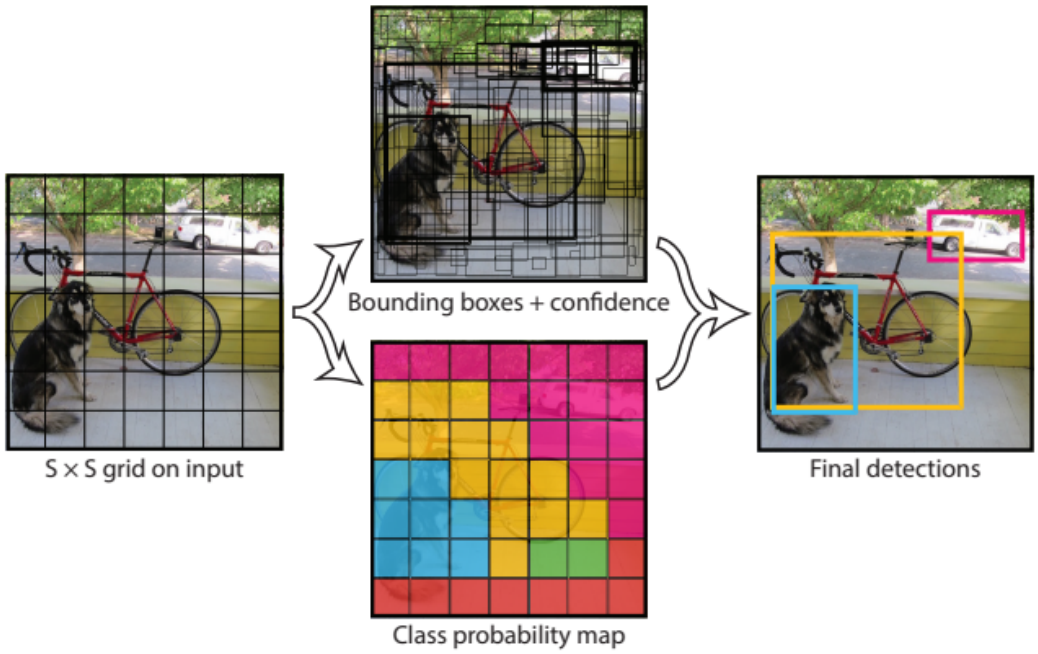

図 2.5: YOLO における検出アルゴリズムの概要 [3]

能性があるにもかかわらず，これを徹底的に評価する枠組みは十分に整備されていない．さらに，自動運転システムの画像認識テストには，テストケースの網羅性が不足しているという課題もある．高速道路での車線変更や交差点での歩行者回避といった安全クリティカルな状況が十分にテストされない場合があり，安全要件を満たすかどうかの評価が不完全となるリスクがあることがあげられている．

2.5 ソフトウェア仕様の品質特性

ソフトウェア開発において，仕様は要求と設計・実装の間を橋渡しする中心的な役割を担っている．本論文では，自動運転システムの画像認識における仕様を明らかにしつつ作成することを試みる．そこで，ここでは，作成する仕様の良し悪しの基準として既存のソフトウェア仕様における品質特性に焦点を当てて説明する．

仕様の役割として，関係者間の共通理解の形成や，設計・実装の指針の提供，検証・テストの基準の提示などがあげられるが，これらの観点から，仕様の品質はソフトウェア全体の品質に直結するものであり，慎重に管理されるべき対象である．仕様の品質に関するガイドラインとして，IEEE 830-1998 が広く用いられている [30]．この規格では，良い仕様を満たすべき特性として，正当性 (Correct)，一貫性 (Consistent)，無曖昧性 (Unambiguous)，完全性 (Complete)，修正容易性 (Modifiable)，検証容易性 (Verifiable)，順序付け (Ranked for Importance and/or Stability)，追跡可能性 (Traceable) の 8 つの特性が示されている．正当性とは，仕様がシステムに必要とされる要求を正確に記述していることを表す性質であり，不必要または誤った要求が含まれていないことを意味する．正当性が確保されていない場合，実装されたシステムがユーザの意図と異なる動作をする危険性がある．一貫性とは，仕様内に矛盾する記述が存在しないことを表す性質である．たとえば，ある機能が

「実行する」と記されている一方で「実行しない」と別の場所で記述されているような場合，一貫性が欠けていると判断される．仕様が一貫していない場合，設計や実装，さらにはテストにおいて混乱を招く可能性がある．無曖昧性とは，各要求がただ一つの意味にのみ解釈できるよう記述されていることを表す性質である．自然言語で仕様を記述する場合，曖昧な表現や主観的な語句が含まれやすいため，注意深く記述する必要がある．無曖昧性が保たれていない仕様は，異なる解釈に基づく不整合な実装やテストを生む原因となる．完全性とは，すべての要求が網羅されており，仕様に抜けや漏れがないことを表す性質である．特に，入出力条件，例外的状況，制約条件などが過不足なく記述されていることが求められる．完全でない仕様は，後の開発工程での手戻りや誤動作の原因となる．修正容易性とは，仕様が変更される際に，変更箇所が容易に特定でき，かつ，他の記述との依存関係が明確であることを表す性質である．この特性は，仕様の保守性や拡張性に密接に関わる．構造的に整理された記述や一貫した用語の使用などが，修正容易性を高める上で重要となる．検証容易性とは，仕様に記述された要求が，実装後にテストや分析によって満たされているかどうか確認可能であることを意味する.「十分に高速である」や「使いやすい」といった定性的な表現は検証困難であり，数値的または論理的に検証可能な形式で記述されることが望ましいとされている．順序付けとは，各要求についてその重要度や安定性に応じて優先順位を付けられることである．これにより，要求変更時の影響範囲を把握しやすくなり，開発やテストの優先順位決定の基準としやすくなるとされている．最後に，追跡可能性とは，各要求がどのような上位要求に由来するのかをたどれるかや，要求と対応する設計要素やテストケースとの関係かを追えるかを表す性質である．これにより，変更管理や影響分析を効率的に行うことが可能となる．以上の特性を満たすことは，仕様を記述する上での実務的な指針となるものであり，開発・テスト・保守の各工程においての仕様の価値として重要な基準となる．

現在では，IEEE 830-1998 の内容を踏襲しつつ，ISO/IEC/IEEE29148:2011 に置き換えられている [31]．この規格では，個々の要求と要求群を明確に区別し，それぞれに対して適切な品質特性が定義されている．たとえば，一貫性（Consistent）は，個々の要求としての一貫性と，要求群への要求としての一貫性が別の特性として求められるなどがある．本論文では，記述特性の説明のしやすさを重視し，古くから広く用いられている IEEE 830 に基づいて説明を与える．

2.6 機能テストと形式仕様記述言語

機能テストとは, ある機能の実装が正しく動作することを検証する手法である．プログラムのソースコードや内部構造に基づいたホワイトボックステストに位置づけられるコードベーステストに対して，一般に，機能テストは，システムの内部構造に依存せず，入力と期待される出力という観点からシステムの振る舞いを評価するブラックボックステストの一種として位置付けられる．そのため，開発者の実装意図とは独立した観点からテストケースを作成でき，特定のプログラムコードに依存しないテストが可能となる．機能テストでは，設計ドキュメントや設計仕様に基づいて機能の確認を行うのだが，その仕様そのものが曖昧であったり，一貫性がなかったりする場合，不適切なテストケースが設計されるリスクが生じる．そのため，本論文では，作成する機能仕様の無曖昧性を担保するための手法の一つとして，形式仕様，そして形式仕様を記述するための形式仕様記述言語の適用を試みる．

形式仕様記述言語とは，数学的な記法を用いてソフトウェアやシステムの仕様を正確に記述する手法であり，仕様の曖昧さを排除し，設計段階から厳密な検証を可能にするものである．このアプローチは，特に安全クリティカルな分野において採用され，航空宇宙や

鉄道，自動車産業などで活用されてきた．形式仕様記述言語が注目されるようになった背景には，1980 年代から 1990 年代にかけて発生した多くのソフトウェア事故がある．これらの事故の多くは，設計段階での仕様の曖昧さや不整合が原因であった．例えば，航空分野では仕様の不備が原因でフライト制御システムが誤作動を引き起こした事例があり，これを教訓に形式的な仕様記述の必要性が高まった．自動車産業においても，システムの複雑化に伴い，機能安全規格 ISO 26262 が制定され，仕様の形式的記述が推奨されるようになった．

これまでの形式仕様記述言語の多くは，システムの動作や状態遷移に焦点を当てて設計されてきた．代表的なもとのして，Z 言語や VDM-SL（Vienna Development Method Specification Language）などがある．Z 言語は，集合論や述語論理を用いてシステムの状態や操作を記述する形式仕様記述言語である [32]．例えば，図 2.6 は Z 言語によって単純な口座の入金システムの仕様を記述した例である．Z 言語では，システムの状態や操作を状態スキーマや操作スキーマなどといったスキーマと呼ばれる形式で記述する．図 2.6 の上に記述している BankAccount は，銀行講座の状態スキーマであり，口座残高を表すために balance という自然数の変数が定義を持ち，balance は必ず 0 以上であるという条件によって制約を明示している．また，図 2.6 の下に記述している Deposit は，入金の操作スキーマである．*Delta*BankAoount が先ほど定義した BankAccount の状態変化を意味し，入力として入金額 amount?が与えられ，新しい残高 balance’ が旧残高 balance に入金額を加えたものであることを定義している．このように，Z 言語ではシステムの状態と操作を数学的に厳密に記述することができ，仕様の不整合や曖昧さを排除し，形式的な検証の基盤として用いられる．

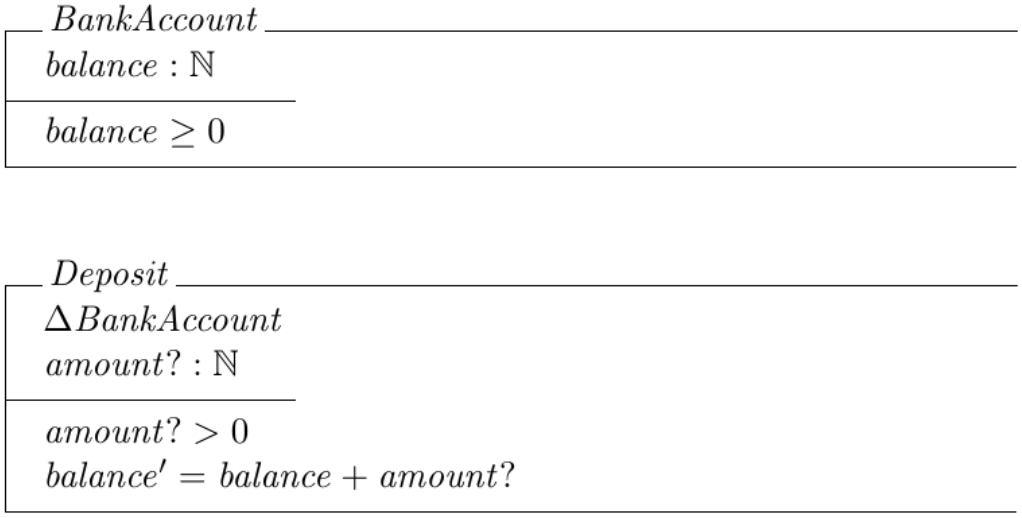

図 2.6: Z 言語による口座の入金システムの仕様の例

また，VDM-SL は，状態変数と操作を用いて仕様を厳密に定義する形式仕様記述言語であり，システムをモジュールという単位で記述し，データ構造や操作を形式的に定義することができる [33]．例えば，図 2.7 は VDM-SL によって単純な口座の入金システムの仕様を記述した例である．図 2.7 では，BankAccount というモジュールとして仕様を定義しており，7 行目,8 行目で BankAccount という型を定義している．具体的には，この型は自然数として口座残高を表す balance を持つことを定義している．そして，10 行目から 14 行目では，先程定義した BankAccount 型の状態変数 acc を定義し，acc の不変条件は 0 以

上であることと，初期状態ととして常に口座残高は 0 であることを示している．16 行目から 30 行目でモジュールで定義された操作について記述している．この仕様では，銀行口座を初期化するための操作と，入金操作についての仕様を記述している．どちらの操作でも，引数と返り値についてのシグネチャ，操作内容，事前条件，事後条件について定義している．具体的には，入金操作を表す Deposit では，シグネチャは，自然数型の金額を入力として，返り値はないことを定義しており，操作内容は，現在の口座残高に入力された金額を加算させて，状態 acc を更新することを定義している．そして，事前条件として，入力される amount は常に 0 以上であることを示し，事後条件として，操作後の口座残高は，常に操作前の口座の残高に入力された amount を加えた値であることを保証することを記述している．このように，VDM では，数学的な型と操作の定義からなるモジュールを通じて，システムがどのように状態を変化させるかを厳密に記述できるため，実装前に仕様の整合性や安全性を検証する基盤として用いられる．

```
 1: module BankAccount
 2:
 3: exports all
 4:
 5: definitions
 6:
 7: types
 8:   BankAccount :: balance: nat;
 9:
10: state AccountState of
11:   acc: BankAccount
12:   inv mk_AccountState(acc) == (acc.balance >= 0)
13:   init s == s = mk_AccountState(mk_BankAccount(0))
14: end
15:
16: operations
17:
18:   initAccount: () ==> ()
19:   initAccount() ==
20:     acc := mk_BankAccount(0)
21:   pre true
22:   post acc.balance = 0;
23:
24:   Deposit: nat ==> ()
25:   Deposit(amount) ==
26:     acc := mk_BankAccount(acc.balance + amount)
27:   pre amount > 0
28:   post acc.balance = (acc~.balance) + amount;
29:
30: end BankAccount
```

図 2.7: VDM によるカウンタモジュールの仕様の例

自動運転システムの認識モジュールにおける画像認識では，物体間の位置関係や動作条件を明確に仕様として記述する必要がある．しかし，従来の形式仕様記述言語では，これらの空間的な情報を記述するための明確な枠組みが不足している．例えば，Z 言語や VDM は，抽象的な関数や集合として仕様を記述するため，車両間の相対的な位置や進行方向と

いった画像上の具体的な要素を自然に表現することが難しい．また，幾何学的な情報を取り扱うための拡張が施された形式仕様記述言語もあるが，これらは主にロボティクスや 3D 空間のシミュレーションに焦点を当てており，自動運転システムの画像認識のように，入力として考えられる多様な画像に対して仕様を定義するには，注目している抽象度の違いにより適していない．本論文でも，安全クリティカルなソフトウェアである自動運転システムにおいて，安全性を向上させるためには，形式仕様記述言語を用いて，形式仕様を設計すべきであるという立場にたっている．しかし，前述したように，画像認識システムの仕様には，既存の形式仕様記述言語では難しい点が存在するため，本論文では，画像認識に特化した画像情報による機能仕様を記述するための言語 BBSL を提案する．

2.7　空間関係の形式化技術

本論文で提案する BBSL は，物体間の位置関係や動作条件を数学的に記述するために，次節で説明する区間演算という数学的体系を参考にしているが，区間演算以外にもこのような関係を記述するために活用できうる既存技術が存在している．本節では，空間関係を形式的に記述するために利用できる可能性のある主要な技術として，二項トポロジー関係（Binary Topological Relationship: BTR），および幾何学的技術（Geometric Descriptions）の詳細について述べる．

まず，BTR は，空間オブジェクト間の基本的な位置関係を形式的に記述するために，数学的な枠組みとして提案されたものである [34]．具体的には，2 つの空間オブジェクトの「内部（Interior）」「境界（Boundary）」「外部（Exterior）」の 3 要素を基に，位置関係を定義するものである．図 2.8 は，BTR で記述することができる，二次元オブジェクト間の 8 つの基本的なトポロジー的関係の例を示している．この図は，データベース上で空間データを扱うための Oracle Spatial のリファレンスから抜粋している [4]．1 つ目は，Contains と呼ばれ，オブジェクト A がオブジェクト B の境界および内部を完全に包含する関係である．これは，図 2.8 の左上に示されているものがこの関係の例に該当し，A の内部に B が完全に収まっている関係であり，A の境界に接することはない．2 つ目は， Covers と呼ばれ，オブジェクト A がオブジェクト B を覆い，その境界が B の外部に広がる場合を指す．この関係は包含に似ているが，図 2.8 の上段中央に示されているものがこの関係の例に該当し，B が A の境界に接することもある．3 つ目は，Touch と呼ばれ，2 つのオブジェクト A と B が互いに境界で接触しているが，内部は重ならない関係である．図 2.8 の右上に示されているものがこの関係の例に該当し，A と B が境界でのみ接触している．4 つ目は，Overlap by Intersection と呼ばれ，オブジェクト A と B の内部が一部重なっており，完全には包含しない関係である．図 2.8 の中段左に示されているものがこの関係の例に該当し，A と B が部分的に交差しており，部分的に重なっている．5 つ目は，Overlap by Disjoint と呼ばれ，オブジェクト A と B の境界が交差するが，内部が重ならない関係である．図 2.8 の中段右に示されているものがこの関係の例に該当し，B は境界のみで内部が存在してないため，境界が交差するが，内部が重ならない例になっている．6 つ目は，Equal と呼ばれ，オブジェクト A と B が形状，大きさ，位置において完全に一致する関係である．図 2.8 の下段左に示されているものがこの関係の例に該当し，A の境界と B の境界，内部，外部がすべて一致している．7 つ目は，Disjoint と呼ばれ，オブジェクト A と B が完全に分離しており，内部や境界が一切交わらない関係である．図 2.8 の下段中央に示されているものがこの関係の例に該当し，A と B が完全に離れている．8 つ目は，On と呼ばれ，オブジェクト B が A の境界上に乗っている関係を指す．これは接触の一種であり，どちらかが特定の点や境界で定義される場合に成立する．図 2.8 の下段右に示されているものがこの関係の例に該当し，B は境界であり，A の境界の上にのっている．以上の枠組みである BTR は，静的な空間データの管理や解析を効率化する手法として広く利用されており，主に地理情報システムや 3D 空間モデリングにおいて利用される．都市環境での建物や道路の配置解析，空間オブジェクトの接触状態を記述するための重要な手法である [35].

対して，幾何学的技術では，物体の相対的な距離，角度，速度，方向といった幾何学的要素を基に空間関係をモデル化する．例えば，単眼カメラを用いて高速道路上の車両と障害物の位置関係を幾何学的に記述する方法では，カメラ視点から得られた画像を幾何学モデルに基づいて解析し，対象物体との距離や角度，相対速度を計算することで，車両間の空間的な関係を明確に示すことが可能となる [5]．図 2.9 は，前方車両（lead vehicle）と後続車両の位置関係を幾何学的記述により，形式的にモデル化された例である．ここでは，車両間の相対距離 d_{image} や，カメラ視点からの角度 α および β，カメラ高さ h_1 および h_2，

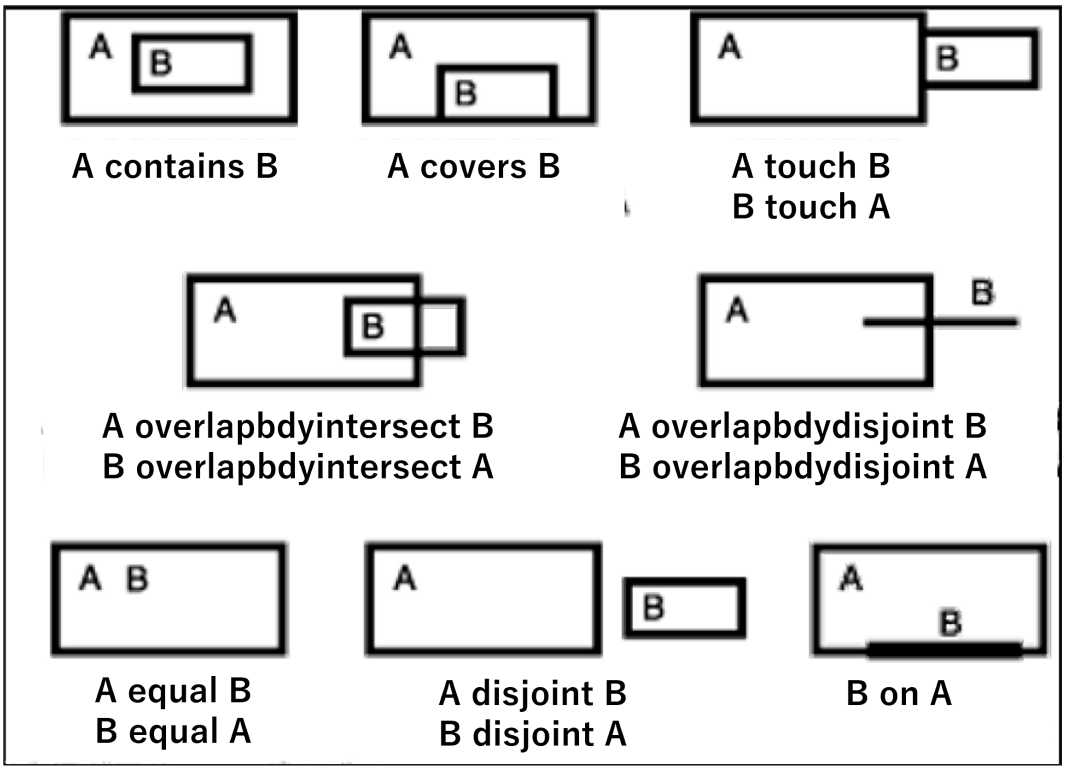

図 2.8: BTR の関係一覧 [4]

車体長 b といった幾何学パラメータを使用して，空間的な位置関係を定義し，物体間の相対位置を数式化することが可能となる．このような幾何学的手法は，車両間の衝突予測や車間距離の制御など，動的なシステム挙動の解析や評価に適用される．例えば，車両がどの時点で減速すべきか，前方車両の速度変化にどのように反応すべきかといった具体的な挙動設計が，幾何学モデルを基に定量的に実施される．

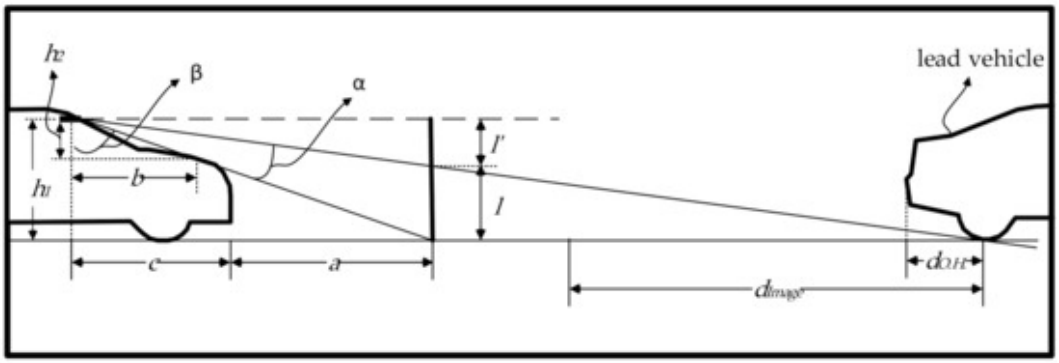

図 2.9: 前方車両と後続車両の位置関係を幾何学的に示す例 [5]

2.8 区間演算

区間演算は，数値の不確実性や丸め誤差，測定誤差などを考慮した演算を行うための実数上の演算体系である．数値を単一の値ではなく，下限と上限からなる区間 $[a, b]$ として表現することで，計算結果に生じる誤差を明示的に扱い，誤差付き精度保証演算を実現することを目的としている [6]．この手法は，数値解析やロバスト制御，科学技術計算など，信頼性が求められる分野において特に重要視される．また，誤差の伝播や数値的不確実性を定量的に扱うことができるため，システム設計や評価の初期段階から，結果の信頼性を保証するための基盤としても利用されている．区間演算の考え方は，入力として与えられる値に不確実性が存在する場合，計算結果も単一の値ではなく，ある範囲内にあると考えるアプローチである．これにより，計算中に発生する丸め誤差や測定誤差，外乱などが，結果にどのように影響するかを安全に評価することができる．前節で説明したように物体間の位置関係や動作条件を数学的に記述するためにいくつかの既存技術が存在している中で，本論文で提案する BBSL では，画像認識の結果として得られる 2 次元のバウンディングボックスやいくつかの演算子の定義において，この区間演算の定義と類似しているものが存在している．ただし，誤差付き精度保証演算を目的とした区間演算に対して，単純に画像上の区間やバウンディングボックスを表現することを目的とした BBSL では，その理念や考え方は根本から大きく異なる．ここでは，本論文で定義し説明を行う BBSL との違いを理解するための準備として，区間演算の基礎理論および関連する定義と例を示す．

初めに，区間の基本的な定義を与える．

定義 2.1. 区間 X とは，$\underline{X}$ と $\overline{X}$ を区間の端点として，以下で定義される．

$$X = [\underline{X}, \overline{X}] = \{x \in R : \underline{X} \leq x \leq \overline{X}\}$$

例 2.1. $\sqrt{2}$ など，コンピュータ上で近似して扱わざるを得ない無理数に対して，$\sqrt{2} = [\underline{\sqrt{2}}, \overline{\sqrt{2}}] = [1.4, 1.5]$ のように区間を用いることで表現することが可能となる．

この定義により，区間は単一の実数ではなく，実際の数値の不確実性を反映した範囲として扱われる．そのため，測定値や計算結果に含まれる丸め誤差を区間として表現することで，最悪の場合や最良の場合の値を同時に評価できるようになる．

また，区間演算では，特殊な区間として縮退区間というものが定義されている．

定義 2.2. 以下を満たす区間 x を縮退区間という．

$$\forall x \in R, x = [x, x]$$

つまり，すべての実数は縮退区間という特殊な区間であると言える．

定義 2.3. 区間 X,Y について，同値関係は以下で定義される．

$$X = Y \Leftrightarrow \underline{X} = \underline{Y}\ and\ \overline{X} = \overline{Y}$$

例 2.2. 区間 $a = [150, 200]$，$b = [150, 200]$，$c = [150, 210]$，$d = [160, 200]$ が与えられた時，関係 $a = b$ は成り立つが，$a = c$，$a = d$ は成り立たない．

区間は実数の集合で定義されているため，集合の演算子 $\cup, \cap$ を持ちることが可能だが，演算子 $\cup$ は区間上で閉じた演算子ではない．そのため，閉じた区間上で閉じた演算子となるように Interval Hull$\underline{\cup}$ を定義している．

定義 2.4. 区間 X，Y について，Interval Hull$\underline{\cup}$ は以下で定義される．

$$X \underline{\cup} Y = [min\{\underline{X}, \underline{Y}\}, max\{\overline{X}, \overline{Y}\}]$$

この定義により，一般的な集合上の和集合 $X \cup Y$ が区間として閉じない場合でも，$X \cup Y \subseteq X \underline{\cup} Y$ は常に成り立つため，最小の閉じた区間として $X \underline{\cup} Y$ を得ることができる．

次に，区間に対する計算を行うための関数を定義する．

定義 2.5. 区間 X について，幅 $w(X)$，絶対値 $|X|$，中間点 $m(X)$ は以下で定義される．

$$w(X) = \overline{X} - \underline{X}$$
$$|X| = max\{|\underline{X}|, |\overline{X}|\}$$
$$m(X) = 1/2(\underline{X} + \overline{X})$$

例 2.3. 区間 $a = [130, 200]$，$b = [150, 190]$，$c = [180, 300]$ が与えられた時，それぞれ，$w(a) = 70$，$w(b) = 40$，$w(c) = 120$ となる．

幅 $w(X)$ は，区間の広がりを示し，計算における不確実性の大きさや丸め誤差の影響を定量化するのに利用される．たとえば，センサの測定結果の信頼性評価や，数値計算における誤差伝播の解析で重要な指標となる．また，絶対値 $|X|$ は，区間の端点の大きさを示し，特に正負の両側での誤差の大きさを評価する際に用いられる．例えば，数値計算で安全な境界条件を設定する際に．どちらの端点がより大きな誤差の可能性を持つかを判定するために役立つ．そして，中間点 $m(X)$ は，区間の代表値として扱われ，近似値や代表的な値を算出するのに用いられる．特に，シミュレーションや最適化問題において，区間の中点を用いて大まかな値を計算し，そこからさらなる詳細な解析を行うことがある．

次に，2 つの区間上の二項関係を定義する．

定義 2.6. 区間 X，Y について，区間上の二項関係 $<, \subseteq$ は以下で定義される．

$$X < Y \Leftrightarrow \overline{X} < \underline{Y}$$
$$X \subseteq Y \Leftrightarrow \underline{Y} \leq \underline{X}\ and\ \overline{X} \leq \overline{Y}$$

例 2.4. 区間 $a = [150, 200]$，$b = [250, 300]$，$c = [190, 260]$，$d = [130, 200]$ が与えられた時，関係 $a < b$,$a \subseteq d$ は成り立つが，$a < c$，$c < b$,$d \subseteq a$ は成り立たない．区間 $a = [130, 200]$，$b = [150, 190]$，$c = [180, 300]$ が与えられた時，関係 $b \subseteq a$ は成り立つが，$a \subseteq b$ や $b \subseteq c$ は成り立たない．

このような二項関係は誤差の評価やアルゴリズムの安定性の解析において利用されている．例えば，二項関係 $<$ は，安全マージンの評価や，区間同士の順序付けに利用される．また，包含関係 $X \subseteq Y$ のように，区間 X のすべての値が区間 Y に含まれる場合の判定は，誤差評価において，ある区間が安全域に完全に収まっているかなどを計算などに利用されている．

次に，区間上の四則演算を定義する．区間上の四則演算も，各区間内のすべての実数同士の演算結果の集合として定義される．

定義 2.7. 区間 X，Y について，$\bigcirc = \{+, -, \cdot, /\}$ とすると，四則演算子は以下で定義される．

$$X \bigcirc Y = \{x \bigcirc y : x \in X, y \in Y\}$$

多くの区間演算の応用では，1 次元の数値だけでなく，複数の次元にまたがる情報を扱う必要がある．そのため，区間演算では，これまで定義してきた区間，演算子を多次元区間として拡張することができる．

定義 2.8. 区間 X_i,$i = 1, ..., n$ とすると，多次元区間 X は以下で定義される．

$$X = (X_i, ..., X_n)$$

さらに，各演算子，および，関数は実数ベクトル $x = (x_1, ..., x_n)$ とすると以下で定義される．

$$\begin{aligned}
x \in X &\Leftrightarrow \text{任意の i について } x_i \in X_i \\
X \cap Y &= (X_1 \cap Y_1, ..., X_n \cap Y_n) \\
X \subseteq Y &\Leftrightarrow \text{任意の i について } X_i \subseteq Y_i \\
w(X) &= max\{w(X_i) \mid \text{任意の } i\} \\
m(X) &= (m(X_1), ..., m(X_n)) \\
||X|| &= max\{|X_i|| \text{ 任意の } i\}
\end{aligned}$$

例 2.5. バウンディングボックス $a = ([350, 400], [200, 300])$，$b = ([390, 500]$ が与えられた時，$a \cap b = ([390, 400], [100, 200])$ となる．特に，二次元の区間 $X, n = 2$ について，区間 X_1, 区間 X_2 および，関数 $w(X)$，$m(X)$，$||X||$ を表した図を図 2.10 に示す．この図は，書籍 Introduction to Interval Analysis で用いられている図を抜粋している [6]．X は図に示すようにバウンディングボックスを形作る二次元の区間となり，X_1 は x 軸方向の区間，X_2 は y 軸方向の区間となる．また，$w(X)$ は，長い方の区間として $w(X_2)$ の値となり，$m(X)$ はこのバウンディングボックスの中点と等しく，$||X||$ は，$\overline{X_2}$ の値となる．

このように，区間演算は区間，演算子を多次元に拡張できるため，センサの複数軸データなどにも利用されている．

以上のように，区間演算は単に数値の範囲を扱うだけではなく，計算に含まれる不確実性や誤差を明示的に管理するための数学的手法である．特に，誤差伝播の解析や数値計算の信頼性向上，またシステムの安全性保証において，区間演算は極めて有用な手法として広く応用されている．本論文で提案する BBSL は，この 2 次元の区間が，画像認識の検出やグラウンドトゥルースデータで用られるバウンディングボックスを表す形になることから，区間計算における 2 次元の区間をバウンディングボックスの定義としている．しかしながら，区間計算の目的は誤差付き精度保証演算の実現であるため，この定義で用いられる演算子には，オブジェクト間の位置関係の記述とは関係がないものも多く含まれる．

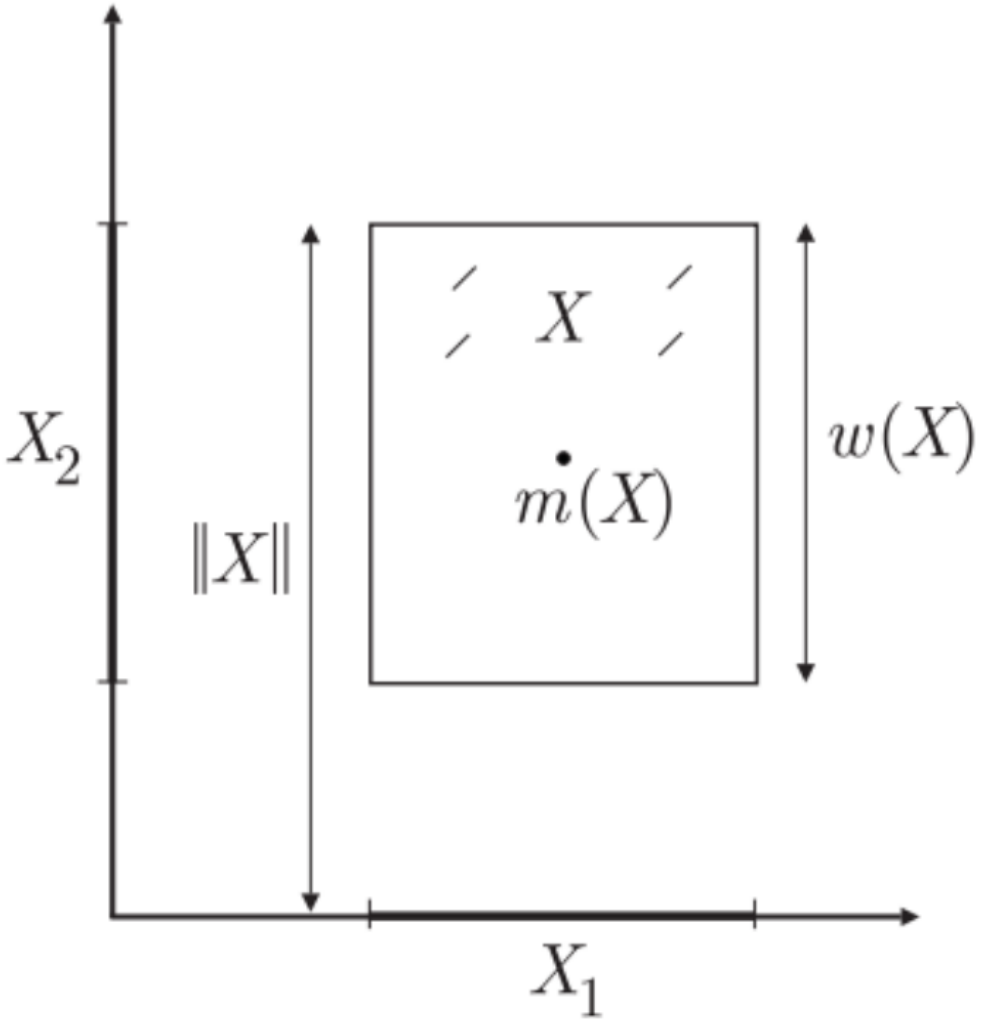

図 2.10: 2 次元の区間と各関数の関係の例 [6]

2.9 テストカバレッジ

テストカバレッジは，ソフトウェアテストや安全性評価において，テストの網羅性を測定するための重要な指標である．具体的には，対象の要素（ソースコード，条件，機能，仕様）がテストケースによって「カバー（網羅）」された割合を定量的に測定する．カバレッジを用いることで，テストが不充分である範囲や，未検証の領域を特定できることが期待できる．本論文で提案する BBSL 仕様ベースカバレッジでは，BBSL で記述された使用上の条件がテストケースによって網羅された割合として与えるが，従来のソースコードを対象に測定されるコードカバレッジという指標を参考に定義している．そこで，ここでは，コードカバレッジの思想，種類に焦点をあてて説明する．

ソフトウェアテストにおけるコードカバレッジは，プログラムのソースコードがテストケースによってどれだけ網羅されているかを評価する手法である．コードカバレッジには複数の種類が存在し，それぞれ異なる要素に着目してテストの網羅性を測定する．以下に，代表的なコードカバレッジとして，ステートメントカバレッジ，デシジョンカバレッジ，コンディションカバレッジ，コンディションデシジョンカバレッジ，複合条件カバレッジの定義とその例を示す [36].

ステートメントカバレッジとは，プログラム内のすべてのステートメントが少なくとも一度実行されることを保証するカバレッジ基準であり，基本的なテストの網羅性を測る指標として利用される．ステートメントとは，プログラム内の個々の命令や操作を指し，C 言語の場合，if 文や while 文のような制御構造も含まれる場合がある．各ステートメントが最低 1 回実行されることで，コード全体が最低限のテストを受けていることを示す．

定義 2.9. ステートメントカバレッジは，カバーされたステートメントの数 $|S_c|$，プログラム内の総ステートメント数 $|S_a|$，到達不可能なステートメントの数 $|S_u|$ として，以下で定義される．

$$\frac{|S_c|}{(|S_a| - |S_u|)}$$

例 2.6. 図 2.11 は，疑似コードで書かれた小さなコードの例である．この例でのステートメントは制御構造を含まない場合，2 行目，8 行目，11 行目，14 行目にあたり，fuction(5,7)，function(5,2)，function(-3,7) のようなテストケースで全てのステートメントがカバーされ，ステートメントカバレッジは 100%となる．

デシジョンカバレッジとは，プログラムの各分岐（if 文や switch 文など）の条件が true と false の両方を評価されることを保証するカバレッジ基準である．これにより，すべての分岐における true,false のパス (決定パス) が実行されることを確認できる．例えば，if(x $>$ 5) という条件があった場合，x $>$ 5 が true と false の両方になるテストケースを用意することで，このカバレッジ基準を満たす。

定義 2.10. デシジョンカバレッジは，true,false が共にカバーされた分岐の数 $|D_c|$，プログラム内の分岐の総数 $|D_a|$，到達不可能な分岐の数 $|D_u|$ として，以下で定義される．

$$\frac{|D_c|}{(|D_a| - |D_u|)}$$

例 2.7. 図 2.11 のコードの例では，分岐は 4 行目，5 行目にあたり，fuction(5,7)，function(5,2)，function(-3,7) のようなテストケースで全ての決定パスがカバーされ，デシジョンカバレッジは 100%となる．デシジョンカバレッジを 100%にするためのテストケースは，仮に図 2.11 の 11 行目の result=0 が記述されていない場合でも影響はないが，ステートメントカバレッジの場合には，当然影響をうけ，fuction(5,7) と function(5,2) のみで 100% に達するようになる．

コンディションカバレッジは，各単純条件（例えば if(x $<$ 5 && y $>$ 3) に対する x $<$ 5 や y $>$ 3 など）が少なくとも一度 true と false の両方を評価されることを保証するカバレッジ基準であう．これは，個々の条件が適切にテストされていることを保証し，誤った条件評価によるバグの発見に役立つ．

定義 2.11. コンディションカバレッジは，true,false が共にカバーされた単純条件の数 $|C_c|$，プログラム内の総単純条件数 $|C_a|$，到達不可能な単純条件の数 $|C_u|$ として，以下で定義される．

$$\frac{|C_c|}{(|C_a| - |C_u|)}$$

例 2.8. 図 2.11 のコードの例では，単純条件は x $>$ 0，y $>$ 0，x + y $>$ 10 であり，fuction(-2,5)，function(2,5)，function(20,-5) のようなテストケースで全ての単純条件が true,false 共にカバーされ，コンディションカバレッジは 100%となる．ただし，このテストケースの場合，(y $>$ 0 && x + y) 全体は，true と評価されないため，ステートメントカバレッジや，デシジョンカバレッジは 100%とならない．

コンディションデシジョンカバレッジは，コンディションカバレッジとデシジョンカバレッジを統合した基準であり，各単純条件が true と false の両方を取ることに加え，各分岐の条件もすべての可能な結果を取ることを保証する．これにより，条件ごとの評価とその影響をより包括的にテストできる．

定義 2.12. コンディションデシジョンカバレッジは，true,false が共にカバーされた分岐の数 $|D_c|$，プログラム内の分岐の総数 $|D_a|$，到達不可能な分岐の数 $|D_u|$．そして，true,false が共にカバーされた単純条件の数 $|C_c|$，プログラム内の総単純条件数 $|C_a|$，到達不可能な単純条件の数 $|C_u|$ として，以下で定義される．

$$\frac{(|C_c| + |D_c|)}{(|C_a| - |C_u| + |D_a| - |D_u|)}$$

例 2.9. 図 2.11 のコードの例では，fuction(-1,7)，function(5,-2)，function(5,8) のようなテストケースで，コンディションデシジョンカバレッジは 100%となる．

複合条件カバレッジは，複数の単純条件が含まれる複合条件のすべての可能な真理値の組み合わせが網羅されることを保証するカバレッジ基準である．これにより，より詳細な条件分岐のテストが可能となり、異なる条件の相互作用に対するバグの発見が容易になる．その定義は，可能な条件の組み合わせの総数に対する網羅した条件の組み合わせの数で定義され，C 個の単純条件がある場合，すべての条件の組み合わせを網羅するためには一般に 2^C 通りあるが，ネストや論理的にありえない組み合わせがある場合によって分母が調整される場合が多い．例えば，図 2.11 のコードの場合では，4 行目の x $>$ 0 が false のとき，y $>$ 0 や x + y はネスト構造により評価されないため，x $>$ 0 が false に対しては組み合わせをテストできない．そのため，今回の例では，fuction(-1,7)，function(5,-2)，function(13,-2)，function(5,1)，function(5,10) のようなテストケースで，複合条件カバレッジは 100%となる．

コードカバレッジにおいては，すべての条件の組み合わせを網羅する複合条件カバレッジが，理論上は最も完全なカバレッジであるが，カバレッジを満たすためのテストケースの最小数が指数的に増大するという問題点がある．そのため，複合条件カバレッジのようにすべての組み合わせを取るのではなく，優先的に調べる必要のある重要な組み合わせのみを考慮することで，効率的かつ高い網羅性を実現するために提案された改良条件判断カ

```
1  function example(x, y) {
2      result = 0
3
4      if (x > 0) {
5          if (y > 0 && x + y > 10) {
6              result = 1;
7          } else {
8              result = −1;
9          }
10     } else {
11         result = 0;
12     }
13
14     return result;
15 }
```

図 2.11: コードカバレッジのためのサンプルコード例

バレッジ (MC/DC) と呼ばれるカバレッジがある．優先的に調べる必要のある重要な組み合わせは，システムの安全性や複雑性に応じた網羅性の要求が異なるため，開発しているソフトウェアの要件によっていくつかの種類の MC/DC が存在している．例えば，一般的に最も広く使用されている標準的な MC/DC は，条件式内のすべての要素が個別にテスト結果に影響を与えることを確認し，その網羅性を最小限のテストケースで達成することを目的としている．一方で，システムの重要性やテストの精度向上をより追求する必要のあるケースでは，条件 MC/DC やマスク付き MC/DC などと呼ばれる派生手法が用いられる．

このように，従来のコードカバレッジには，焦点をあてる要素によって複数の種類が存在している．本論文で提案する BBSL 仕様ベースカバレッジにおいても，要素によって複数のカバレッジを提供し，複合条件カバレッジを満たすことが現実的でない仕様のために，BBSL で記述した機能仕様を用いた機能テストにおいて優先度の高い条件の組み合わせを特定し，独自の MC/DC カバレッジを提案する．

第3章　BBSL

3.1　概要

節 1.1.3 で説明したように，自動運転システムの仕様を設計する際は，走行環境の多様性から一般にはシナリオ毎に整理して設計される．シナリオでは，車両の位置や挙動を表現するために，視覚的な図が用いられることが多い．しかし，そのような図は非形式的なものであり，異なる開発者の間で齟齬が生じる可能性があるため，このように表記された仕様に基づいてテストを行うことは安全性を評価する上で妥当とは言い難い．また，既存の自動運転システムの仕様とは異なり，画像情報に対する自動運転システムのレスポンスという観点での仕様は現状不明である．しかし，少なくとも人間は自動運転システムが停止すべきかどうかなどの機能の判定を画像情報のみからでも直感的に行うことができる場合がある．例えば，節 1.1.2 で説明した，図 1.1 の場合，画像情報から車両が進行方向を塞いでいるかどうかや，車両と適切な車間距離を維持できているかどうかから停止すべきかどうかを判断できる．これを考慮すると，人間がどのように画像上の情報から判断しているかは，図 1.1 に示した進行方向や適切な車間距離などのような画像上の特定の領域に対して，車両や歩行者などのオブジェクトの相対位置を判断基準に含んで評価していると推測できる．

そこで，こうした人間の直感的な判断基準のうち，画像上の特定領域に対するオブジェクトの相対位置を形式的に記述するために，BBSL という形式仕様記述言語を提案する．BBSL は，自動運転システムの OEDR 仕様を形式的に記述するための言語であり，画像認識の入力画像に対して，自動運転システムのレスポンスがどのように対応づけられるかを定義することを目的としている．つまり，BBSL で記述する OEDR 仕様では，進行方向や適切な車間距離のように画像上に暗に存在する特定領域をセマンティック情報と対応付け，その特定領域とオブジェクトの相対位置によって，自動運転システムの取るべきレスポンスを定義する．また，BBSL では車両などの画像上のオブジェクトをバウンディングボックスで表現する．本研究のテスト対象の画像認識の出力や既存のデータセットは，検出対象のオブジェクトをバウンディングボックスで表現するデータ形式であるため，それらと対応を確認しやすくなることが期待できる．さらに，仕様の抽象度に関しても考慮する必要がある．画像認識の入力で想定される画像は膨大であるため，細部まで正確に表現しすぎてしまうと，現実的なコストで仕様を記述することができなくなってしまう．そのため，画像上のオブジェクトを単純化して，自動運転システムのレスポンスと対応付けられる粒度で表現することが必要である．例えば，図 3.1 は，左の実際の画像例に対して，注目したい要素のみをバウンディングと区間だけで単純化して抽象化したものを右に掲載した図であり，このような抽象度で記述することは，人間の運転タスクからも直観的に理解しやすく，複数の画像を定めた条件により判定しやすいことが期待できる．このように．この章で提案する仕様では，機能の判断に必要な画像上の特定領域をセマンティック情報と対応づけ，画像情報上でオブジェクトの位置がその特定の領域に対してどのような相対位置にあるかを記述し，その相対位置が自動運転上でどのような機能に対応づくかを定義する．よって，オブジェクトの位置以外の形状や色といった画像情報は捨て，バウン

ティングボックスで抽象化し，扱うセマンティック情報は，図 3.1 の travelingSectionSet や，deceleratingDistance のように画像内の領域で表現できる情報と，停止や停止しないなどの自動運転システムのレスポンスに該当する機能のみで，画像情報に含まれないオブジェクトの正確な距離や速度などの情報は考慮しない．以上の要件を満たす言語を定義するために，BBSL では，区間演算という既存の体系における区間を活用することで，バウンディングボックスを定義する．この体系における 2 次元の区間でバウンディングボックスを定義することで，オブジェクト同士が上下左右のどのように位置関係を有しているかの一部は区間の大小関係によって記述することができる．一方で，元来，区間演算は誤差付きの実数値を計算するために定義されているため，位置関係を記述するためには独自の演算子が必要になる可能性が高い．そこで，自動運転システムの画像認識の仕様として必要な位置関係を，実際の画像から特定をしながら，各データ型と各演算子を定義する．そして，自動運転システムで現状使用されている安全性フレームワークに則り，BBSL で記述した位置関係の仕様のみで，OEDR のレスポンスを全て書き分けることができるかを評価する．

図 3.1: BBSL で抽象化される画像のイメージ

3.2 データ型

まず，初めに自動運転システムの画像認識の仕様に必要なデータ型を説明する．表 3.1 は BBSL がサポートするデータ型のすべてを示す．BBSL は，real 型，bool 型，interval 型，bb 型，setBB 型を有す．

表 3.1: BBSL がサポートするデータ型

型指定文字	格納できるデータ	データ例
real	実数	$1, 0.3, 1/3$
bool	ブール値	$true, false$
interval	区間	$[1,3], [3,3], [100,500]$
bb	バウンディングボックス	$([1,3],[0,10]), ([3,3],[1,10])$
setBB	バウンディングボックスの集合	$\emptyset, \{([1,3],[0,10]), ([3,3],[1,10])\}$

自動運転システムの OEDR 仕様におけるオブジェクトの位置関係は，図 3.1 の deceleratingDistance のように特定の車間距離である長さの概念に対する物体の相対位置の記述が必要になる場合がある．そのため，こういった長さを記述するためのデータ型として，

interval 型をサポートする．BBSL において，interval 型は，区間演算で定義される区間と同様に定義 3.1 で与えられる．

定義 3.1. 区間 X とは，$\underline{X}$ と $\overline{X}$ を区間の端点として，以下で定義される．

$$X = [\underline{X}, \overline{X}] = \{x \in R : \underline{X} \leq x \leq \overline{X}\}$$

例 3.1. decelerationDistance が画像上の y 軸座標 150 と 200 の区間で定義されている場合，interval 型として $decelerationDistance = [150, 200]$ として表される．

このように，BBSL の Interval は，区間演算と同様に実数の組として定義するが，区間演算のように，コンピュータ上で近似して扱わざるを得ない無理数を表すものではない．

OEDR 仕様において，オブジェクトの位置関係を記述するためには，当然ながら図 3.1 上のオブジェクトの 1 つである Debrisstatic のような物体そのものを表現する必要があり，このようなオブジェクトは，画像認識システム上のデータ形式に則してバウンディングボックスとして表現する．そのため，このような物体を記述するためのデータ型として，bb 型をサポートする．bb 型は，区間演算で定義される多次元の Interval における $n = 2$ の時と同様で，定義 3.2 で与えられる．

定義 3.2. x 軸方向の区間 a_x,y 軸方向の区間 a_y とすると，バウンディングボックス a は以下で定義される．

$$a = (a_x, a_y)$$

例 3.2. 図 3.1 の obstacle は，x 軸座標が 400 から 410，y 軸座標が 130 から 180 のバウンディングボックスである．この時 BBSL ではこの obstacle を bb 型で $obstacle = ([400, 410], [130.180])$ として表すことができる．

OEDR の仕様において，他車が自車と同じレーンにいるのか，違うレーンにいるのかという車線に対する位置の条件は，自車の動作を決定する上で重要な情報である．しかし，前方カメラの画像において車線のような物体はバウンディングボックス単体で表現する場合に著しく，実際の形で乖離がうまれる．そのため，このようなオブジェクトを表現するためのデータ型として，バウンディングボックスの集合である setBB 型は定義 3.3 で与えられる．

定義 3.3. バウンディングボックスの集合 B は，バウンディングボックスの有限集合として以下に定義される．

$$B = \{a_i = (a_{x_i}, a_{y_i}) \mid a_{x_i} = [\underline{a_{x_i}}, \overline{a_{x_i}}], a_{y_i} = [\underline{a_{y_i}}, \overline{a_{y_i}}], i \in 1, 2, \ldots, n\}$$

ここで，

- a_{x_i} は i 番目のバウンディングボックスの x 軸方向の区間を表す．
- a_{y_i} は i 番目のバウンディングボックスの y 軸方向の区間を表す．
- $\underline{a_{x_i}}, \overline{a_{x_i}}, \underline{a_{y_i}}, \overline{a_{y_i}} \in R$ は区間の端点であり，$\underline{a_{x_i}} \leq \overline{a_{x_i}}$, $\underline{a_{y_i}} \leq \overline{a_{y_i}}$ を満たす．

例 3.3. 図 3.1 における自車の進行領域に該当する travelingSectionSet は手前の x 軸に合わせて単一のバウンディングボックスで与えると，画像の奥の方では他の車線を多く含んでします．このような，バウンディングボックス単体では実際の物体と乖離がうまれる複雑な形状の物体に対して，バウンディングの集合で被覆して表現できるように BBSL では setBB 型をサポートしている．

加えて，一般的な実数を表現するための real 型もサポートしている．これは，OEDR 仕様において，車線のような領域に対する他車の位置について，単純に方向でのみで与えれる位置関係ではなく，どの程度侵入しているかという割合で記述できるようにするための割合を指定する場合などに使用することができる．

以上のデータ型に加えて，位置関係の true,false を指定するための bool 型の 5 種類の型が OEDR 仕様を記述するために必要な型として BBSL でサポートされている．

3.3 演算子

次に，先ほど説明したデータ型に対して，自動運転システムの画像認識の仕様において必要となる演算子を特定し，定義する．表 3.2 に BBSL がサポートする演算子を示す．節 2.8 で前述した通り，区間演算は，誤差付き精度保証演算を目的とした体系であるため，オブジェクトの位置関係の記述に必要な多くの演算子は BBSL で新たに定義している．具体的には，演算子 (3) と (7) と (8) のみ区間演算で定義されているものと同様である．

表 3.2: BBSL がサポートする演算子

	演算子	使用例
(1)	not: bool → bool	$not(true) = false$
	and, or: bool × bool → bool	$true\ or\ false = true$
(2)	$<, >, =$: real × real → bool	$5 < 6 = true$
(3)	$<, >, =$: interval × interval → bool	$[2,5] < [6,8] = true$
(4)	$PROJ_i$: bb → interval , $i \in \{x, \underline{x}, \overline{x}, y, \underline{y}, \overline{y}\}$	$PROJ_{\overline{x}}(([3,5][2,8])) = [5,5]$
(5)	$\approx$: interval × interval → bool	$[3,8] \approx [5,10] = true$
(6)	$\approx$: bb × bb → bool	$([3,5],[2,8]) \approx ([1,4],[7,13]) = true$
(7)	$\subseteq, \supseteq$: interval × interval → bool	$[1,8] \subseteq [2,5] = true$
(8)	w: interval → real	$w([1,11]) = 10$
(9)	$\cap$: bb → bb	$([3,5],[2,8]) \cap ([1,4],[7,13]) = ([3,4],[7,8])$
(10)	$\cap, \cup$: setBB × setBB → setBB	図 3.4 参照
(11)	RAT: setBB × setBB → real	$RAT(([3,4],[2,3]),([1,2],[2,8])) = 1/6$

演算子 (1) は, ブール値で評価される関係同士を組み合わせるために導入されており，論理否定，論理積，および論理和である．また，演算子 (2) は，実数の大小関係と同値関係である．演算子 (3) は区間同士の位置関係を表現するための演算子であり，区間演算で定義される関係と同様であり，定義 3.4 で与えられるが，その使用方法は大きく異なる．

定義 3.4. $a = [\underline{a}, \overline{a}]$ と $b = [\underline{b}, \overline{b}]$ を区間とする．このとき，区間上の大小関係 $<$ と同値関係 $=$ は以下で定義される．

$$a < b \Leftrightarrow \overline{a} < \underline{b}$$
$$a = b \Leftrightarrow \underline{a} = \underline{b}\ and\ \overline{a} = \overline{b}$$

例 3.4. 図 3.2 は，上部に実際の前方カメラからの画像，下部にそこから前方車両との y 軸座標の位置関係に使用する要素のみ抽出して図示した図である．BBSL における区間同士の大小関係 $<$ は，この前方車両の y 座標断面 (y-coodinate section of lead vehicle) と停止距離 (stoppingDistance) のような位置関係を記述するために用いる演算子である．このとき，常に停止区間より前方車両が奥にあることを示す位置関係は，下方向を起点とし上方向に向かって値が大きくなる座標系の場合，$y-coodinate section of lead vehicle > stoppingDistance$ となる．

図 3.2: 前方車両との y 軸座標に関する位置関係の例

演算子 (3) は，1 次元の区間上の演算子だが，BBSL で記述する主な対象は 2 次元の区間で定義されるバウンディングボックスである．したがって，バウンディングボックスが x 軸と y 軸方向に沿った 2 つの区間で構成されることを前提として，演算子 (4) をバウンディングボックスを区間に写すプロジェクション関数として導入する．この定義を定義 3.5 に示す．

定義 3.5. $a = (a_x, a_y)$ をバウンディングボックスとし，$a_x = [\underline{a_x}, \overline{a_x}]$，$a_y = [\underline{a_y}, \overline{a_y}]$ を区間とする．バウンディングボックスから区間へのプロジェクション関数 $PROJ_i$ は以下で定義する．

$$PROJ_i(a) = a_i, PROJ_{\overline{i}}(a) = [\overline{a_i}, \overline{a_i}], PROJ_{\underline{i}}(a) = [\underline{a_i}, \underline{a_i}] \\ , i \in \{x, y\}$$

例 3.5. バウンディングボックス $a = ([350, 400], [200, 300])$ が与えられた時，$PROJ_x(a) = [350, 400]$，$PROJ_y(a) = [200, 300]$，$PROJ_{\overline{x}}(a)$，$= [400, 400] = 400$，そして，$PROJ_{\underline{x}}(a) = [350, 350] = 350$ となる．

この演算子によって，図 3.1 の obstacle(Debris static) のようにバウンディングボックスで表現されたオブジェクトの位置を deceleratingDistance との相対位置で記述するこ

とができる．例えば，obstacle が deceleratingDistance より奥に位置していることを記述するとき，y 座標は下方向を起点とし上方向に向かって値が大きくなる座標系の場合，$PROJ_y(obstacle) > deceleratingDistance$ と書くことができる．

以上の演算子によって，画像上のオブジェクト間の豊富な位置関係を記述ことができるが，その中でも特に頻繁に使用する位置関係の簡略表現として，演算子 (5)(6)(7) を導入する．演算子 (5) は 2 つの区間が重なっている関係を簡略するために使用するもので，この定義を定義 3.6 に示す．

定義 3.6. $a = [\underline{a}, \overline{a}]$ と $b = [\underline{b}, \overline{b}]$ を区間とする．区間の関係演算子 $\approx$ は以下で定義する．

$$a \approx b \Leftrightarrow (\underline{b} < \overline{a}\ and\ \underline{a} < \overline{b})\ or\ (\underline{a} < \overline{b}\ and\ \underline{b} < \overline{a})$$

例 3.6. 区間 $a = [150, 200]$，$b = [250, 300]$，$c = [190, 260]$ が与えられた時，関係 $a \approx c$ や $b \approx c$ は成り立つが， $a \approx b$ は成り立たない，

演算子 (6) は 2 つのバウンディングボックスが重なっている関係を簡略するために使用するもので，この定義を定義 3.7 に示す．

定義 3.7. $a = (a_x, a_y)$ と $b = (b_x, b_y)$ をバウンディングボックスとする．バウンディングボックスの関係演算子 $\approx$ は以下で定義する．

$$a \approx b \Leftrightarrow a_x \approx b_x\ and\ a_y \approx b_y$$

例 3.7. バウンディングボックス $a = ([350, 400], [200, 300])$，$b = ([390, 500], [100, 250])$，$c = ([360, 380], [100, 250])$ が与えられた時，関係 $a \approx b$ や $a \approx c$ は成り立つが， $b \approx c$ は成り立たない，

また，演算子 (7) は 2 つの区間の包含関係を簡略するために使用するもので，この定義は区間演算で用いられている区間上の包含関係の定義と同様であり，定義 3.8 で与えられる．

定義 3.8. $a = [\underline{a}, \overline{a}]$ と $b = [\underline{b}, \overline{b}]$ を区間とする．このとき，区間上の包含関係 $\subseteq$ は以下で定義される．

$$a \subseteq b \Leftrightarrow \underline{b} \leq \underline{a}\ and\ \overline{a} \leq \overline{b}$$

例 3.8. 区間 $a = [130, 200]$，$b = [150, 190]$，$c = [180, 300]$ が与えられた時，関係 $b \subseteq a$ は成り立つが， $a \subseteq b$ や $b \subseteq c$ は成り立たない．

BBSL は，以上のオブジェクト上の演算子とプロジェクション関数を用いることで，非常に多くの位置関係を区別して記述することができる．図 3.3 は，これまでの演算子を用いて BBSL で記述された 2 つのバウンディング a と b の位置関係の 9 種類の例を示しており，BBSL ではこれらの位置関係をすべて区別して書き分けることが可能である．例えば，上段左の位置関係の場合，演算子 (3)，演算子 (4)，演算子 (7) を使用して，b よりも a が上にあり，かつ，b の x 軸座標が a の x 軸座標に包含している関係を表現することができる．また，下段の左の位置関係の場合，演算子 (3)，演算子 (4)，演算子 (5) を使用して，a と b は y 軸座標において重複があり，x 軸座標に関しては，b は a より右側に位置している関係を表現することができる．

最後に，特定の計算のために BBSL でサポートされている他の演算子を定義する．演算子 (8) は，区間の幅を計算するために使用される演算子である．この定義は区間演算で用いられている w の定義と同様であり，定義 3.9 で与えられる．

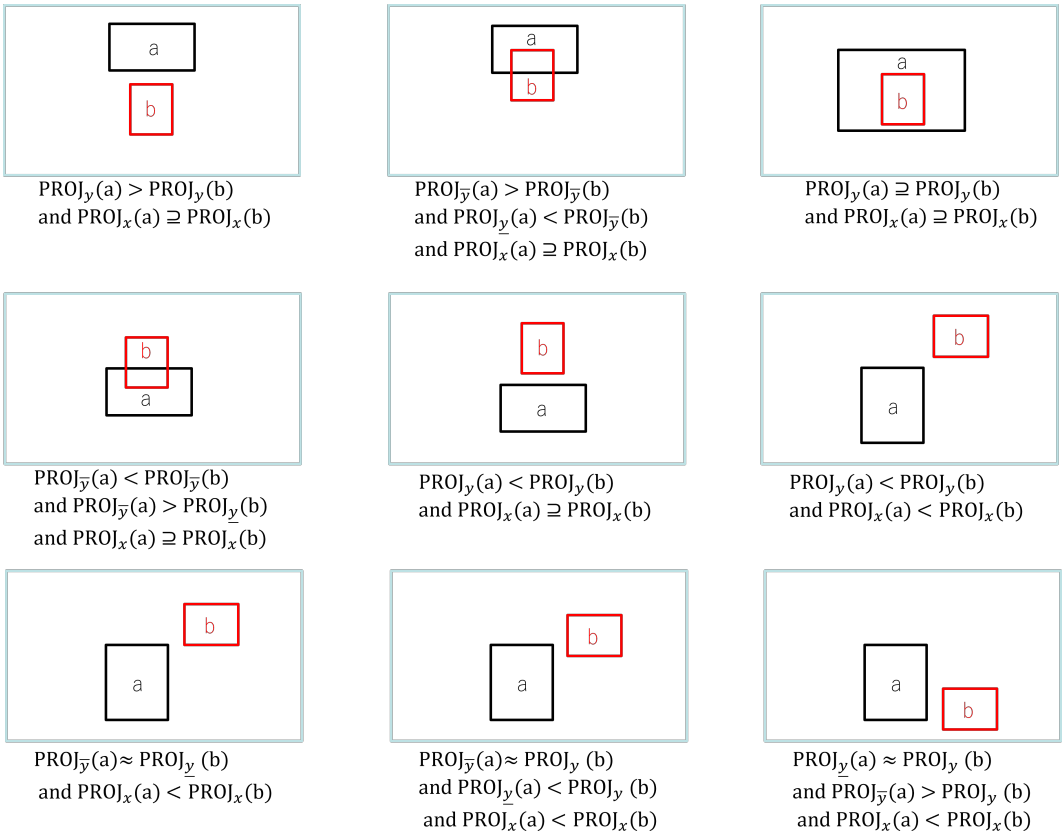

図 3.3: BBSL で書き分けることが可能なバウンディングボックス同士の位置関係の例

定義 3.9. $a = [\underline{a}, \overline{a}]$ を区間とすると，区間を受取り実数を返す関数 w は以下で定義される．

$$w(a) = \overline{a} - \underline{a}$$

例 3.9. 区間 $a = [130, 200]$，$b = [150, 190]$，$c = [180, 300]$ が与えられた時，$w(a) = 70$, $w(b) = 40$,$w(c) = 120$ となる．

さらに，演算子 (9) は，2 つのバウンディングボックス同士が重なっている場合に，重複部分のバウンディングボックスを計算する演算子である．バウンディングボックス上の $\cup$ に関しては，バウンディングボックス上で閉じた演算子ではないため，BBSL では導入していない．バウンディングボックス上の $\cap$ の定義は，定義 3.10 で与えられる．

定義 3.10. $a = (a_x, a_y)$ と $b = (b_x, b_y)$ をバウンディングボックスとし，$a_x = [\underline{a_x}, \overline{a_x}]$, $a_y = [\underline{a_y}, \overline{a_y}]$，$b_x = [\underline{b_x}, \overline{b_x}]$，$b_y = [\underline{b_y}, \overline{b_y}]$ を区間とする．$a_x \approx b_x \; and \; a_y \approx b_y$ である場合，バウンディングボックス上の演算子 $\cap$ は以下で定義する．

$$a \cap b = ([max\{\underline{a_x}, \underline{b_x}\}, min\{\overline{a_x}, \overline{b_x}\}], [max\{\underline{a_y}, \underline{b_y}\}, min\{\overline{a_y}, \overline{b_y}\}])$$

例 3.10. バウンディングボックス $a = ([350, 400], [200, 300])$，$b = ([390, 500], [100, 250])$ が与えられた時，$a \cap b = ([390, 400], [200, 250])$ となる．

演算子 (10) は，2 つのバウンディングボックスの集合に対する計算のために導入している．ここで，図 3.4 に演算子 (10) で計算した時のそれぞれのバウンディングボックスの集合を示す．バウンディングボックスの集合 A と B に対して，左の画像のグレーの領域が，$A \cup B$ で計算される領域を示しており，右の画像のグレーの領域が，$A \cap B$ で計算される領域を示している．このように，演算子 $\cup$ の場合は 2 つの集合の持つすべてのバウンディングボックスを保持するように計算し，演算子 $\cap$ の場合は 2 つの集合の持つバウンディングボックス同士の演算子 (9) の $\cap$ で得られるバウンディングボックスを保持するように計算する．バウンディングボックスの集合上の $\cap$ を定義 3.11 に，$\cup$ を定義 3.12 に示す．

定義 3.11. A, B をバウンディングボックスの集合とする．バウンディングボックスの集合上の演算子 $\cap$ は以下で定義する．

$$A \cap B = \{a \cap b | a \in A, b \in B\}$$

例 3.11. バウンディングボックス $a = ([300, 400], [100, 150])$，$b = ([300, 400], [130, 200])$, $c = ([350, 500], [120, 150])$ が与えられた時,
$\{a, b\} \cap \{c\} = \{([300, 400], [100, 150]), ([300, 400], [130, 200]), ([350, 500], [120, 150])\}$ となる．

定義 3.12. A, B をバウンディングボックスの集合とする．バウンディングボックスの集合上の演算子 $\cup$ は以下で定義する．

$$A \cup B = \{x | x \in A \vee x \in B\}$$

例 3.12. バウンディングボックス $a = ([300, 400], [100, 150])$，$b = ([300, 400], [130, 200])$, $c = ([350, 500], [120, 150])$ が与えられた時,
$\{a, b\} \cap \{c\} = \{([350, 400], [120, 150]), ([350, 400], [130, 150])\}$ となる．

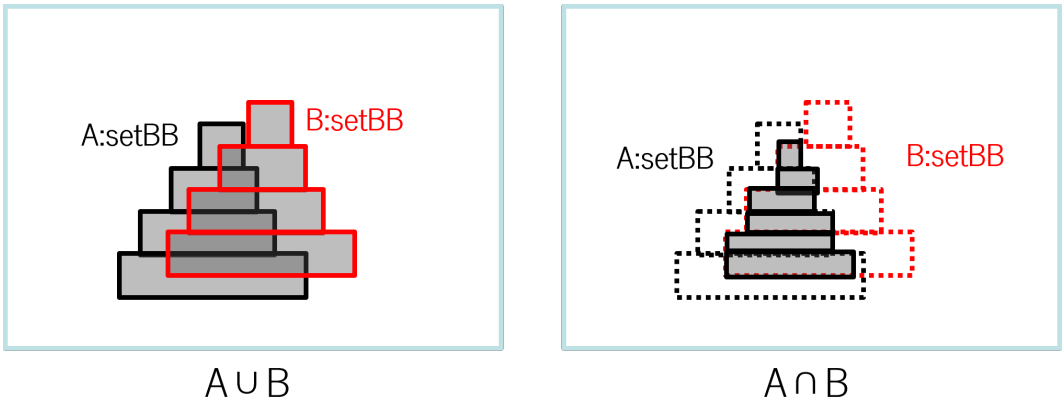

図 3.4: 演算子 (10) で計算されるバウンディングボックスの集合の例

演算子 (11) は 2 つのバウンディングボックスの集合が表す画像上の領域の面積比の計算のために導入している．この定義を定義 3.13 に示す．

定義 3.13. A, B をバウンディングボックスの集合とする．演算子 RAT は以下で定義する．

$$RAT(A, B) = (A \text{ が占める総面積})/(B \text{ が占める総面積})$$

例 3.13. バウンディングボックス $a = ([390, 400], [100, 120])$，$b = ([390, 400], [90, 110])$，$c = ([250, 260], [110, 120])$ が与えられた時，$RAT(\{c\}, \{a, b\}) = 10/300$ となる．RAT で計算されるバウンディングボックスの集合における面積とは，例えば，$\{a, b\}$ の場合，$(10 \times 20) + (10 \times 20) - (10 \times 10) = 300$ と計算される．

演算子 (10) 及び，(11) は自動運転システムの画像において，主に車線のような領域にどの程度車両が侵入しているかなどの条件を記述するのに使用することができる．例えば，図 3.5 では，ObstacleSet とは緑のバウンディングボックスで囲まれた 2 種類の obestacle(車と自転車) を表しており，travelingSectionSet は自車の進行方向の領域を表している．この場合，ObstacleSet を対象に，travelingSectionSet をどの程度妨げるているかの割合を $RAT((ObstacleSet \cap travelingSectionSet), travelingSectionSet)$ で計算している．この関数により，例えば $RAT((ObstacleSet \cap travelingSectionSet), travelingSectionSet) > 0.8$ とすることで，他車両が自車の進行方向の 80%を妨げている位置関係を記述することができる．このようにバウンディングボックスの集合とその上の演算子や関数を用いることで，BBSL では画像上の特定の領域をバウンディングボックスの集合で被覆して表現することもできる．

3.4 構文

ここでは，BBSL を使用して OEDR 仕様を記述するために不可欠な BBSL の構文について説明する．この構文により，BBSL はオブジェクトの位置関係によって自車のとりうる動作を定義することができる．まず，BBSL で記述した OEDR 仕様の一般形を図 3.6 に示す．図 3.6 の 1 行目から 7 行目に該当する「exfunction」と「endexfunction」で定義される範囲を外部関数ブロック，9 行目から 13 行目に該当する「precondition」と「endprecondition」で定義される範囲を前提条件ブロック，そして，15 行目から 20 行目や，22 行目から 24 行

図 3.5: 演算子 (10) 及び，(11) を使用する位置関係の例

目に該当する「case ケース名」と「endcase」で定義される範囲をケースブロックと呼ぶ．BBSL で記述される仕様はこのように 3 種類のブロックで構成される．図 3.7 は，具体的な仕様の例である．この仕様では，図 3.2 で示した位置関係を記述して，他車両が停止距離を表す stoppingDistance より奥にあれば停止する必要なく"NOT stop"，他車両の位置が停止距離と重なる場合は停止すること"stop"を定めている．

外部関数ブロックは，外部関数と呼ばれる関数を宣言する．BBSL で宣言する外部関数には以下の 3 種類の関数がある．

- 画像上の特定オブジェクトを取得するための関数．
- 位置関係だけでは表現できない物体の状態を判定するための関数．
- 画像上の特定領域をセマンティック情報として定義するための関数．

1 種類目の関数は，画像内に存在する特定の物体を取得するために使用される．例えば，車両や歩行者や自転車などの物体に対して，そのバウンディングボックスを返すことが典型的な用途である．BBSL においては，物体の位置や形状を取得するための必須要素として機能する．これにより，物体の位置を形式的に記述する際に，基礎となるオブジェクト情報を提供できる．2 種類目の関数は，BBSL で位置関係として定義することが難しい物体の特性や状態を判定するために利用される．例えば，車両が画像内に存在するかどうかの判定や，物体が特定の属性（歩行者や車両など）を持っているかどうかを判定するために使用する．この機能は BBSL において必須要素ではないが，後述する前提条件ブロック等で単純な位置関係だけではなく，物体自体の性質や状況に関する条件を取り扱わざるを得ない場合に利用される．3 種類目の関数は，画像上の特定領域をセマンティックな意味を持つ領域として明示的に定義するために使用される．これにより，画像とセマンティック情報を結び付ける役割を持ち，OEDR 仕様を画像上にマッピングするために必須となる．BBSL を使用して仕様を記述することは，暗に自動運転システムの機能を決定づける車両などのオブジェクトの画像上の位置を特定し，ここで表す画像上の領域との相対的な位置

で機能を定義することにある．このアプローチにより，画像上の位置情報を基に，安全要件や機能仕様を直接反映できる．例えば，進行方向や停止区間といった領域をこの外部関数として宣言することで，自動運転システムが判断すべきオブジェクトの位置情報を形式化できる．これらの外部関数は全て設計者が必要に応じて自由に宣言できる関数で，外部関数を宣言して使用することで，設計者はな OEDR 要件を考慮しつつ，位置関係によって決まる条件に焦点を絞って仕様を記述するための補完手段として設計されている．これらの関数は，一度宣言されると，その具体的な実装を考慮することなく，後続のブロックで参照することができる．例えば，図 3.7 の 1 行目から 5 行目では，3 つの外部関数を宣言している．1 つ目の外部関数 vehicleExists() は位置関係だけでは表現できない物体の状態を判定するための関数に該当し，ここでは，画像内に車両が存在するかどうかを判定する関数となっている．2 つ目の外部関数 vehicle() は画像上の特定オブジェクトを取得するための関数に該当し，ここでは，対象車両のバウンディングボックスを返す関数となっている．3 つ目の外部関数 stoppingDistance() は画像上の特定領域をセマンティック情報として定義するための関数に該当し，自動運転システムが停止するかどうかを位置関係で定義する上で必要となる適切な車間距離に該当する画像上の区間を返す関数である．

前提条件ブロックでは，仕様の前提条件を定義できる．このブロックで条件を記述することで，設計者は仕様が適用される範囲を定めることができる．ここでの条件式は，具体的に外部関数や BBSL の演算子を使用して，任意の条件式を記述できる．例えば，図 3.7 の 7 行目から 9 行目がこのブロックを示している．ここでは，この仕様は，画像内で車両が必ず検出された場合を前提に仕様が記述されていることが記されている．外部関数 vehicleExists を用いることで，車両が検出されない場合 false と評価され，前提条件を満たさない．つまり，このとき，この仕様の適用外であると判定され，記述された動作の範囲外とみなされる．ここで使用した外部関数 vehicleExists のように，特定のオブジェクトの存在判定やラベルに関する判定など，オブジェクトの空間情報以外の判定は便宜上，外部関数を用いて明示する．

ケースブロックでは，自動運転システムの各動作条件を，オブジェクトの位置関係を用いた条件で記述する場所である．仕様上で任意の数だけ記述できるのはこのケースブロックのみで，他のブロックはすべて仕様上で 1 つのみ記述する．このブロックは「case」の直後に，そのケースで記述する機能の名前を記述することで定義される．条件を定義するために使用されるすべての変数は，終端記号"let"の直後に変数名を宣言し，変数には，外部関数，リテラル，または何かしらの演算子を使用して値を割り当てる必要がある．そして,「in」の後に，宣言した変数を用いて条件式を記述する．この条件式に，節 3.3 で定義した演算子を使用して，バウンディングボックスの位置情報を記述することになる．例えば，図 3.7 の 11 行目から 15 行目と 17 行目から 21 行目がそれぞれこのケースブロックに該当する．最初のケースである stop は，外部関数 vehicle() から値が代入される変数 vehicle と外部関数 stoppingDistance() から値が代入される変数 stoppingDistance を使用して，自車が停止すべき条件を 14 行目に記述している．一方で，2 番目のケースである NOT stop では，同様の変数を宣言して，自車が停止する必要のない条件を 20 行目に記述している．

このような仕様を記述するための BBSL の構文を，拡張バッカス・ナウア記法 (EBNF) を使用して図 3.8 に示す．ここで，非終端記号は $\langle\rangle$ で，終端記号は”” で表している．節 3.2 で説明したように，BBSL では real 型，bool 型，interval 型，bb 型，setBB 型の 5 種類のデータ型をサポートしている．これらの型名で定義される非終端記号が $\langle type \rangle$ で定義されている．さらに，各型に対応するリテラルを表す非終端記号 ($\langle real\_lit \rangle$,$\langle bool\_lit \rangle$,$\langle interval\_lit \rangle$, $\langle bb\_lit \rangle$,$\langle setBB\_lit \rangle$) が定義され，各型に関連付けられた値を指定するために使用されている．節 3.3 で説明したように，BBSL では，値を表す非終端記号 $\langle value \rangle$ を作るために，様々な演算子をサポートしている．具体的には，特定の計算を処理する w や RAT, $PROJ$ などで定義されるプライマリ関数 $\langle prim\_func \rangle$ や，集合やバウンディングボック

スの二項関係である cap や cup で定義されるバイナリ演算子 $\langle binary\_op \rangle$ が含まれる．また，$\approx$，$<$，$>$ などで定義される関係演算子 $\langle binary\_rel \rangle$ と"forall"や"exists"を使用する量化表現 $\langle quant\_exp \rangle$，および，and や or などで定義される論理演算子 $\langle fml\_op \rangle$ を使用して，条件式を表す $\langle formula \rangle$ を定式化できる．これらの要素を使用して作成された仕様 $\langle spec \rangle$ は，前述したように $\langle exfunc\_block \rangle$，$\langle precond\_block \rangle$ および $\langle case\_block \rangle$ の 3 つのブロックで構成される．ブロック $\langle exfunc\_block \rangle$ 内では，$\langle name \rangle$ の規則に従って任意の数の外部関数 $\langle exdeclar \rangle$ を宣言できる．また，ブロック $\langle precond\_block \rangle$ 内で前提条件を記述し，ブロック $\langle case\_block \rangle$ 内では，条件を定義するために使用されるすべての変数は，終端記号"let"の直後に $\langle vardeclar \rangle$ に従って宣言される．変数には，外部関数，リテラル，または何かしらの演算子を使用して値 $\langle value \rangle$ を割り当てる必要があり，ケースや変数名は $\langle name \rangle$ の規則に従う．

以上の型，演算子，構文で構成される言語が BBSL である．BBSL がサポートする区間やバウンディングボックス (または，その集合) の型により，特定の数値に依存せずに抽象的な位置関係を記述することができる．また，BBSL がサポートする演算子は，多様な位置関係を記述することができる．そして，BBSL の構文では，外部関数により画像上の対象オブジェクトと，セマンティック情報と対応づいた画像上の特定領域を示し，それらの相対位置によって記述した条件を case 文のラベルとして自動運転システムの機能と対応付ける構造になっている．これらの提案により，自動運転システムのタスク OEDR のうち，画像上の要素のみから決まる OEDR の形式仕様を設計することができる．この構造化されたアプローチにより，BBSL は自動運転のコンテキストで画像認識タスクを考えることを助長し，厳密に書き下すことができる．

```
exfunction
    //外部関数の宣言
    get 画像上のオブジェクト():型
    get 画像上の特定領域():型
    get 位置情報以外の判定式():bool
    ....
endexfunction

precondition
    [
    条件式
    ]
endprecondition

case action1
    let ローカル変数１ : 型 = 値や関数,
        ローカル変数２ : 型 = 値や関数,
                              .... in
          条件式
endcase
...
case actionX
        ...
endcase
```

図 3.6: BBSL で記述した OEDR の仕様の一般形

```
1  exfunction
2        vehicleExists():bool
3        vehicle():bb
4        stoppingDistance():interval
5  endexfunction
6
7  precondition
8        [vehicleExists() = true]
9  endprecondition
10
11 case stop
12       let vehicle : bb = vehicle(),
13       stoppingDistance : interval = stoppingDistance()
                 in
14             PROJ_y(vehicle)≈ stoppingDistance
15 endcase
16
17 case NOT stop
18       let vehicle : bb = vehicle(),
19       stoppingDistance : interval = stoppingDistance()
                 in
20             not(PROJ_y(vehicle)≈ stoppingDistance)
21 endcase
```

図 3.7: 車間距離に基づく単純な仕様の例

<spec> ::= <exfunc_block> <precond_block> <case_block>+
<exfunc_block> ::= “exfunction” <exdeclar>* “endexfunction”
<precond_block> ::= “precondition” “[“ <formula> “]” “endprecondition”
<case_block> ::= “case” <name> “”let” <vardeclar> “in” <formula>
<exdeclar> ::= <name> “(” <type> (“,” <type>)* “)” <type>
<vardeclar> ::= <name> <type> “=“ <value> (“,” <name> <type> = <value>)*
<name> ::= ([A-Z] | [a-z] | [0-9] | “-” | “_” | “ ”)+
<type> ::= “:” (“real” | “bool” | “interval” | “bb” | “setBB”)
<formula> ::= <atomic_fml> | <neg_fml> | (< formula > <fml_op> < formula >)
| <quant_ exp> “.” “(” <formula> “)”
<atomic_fml> ::= <bool_lit> | (<value> <binary_rel> <value>) | <exfunc>
<binary_rel> ::= “<” | “>” | “=” | “≈ “ | “⊆” | “ ⊇”
<neg_fml> ::= “not” <formula>
<fml_op> ::= “and” | “or”
<quant_exp> := <quant> <name> “∈” <value> (“,” <name> “∈” <value>)*
<quant> ::= “forall” | “exists”
<value> ::= <literal> | <func_app> | <var> | (<value> <binary_op> <value>)
<literal> ::= <real_lit> | <bool_lit> | <interval_lit> | <bb_lit> | <setbb_lit>
<real_lit> ::= <pos> | < neg_pos >
<pos> ::= (“0” | [1-9] [0-9]*) (“.” [0-9]+)?
<neg_pos> ::= “-” <pos>
<bool_lit> ::= “true” | “false”
<interval_lit> ::= (“[” < real_lit > “,” < real_lit > “]”) | < real_lit >
<bb_lit> ::= “(“ <interval_lit> “,” <interval_lit> “)”
<setbb_lit> ::= “{“ <bb_lit> (“,” <bb_lit>)* “}”
<binary_op> :== “∩” | “∪”
<func_app> :== <func> “(“ <value> (“,” <value>)* “)”
<func> ::= <prim_func> | <exfunc>
<var> ::= <name>
<exfunc> :== <name>
<prim_func> ::= “w” | “RAT” | <proj>
<proj> ::= “PROJ” <proj_index>
<proj_index> ::= “x” | “$\overline{x}$” | “$\underline{x}$” | “y” | “$\overline{y}$” | “$\underline{y}$”

図 3.8: EBNF による BBSL の構文定義

3.5　仕様上の性質

BBSL で記述した OEDR の仕様は，節 3.4 で定義した構文規則内で実際どのように記述するかによって，仕様上の品質に大きくばらつきがある．例えば，構文規則上 $\langle case\_block \rangle$ はいくらでも追加することができるため，ひたすら冗長な仕様を記述することも可能である．そこで，本節では構文規則上記述可能な全ての BBSL による OEDR の仕様を対象に，3 種類の仕様上の性質として網羅性，排他性，非冗長性を定義し，これらの性質が仕様としてどのような品質を表現するかを説明する．

まず初めに，BBSL で記述した仕様全体のセマンティックを考える．BBSL で記述した仕様は，画像情報を抽象化し，その位置関係により，自動運転システムのレスポンスを定義している仕様である．そのことを図示したものを図 3.9 に示す．この図は，正面カメラからの画像 (Reality Image) のうち，ラベル付けされたバウンディングボックスの情報 (BBSL Abstract) から，BBSL で書かれた仕様 (Specification) を通して，その抽象化された画像情報から自車がどのような反応を取るべきか (Action) を判定することを表している．ここでは，まず，このような概念に定義を与える．

はじめに，図 3.9 の BBSL Abstract に該当するラベル付けされたバウンディングボックスを持つ画像の全体集合を定義 3.14 に示す．

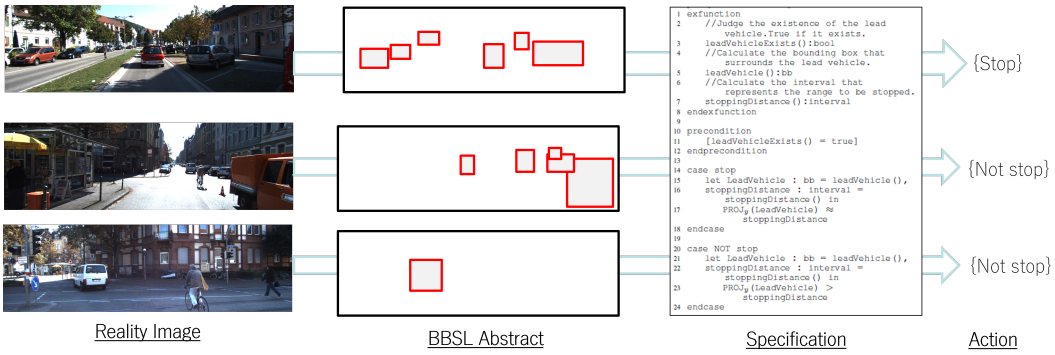

図 3.9: BBSL によって抽象化される画像と自動運転システムのレスポンスとの対応

定義 3.14. I を画像の全体集合，BB をバウンディングボックスの全体集合，L を車両や歩行者を含むラベルの全体集合とする．複数のラベル付きバウンディングボックスを持つ画像の全体集合 I_{BB} は以下で定義する．

$$I_{BB} = I \times 2^{BB \times L}$$

例 3.14. ここでの I は自動運転システムの前方カメラで入力される可能性のあるあらゆる画像を含む集合である．BB は，これらの画像上に表現可能なすべてのバウンディングボックスの集合を表し，L は車両，歩行者，その他のオブジェクトを含むすべてのラベルの集合を表す．I_{BB} の各要素は，画像とその画像内のラベル付きのバウンディングボックスの集合で構成され，自動運転システムの前方カメラから入力される任意の画像は I_{BB} の要素として扱う．

次に，図 3.9 の Specification に該当する BBSL で記述した仕様全体のセマンティックを定義 3.15 に定義する．BBSL で記述された仕様は，画像情報はバウンディングボックスの位置関係として抽象化され，自動運転システムの反応をケースとして定義する．つまり，図 3.9 に示すように，画像情報のうち位置情報を受け取り，自動運転システムの機能としてのレスポンスを返す関数として定義することができる．

定義 3.15. BBSL で記述された仕様は以下に示す C と f の列 $S = (C, f)$ で定義する.

- C は仕様に記述されたケースの集合とする.
- f は関数 $f : I_{BB^-} \to 2^C$ とする．この時，$I_{BB^-} = \{i_{BB} \in I_{BB} \mid i_{BB}$ は仕様に書かれた前提条件を満たす $\}$.

以降，この論文では，仕様 $S = (C, f)$ の f に対する I_{BB^-} を $dom(f)$ と表記する.

例 3.15. 図 3.7 に示す仕様を例に考える．この仕様では，自動運転システムが画像内の他車の位置に基づいて停止するかどうかが記述されている．このとき，C は，仕様に記述された自動運転システムのレスポンス，つまり，ケースの集合であり，$C = \{\text{stop}, \text{NOT stop}\}$ となる．また，前提条件には，画像内に必ず 1 つは他車両が存在することが記述されており，その前提条件を満たす画像を入力とする関数 f は，停止距離に対する他車の位置に応じて，各画像 $i_{BB} \in I_{BB^-}$ を 2^C にマッピングする．例えば，あるバウンディングボックスを持つ画像 i_{BB} において，停止距離に他車が含まれている場合は，$f(i_{BB}) = \{\text{stop}\}$ となり，他車が停止距離に含まれていない場合は，$f(i_{BB}) = \{\text{NOT stop}\}$ となる．この仕様における $dom(f)$ は，他車が存在するすべての画像の集合となる.

以上の定義を用いて，仕様上の性質として網羅性，排他性，非冗長性を考える.

まず，網羅的な仕様とは，任意の画像に対して，仕様上の前提条件を満たす場合 (I_{BB^-} の要素である場合)，必ず何かしらの自動運転システムのレスポンスが規定されている仕様を表す．これにより, 未知の画像や特殊な画像でも仕様上の前提条件を満たすならば，必ず機能が定義されている仕様であることを保証できる．つまり，網羅的な仕様は，いかなる状況においても動作が未定義にならないことを保証するものであり，IEEE 830 における完全性と部分的に一致している．このような網羅的な仕様を定義 3.16 に定義する.

定義 3.16. BBSL で記述された任意の仕様 $S = (C, f)$ に対して，網羅的な仕様であることは以下の数式を満たす.

$$\forall i \in dom(f).(f(i) \neq \emptyset)$$

例 3.16. 図 3.7 に示す仕様を例に考える．この仕様は $C = \{\text{stop}, \text{NOT stop}\}$，関数 f は，停止距離に対する他車の位置に応じて，各画像 $i_{BB} \in I_{BB^-}$ を 2^C にマッピングする関数である $S = (C, f)$ として定義できる．このとき，他車が検出された任意の画像 $i_{BB} \in dom(f)$ に対して，関数 $f(i_{BB})$ は stop または NOT stop のいずれかを含む応答を必ず返す．そのため，図 3.7 の仕様は網羅的な仕様である.

次に，排他的な仕様とは，ある特定の画像に対して，複数の異なるレスポンスが規定されていない仕様である．これにより，同一の状況で異なる 2 つ以上のレスポンスが発生する曖昧性が排除されていることを保証できる．つまり，排他的な仕様は，ある入力に対して複数の異なるレスポンスが定義されることを防ぎ，動作の曖昧さや矛盾を排除するものであり，IEEE 830 における無曖昧性および一貫性と部分的に一致している．このような排他的な仕様を定義 3.17 に定義する.

定義 3.17. BBSL で記述された任意の仕様 $S = (C, f)$ に対して，排他的な仕様であることは以下の数式を満たす.

$$\forall i \in dom(f), \exists c \in C.(f(i) = \{c\} or f(i) = \emptyset)$$

例 3.17. 図 3.7 に示す仕様を例に考える．この仕様を $C = \{\text{stop}, \text{NOT stop}\}$，関数 f は，停止距離に対する他車の位置に応じて，各画像 $i_{BB} \in I_{BB^-}$ を 2^C にマッピングする関数である $S = (C, f)$ として定義した時，他車が検出された任意の画像 $i_{BB} \in dom(f)$ に対して，関数 $f(i_{BB})$ は stop または NOT stop の両方に割り当てることはない．そのため，図 3.7 の仕様は排他的な仕様である．

最後に，非冗長な仕様とは，仕様上に定義されているすべてのケースが，少なくとも 1 つの画像に対して適用される仕様を指す．つまり，非冗長な仕様は，仕様内に存在するすべての条件が，少なくとも 1 つの入力に対して適用可能であることを保証し，意味のない記述や未使用の定義（いわば仕様上のデッドコード）を排除するものであり，IEEE 830 における正当性と部分的に一致している．また，このような仕様は変更や保守の観点からも扱いやすく，修正容易性にもつながる．これにより，無駄なケースや使われない機能が存在しないことが保証できる．このような非冗長な仕様を定義 3.18 に定義する．

定義 3.18. BBSL で記述された任意の仕様 $S = (C, f)$ に対して，非冗長な仕様であることは以下の数式を満たす．

$$\forall c \in C, \exists i \in dom(f).(c \in f(i))$$

例 3.18. 図 3.7 に示す仕様を例に考える．この仕様を $C = \{\text{stop}, \text{NOT stop}\}$，関数 f は，停止距離に対する他車の位置に応じて，各画像 $i_{BB} \in I_{BB^-}$ を 2^C にマッピングする関数である $S = (C, f)$ として定義した時，stop，NOT stop の条件を満たす $dom(f)$ の要素が少なくとも 1 つは存在する．そのため，図 3.7 の仕様は非冗長な仕様である．

以上の 3 つの性質は，BBSL で記述した OEDR の仕様が，形式仕様として品質を担保するための必須要件である．これらを満たすことで，仕様で定義した自動運転システムのレスポンスがテスト可能であり，あらゆる画像に対して仕様外なのか仕様内であればどのレスポンスに該当するかが定義されていることが保証される．本論文では以降，特に明記しない限り，BBSL で記述された仕様はすべて網羅的，排反的，非冗長な仕様である．

第4章　機能テスト

4.1　概要

ここでは，前章で定義した OEDR の仕様を機能仕様として用い，画像認識処理を対象とした機能テストを提案する．BBSL で記述した機能仕様は，画像情報の中でもオブジェクトの位置に対して，対応する機能が記述されている．節 1.1.2 で説明したように，一般のデータセットにおけるグラウンドトゥルースデータを期待値として画像認識の出力とつきあわせてテストをする場合は，自動運転システム全体の機能への影響が考慮できない問題点があった．そこで，提案する機能テストでは，画像認識処理が機能仕様の通りに動作していることを確認することを目的としている．提案テストにおける期待値は，データセットにおけるグラウンドトゥルースデータではなく，グラウンドトゥルースデータはあくまで正しい位置情報として参照し，機能仕様に定義した正しい機能 (stop,NOT stop) を期待値とする．ここで提案するテストの判定方法の俯瞰図を図 4.1 に示す．前方カメラからの画像が与えられたときに，グラウンドトゥルースデータを参照して，その画像に対する機能を期待値として定義し，SUT(画像認識) が検出したオブジェクトのバウンディングボックスが該当する機能仕様上の機能を出力として期待値とつきあわせる．テストケースは，画像と期待値の組で定義する．本論文では，テストデータは既存のオープンデータセットを用いるため，テストケースはテストデータと BBSL で記述した機能仕様があれば作成することができる．このようなアプローチにより，画像認識処理が機能仕様に従っているか違反しているかを判定する．

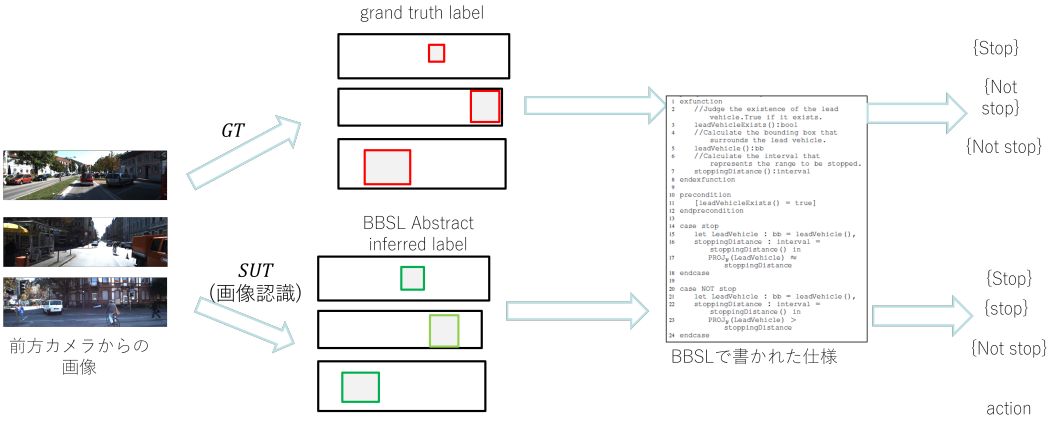

図 4.1: BBSL で書かれた仕様を用いた判定方法の俯瞰図

以上のテストケース，及び，判定方法により，ある 1 画像と期待値で定義されるテストケースについて，機能仕様に従って画像認識処理をおこないているか否かの判定を行うが，

自動運転システムを取り巻く膨大な環境を考えたとき，可能性のある全ての画像についてテストケースを準備することは現実的でない．そこで，ここでは，提案した機能テストにおけるデータセットの充分さを定量的に確認するための指標も提供する．本研究では，考え方の異なる 2 種類のカバレッジを提案する．1 種類目は，BBSL で記述された仕様に対してどの程度テストされたかを示すカバレッジである．BBSL で記述した仕様は複数の条件式によって自動運転システムの反応が定義されているため，あらゆる条件が一度は判定に使われてることをテストの充分性に用いる．ただし，BBSL で記述した仕様内には様々な粒度の条件が存在している．そこで，本研究では，従来のソフトウェアのテストにおいて，プログラムの構造によって定義されるコードカバレッジの考え方を応用する [37]．本研究で提案する BBSL 仕様ベースのカバレッジは，BBSL 仕様ベース判断カバレッジ，BBSL 仕様ベース条件カバレッジ，BBSL 仕様ベース条件判断カバレッジ，BBSL 仕様ベース改良条件判断カバレッジ，BBSL 仕様ベース複合条件カバレッジの 5 つである．特に，BBSL 仕様ベース条件判断カバレッジ，BBSL 仕様ベース改良条件判断カバレッジ，BBSL 仕様ベース複合条件カバレッジの 3 つに関しては後者ほど粒度が細かいカバレッジである．これらのカバレッジのどれが適切であるかは，コードカバレッジにおけるソースコードと同様に，仕様によって異なることが想定される．そのため，これらのカバレッジを複数の仕様において測定し，その効果に関して調査を行う．また，一般にコードカバレッジにおいて，改良条件判断カバレッジは，全ての条件の組み合わせを考慮する複合条件カバレッジ中から重要で優先度が高い組み合わせに絞って定義するカバレッジであり，何に着目し優先度をつけるかは，分野により異なる [38][39][40]．BBSL で記述された仕様では，認識モジュールのコンテキストより，判断の境界に近い組み合わせほど，自車の誤動作を誘発しやすいという意味で，優先度の高い重要な条件の組み合わせと考えることができる．例えば，ある 2 種類の位置関係 A，B によって，stop の条件は「A and B」と定義されており，それ以外のとき Not stop と定義されているとする．このとき，(A,B) が (true,true) の場合は，どちらかが変われば，stop 条件を満たさなくなる．対して，(false,false) の場合は，どちらかが変わっても，Not stop の条件のままである．この場合，(true,true) は，(false,false) よりも誤動作を誘発しやすく，優先度の高い重要な条件の組み合わせと考えれる．このように，自動運転システムの別の反応 (stop に対する Not stop) を誘発しやすい条件の組み合わせをセンシティブな条件の組み合わせとして定義し，その組み合わせを優先度が高いものとして BBSL 仕様ベース改良条件判断カバレッジという名前で定義し，提供する．これらのカバレッジによって，テストケースが仕様のどの部分を網羅しているかを定量的に評価し，機能テストの十分性を担保する．

ただし，これらのカバレッジは具体的な仕様の分量や条件に依存するため，機能テストをどの時点で終了すべきかを明確に定めることは困難である．そこで，2 種類目のカバレッジでは，画像上のオブジェクトに関する多様性を仕様とは独立して測定し，テスト終了基準の 1 つとして活用できる形式で定義することを試みる．本論文では，オブジェクトに関する多様性として，オブジェクトが画像内で配置される位置の分布や，スケールの分布を考慮してカバレッジを定義し，そのカバレッジが一定の飽和点に達した場合を暫定的な終了基準として採用する．しかし，すべての理論的な画像上の位置やサイズの範囲に対して，自動運転システムの画像認識における現実的な位置やサイズは物理的な制約を伴い限定的であることが考えられる．そこで，既存のデータセット上において，オブジェクトの位置やサイズの多様性を測定し，その結果から画像上の空間に対するオブジェクトの多様さを示すカバレッジとして，空間位置カバレッジと空間サイズカバレッジの 2 つを提案する．これらのカバレッジでは，仕様に依存せずにテストケースの多様性を評価することに重きを置く．愚直にテストケースを増やすことよりも少し効果的に，この指標により飽和点を計測し，暫定的な終了基準を提案する．

4.2　テスト判定

機能テストにおけるテスト判定を定義するための準備として，はじめに，テストデータ(生画像)，グラウンドトゥルースデータ，テスト対象のシステムを定義する．ここでの，テスト対象のシステムは，図 4.2 に示すように，画像を入力として，推論結果のラベル付きバウンディングボックスを持つ画像を出力する自動運転システムの画像認識システムである．これらの定義を，定義 4.1 に示す．

図 4.2: 本研究におけるテスト対象システムである画像認識

定義 4.1. テストデータ Td は，画像の全体集合 I のサブセット $Td \subset I$. また，テスト対象システム SUT は．関数 $SUT : I \to I_{BB}$ で与えられる画像を入力として推論結果のラベル付きバウンディングボックスを持つ画像を出力する自動運転システムの画像認識システム．さらに，グラウンドトゥルースデータは関数 $GT : I \to I_{BB}$ によって与えられるとする．

例 4.1. このテストにおけるテスト対象システム SUT は，図 4.2 に示したように，画像を入力として受け取り，推論されたラベルを持つ画像を出力として生成する画像認識システムである．テストに使用される画像のセットは Td で示し，Td に対応するラベル情報，つまり，画像認識システムにおけるグラウンドトゥルースラベルは関数 $G(i \in Td)$ によって示す．

図 4.3 は機能テストプロセスの全体像であるが，これまでの定義により，GT，SUT，そして仕様 S が定義された．これにより，任意の画像に対して，GT と SUT は各々ラベル付きのバウンディングボックスを返し，仕様 S を通して,「pseudo-oracle response」と,「SUT response」に該当する各々の反応を示すことができる．残った定義として，まず機能テストのテストケースを定義 4.2 に定義する．

定義 4.2. $S = (C, f)$ を網羅的，排反的，非冗長な仕様とする．Td，GT が与えられる時，テストケース $CASE$ は以下で定義する．

$$CASE = \{(td, f(GT(td))) \mid td \in Td)\}$$

つまり，この定義における $f(GT(td))$ は仕様ベースの疑似オラクルとして機能している，

例 4.2. 図 4.3 を例に考えると，$CASE = \{(td1, \{\text{Stop}\}),(td2, \{\text{Not stop}\})$ $,(td3, \{\text{Not stop}\})\}$ となり，3 種類のテストケースを持っていることになる．

最後に，機能テストの判定方法を定義 4.3 に定義する．この定義により，T であればこのテストケースは仕様に違反していないと判定され，F であれば仕様に違反していると判定される．

定義 4.3. $S = (C, f)$ を網羅的，排反的，非冗長な仕様とする．テスト対象システム SUT とテストケース $CASE$ が与えられる時，任意のケース $(td, c) \in CASE$ について，テストの判定条件 $P : CASE \to \{T, F\}$ は以下で定義する．

$$P(td, c) = \begin{cases} T & f(SUT(td)) = c \\ F & otherwise \end{cases}$$

例 4.3. 図 4.3 を例に考えると，前述したように，$CASE = \{(td1, \{\text{Stop}\}), (td2, \{\text{Not stop}\}), (td3, \{\text{Not stop}\})\}$ となる．このとき，テストの判定条件は $P(td1, \{\text{Stop}\}) = T$，$P(td2, \{\text{Not stop}\}) = F$，および $P(td3, \{\text{Not stop}\}) = T$ となる．

以上により，BBSL で記述された機能テストが定義された．これまで定義した記号を用いて機能テストプロセスの全体像を示した図 4.3 を用いてまとめると，テスト対象の物体検出システム SUT は，画像 $td \in Td$ を入力として，ラベル付きのバウンディングボックス $SUT(td)$ を出力する．BBSL で記述された仕様は，各オブジェクトのバウンディングボックスの位置関係を元に自動運転システムの応答と対応付ける．従って，SUT によって検出されたバウンディングに対する自動運転システムの応答 $f(SUT(td))$ とグラウンドトゥルースデータに対する自動運転システムの応答 $f(GT(td))$ を仕様から得ることができ，比較することで判定できる．このように，BBSL で記述された機能テストは，自動運転システムの OEDR の仕様を考慮して画像認識をテストすることができるようになる．例えば，図 4.4 の一番左に示す画像データを td とするテストケース (td,Stop) の場合を考える．テスト対象の SUT はこの画像 td を受け取ったとき，出力として図 4.4 の中央上の画像の緑色のバウンディングボックスを車両クラスとして出力する．テストの実施のためには，テストデータ (この場合は画像 td のみ) に対応する $GT(td)$ を得るために，既存のオープンデータセットを使用して事前に対応付けをする必要がある．今回の場合は，図 4.4 の中央下の画像の赤のバウンディングボックスを $GT(td)$ としている．ここでの仕様 $S = (C, f)$ は，stoppingDistance という区間との重なりで"stop"と"NOT stop"の条件を定義した図 3.7 に示した仕様とする．この仕様を実行するための $S = (C, f)$ における関数 f を実装する．この仕様の場合は stoppingDistance に関してのみ，与え方をこの実装時に決める必要があるため，図 3.7 の右上や，右下の画像でグレーで記した区間が stoppingDistance となるように決定し，残りは機械的に実装ができる．本研究では，これらの仕様のための実行コードはすべて Python を使用して開発した．以上の準備と実装により，f(SUT(td))=Not stop と f(GT(td))=Stop を計算することができ，その 2 つの値の比較により，この判定は P(td,Stop)=T と判定されることになる．このように，従来的には単純なバウンディングボックスの比較のみで評価されていた画像認識に対して，仕様を介して対応する"Not stop"や"stop"といった自動運転システムの全体としてのコンテキストで判定することがこの手法の特徴である．これらの仕様とサンプルコードの例は，節 5.3 で改めてとりあげる．

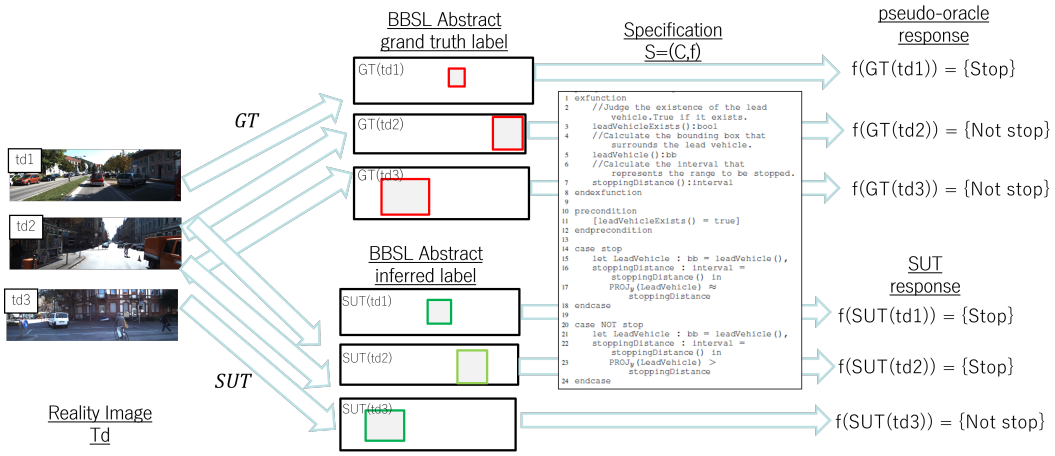

図 4.3: 機能テストプロセスの全体像

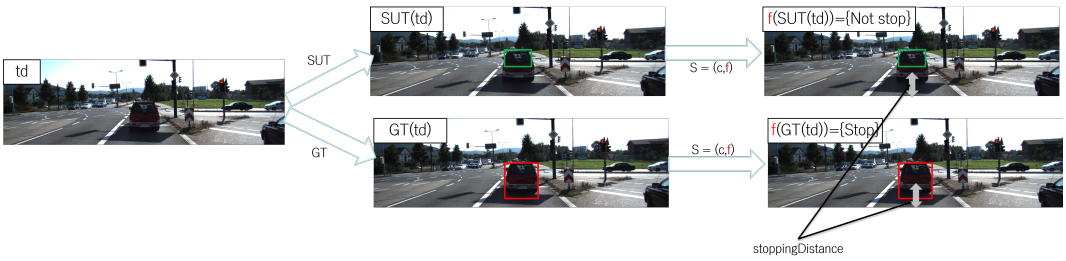

図 4.4: テストケース (td,Stop) おける判定の例

4.3 BBSL 仕様ベースカバレッジ

BBSL 仕様ベースカバレッジは，テストデータのグラウンドトゥルースが，BBSL で記述された仕様内の条件やケースをどの程度カバーしているかを示すカバレッジである．BBSL で記述した仕様内には様々な粒度の条件が存在しているため，仕様の構文構造に照らしながら，各要素ごとにカバレッジを提案する．この節で定義する全ての BBSL 仕様ベースカバレッジは網羅的かつ，排反的で非冗長な仕様に対して定義されている．

まず，仕様における $\langle case\_block \rangle$ がどの程度カバーされているかを示す BBSL 仕様ベース判断カバレッジ BC_d を定義 4.4 に定義する．3.4 節にて説明したように，BBSL で記述された仕様には，条件を含む 1 つ以上の $\langle case\_block \rangle$ がある．特に，網羅的かつ，排反的で非冗長な仕様においては，グラウンドトゥルースデータに対して常に 1 つのケースのみに対応する，

定義 4.4. グラウンドトゥルースデータを返す関数 GT が与えられた時，仕様 $S = (C, f)$，テストデータ Td に対する BBSL 仕様ベース判断カバレッジ BC_d は以下で定義する．

$$BC_d(Td, S) = \frac{|\{c | td \in Td, c \in C, f(GT(td)) = \{c\}\}|}{|C|}$$

例 4.4. 図 3.7 の仕様を例に考えると，$C = \{stop, Notstop\}$ であるため，BBSL 仕様ベース判断カバレッジ BC_d の分母 $|C| = 2$ であり，$f(GT(td))$ が stop の場合も Not stop の場合もテストされた時，BC_d は 100%となる．

仕様上の各ケース内でブール値として評価される論理式のうち，"and"や"or"や"not"を含まない単位をリテラルと呼ぶ．例えば，論理式"$(a > b)$ and C"は，"$a > b$"と"C"という 2 つのリテラルで構成されている．仕様 S 上のそのようなリテラルの列を L_s と表記する．また，$l_s \in L_s$ がテストデータ $td \in Td$ を使用して真と評価される場合，$l_s(td) = true$ と表記する．これらの表記法を用いて，仕様内のリテラルも焦点を当てて，リテラルの評価された値に基づいて BBSL 仕様ベース条件カバレッジ BC_c を定義 4.5 に定義する．

定義 4.5. 仕様 S におけるリテラルの列 $L_s = (l_1, l_2, ..., l_n)$ とテストデータ Td に対する BBSL 仕様ベース条件カバレッジ BC_c は以下に定義する．

$$BC_c(Td, L_s) = \frac{|\{(i, l_i(td)) | td \in Td, l_i \in L_s\}|}{|L_s| \times 2}$$

ここで特筆すべき特徴として，仕様上に同一のリテラルが複数出現していても，BBSL 仕様ベース条件カバレッジ BC_c では全て異なるリテラルとして分母を定義していることである．これは，仕様上で複数出現している条件がテストされることは，一度しか書かれていない条件をテストすることよりも重要であるという考えを反映している．また，BC_c が 100%になるテストデータであっても，BC_d も 100%になるとは限らないことにも注意が必要である．

例 4.5. 図 4.5 に示す仕様を例に考える．この仕様は，図 3.7 の仕様に，他車両が自車の進行方向に位置しているか x 軸方向の条件を付け加えた OEDR の仕様となる．図 4.5 に示す仕様においてリテラルは，16 行目，17 行目，23 行目，24 行目の 4 つのリテラルが含まれる．16 行目と 23 行目，または，17 行目と 24 行目は"not"を除いたリテラルとしては全く同じだが，定義上異なるリテラルとして数える．従って，この仕様においてのリテラルの列は $L_s =$ ("$PROJ_x(vehicle) \approx directionAreaDistance$","$PROJ_y(vehicle) \approx stoppingDistance$","$PROJ_x(vehicle) \approx directionAreaDistance$","$PROJ_y(vehicle) \approx$

$stoppingDistance$") となり，BC_c の分母は $4 \times 2 = 8$ となる．この時，"$PROJ_x(vehicle) \approx directionAreaDistance$"が true かつ"$PROJ_y(vehicle) \approx stoppingDistance$"が false となるテストケースと"$PROJ_x(vehicle) \approx directionAreaDistance$ が false かつ"$PROJ_y(vehicle) \approx stoppingDistance$"が true となるテストケースの 2 つで，$BC_c$ は 100%となるが，どちらのテストケースも該当する仕様上のケースは"NOT stop"であるため，BC_d は 50%になる．

次に，BBSL 仕様ベース判断カバレッジ BC_d と BBSL 仕様ベース条件カバレッジ BC_c の組み合わせである BBSL 仕様ベース条件判断カバレッジ BC_{cd} を定義 4.6 に定義する．

定義 4.6. グラウンドトゥルースデータを返す関数 GT が与えられた時，仕様 $S = (C, f)$ におけるリテラルの列 $L_s = (l_1, l_2, ..., l_n)$ とテストデータ Td に対する BBSL 仕様ベース条件判断カバレッジ BC_{cd} は以下に定義する．

$$BC_{cd}(Td, S, L_s) = \frac{|\{c|td \in Td, c \in C, f(GT(td)) = \{c\}\}|}{|C| + |L_s| \times 2} + \frac{|\{(i, l_i(td))|td \in Td, l_i \in L_s\}|}{|C| + |L_s| \times 2}.$$

例 4.6. 例 4.5 と同様に，図 4.5 に示す仕様を例に考えると，16 行目のリテラルが true かつ，17 行目のリテラルが false となるテストケースと，16 行目のリテラルが false かつ，17 行目のリテラルが true となるテストケースからなる 2 つのテストケースでテストを行った場合，BBSL 仕様ベース条件判断カバレッジは，BC_{cd} は $(1 + 8)/(2 + 8) = 90\%$ となる．

```
1  exfunction
2      vehicleExists():bool
3      vehicle():bb
4      directionAreaDistance():interval
5      stoppingDistance():interval
6  endexfunction
7
8  precondition
9      [vehicleExists() = true]
10 endprecondition
11
12 case stop
13     let vehicle : bb = vehicle(),
14     directionAreaDistance : interval =
           directionAreaDistance(),
15     stoppingDistance : interval = stoppingDistance()
           in
16         PROJ_x(vehicle) ≈ directionAreaDistance
17         and PROJ_y(vehicle) ≈ stoppingDistance
18 endcase
19
20 case NOT stop
21     let vehicle : bb = vehicle(),
22     directionAreaDistance : interval =
           directionAreaDistance() in
23         not(PROJ_x(vehicle) ≈ directionAreaDistance
               ) or
24         not (PROJ_y(vehicle) ≈ stoppingDistance)
25 endcase
```

図 4.5: 図 3.7 の仕様に x 軸方向の条件を追加した仕様

次に，BBSL で記述された仕様においてより優先度の高い条件の組み合わせをセンシティブな条件の組み合わせとして定義し，BBSL 仕様ベース改良条件判断カバレッジという名前でカバレッジを定義する．センシティブな条件の組み合わせを定義 4.7 に定義する．BBSL 仕様ベース条件カバレッジ BC_c では，ケース内の論理式の最小単位であるリテラルに対して定義を行ったが，ここではケース内で"and,or,not"が付与された最大単位の論理式を扱う．

定義 4.7. BBSL で記述された仕様 S に対して，各ケース内の条件を表す最大単位の論理式を"$p = l_1$ and/or l_2 and/or ... and/or l_n"とする．ここで，論理式 p は n 個のブール列 $(x_1, x_2, ..., x_n)$ を受け取り，それらを $l_1 ... l_n$ に割り当て，全体の真偽値を返す関数とみなすことができる．n 個のブール列 v が，$\exists i \in \{1, ..., n\}.p(v) \neq p(v^i)$ を満たすとき，v を p のセンシティブな条件の組み合わせと定義する．ここで，v^i とはブール列 v の i 番目のブール値のみ反転したブール列を表す．

例 4.7. 図 4.5 の仕様においては，"stop"ケース内の論理式 p_1 と"NOT stop"ケース内の論理式 p_2 がある．定義 4.7 の表記に合わせた場合，"$p_1 = l_1$ and l_2"と"p_2 =not l_3 or not l_4"になる．この時，p_1 と p_2 のセンシティブな条件の組み合わせを表 4.1 に示す．この表は，各リテラルに割り当てる真偽の全組み合わせを書き起こした上で，その組み合わせがセンシティブであるとき，チェックマークを記述している．

表 4.1: "$p_1 = l_1$ and l_2"と "p_2 =not l_3 or not l_4"に対するセンシティブな条件の組み合わせ

p_1 l_1 and l_2	l_1	l_2	Sensitive
T	T	T	✓
F	T	F	✓
F	F	T	✓
F	F	F	

p_2 not l_3 or not l_4	not l_3	not l_4	Sensitive
T	T	T	
F	T	F	✓
F	F	T	✓
F	F	F	✓

次に，任意の $td \in Td$ と $p \in Ps$ を受け取り，td を p の入力の型にプロジェクションする関数 M_p を定義 4.8 に定義する．

定義 4.8. 論理式 $p = l_1$ and/or l_2 and/or ... and/or l_n，テストデータ Td が与えられた時，任意の $td \in Td$ に対する関数 M_p を以下に定義する．

$$M_p(td) = (l_1(td), l_2(td), ..., l_n(td))$$

例 4.8. a と b がバウンティングボックスである論理式 $p = (PROJ_x(a) \approx PROJ_x(b))$ および $(PROJ_y(a) \leq PROJ_y(b))$ を例に考える．バウンティングボックス a の値が $([160, 180], [180, 200])$ で，b の値が $([170, 190], [190, 210])$ であるテストデータ td が与えられた時，関数 $M_p(td)$ は以下のように td を p の入力にプロジェクションする．

$$M_p(td) = (T, F).$$

つまり，論理式 p では，テストデータ td に対して，最初の条件は true と評価され，2 番目の条件は false と評価される．

以上の表記法と定義に基づき，BBSL 仕様ベース改良条件判断カバレッジ BC_{mcd} を定義 4.9 に定義する．

定義 4.9. 仕様 S 内の各ケースの最大単位の論理式の列を $P_s = (p_1, ..., p_n)$，論理式 $p \in P_s$ に対するセンシティブな条件の組み合わせのセットを V_p，テストデータを T_d，関数 M_p が与えられた時，$V_s = (V_{p_1}, ..., V_{p_n})$ とすると，BBSL 仕様ベース改良条件判断カバレッジ BC_{mcd} は以下で定義される．

$$BC_{mcd}(Td, P_s, V_s) = \frac{\sum_{p \in P_s} |\{M_p(td) | td \in Td, M_p(td) \in V_p\}|}{\sum_{V_p \in V_s} |V_p|}$$

例 4.9. 図 4.5 の仕様の場合，V_{p_1}={(T,T),(T,F),(F,T)}，V_{p_2}={(T,F),(F,T),(F,F)}となる．この場合，BC_{mcd} の分母は，$|V_{p_1}| + |V_{p_2}| = 3 + 3 = 6$．これまでに提案した BBSL 仕様ベースカバレッジとは異なり，BC_{mcd} の分母は仕様内の論理式やリテラルの数のみでは決定できず，"and","or","not"がどのように各リテラルを結合しているかに依存する．例えば，表 4.2 と表 4.3 には，それぞれ"$p_1 = l_1$ and l_2 and l_3"と"$p_2 = l_4$ or (l_5 and l_6)"におけるセンシティブな条件の組み合わせが示されている．どちらの論理式もリテラルの数は 3 つだが，$|V_{p_1}| = 4$ と $|V_{p_2}| = 7$ になる．

表 4.2: "$p_1 = l_1$ and l_2 and l_3"に対するセンシティブな条件の組み合わせ

p_1 l_1 and l_2 and l_3	l_1	l_2	l_3	Sensitive
T	T	T	T	✓
F	T	T	F	✓
F	T	F	T	✓
F	T	F	F	
F	F	T	T	✓
F	F	T	F	
F	F	F	T	
F	F	F	F	

最後に，仕様内に存在する全ての原子的な条件の組み合わせをとるカバレッジとして，BBSL 仕様ベース複合条件カバレッジを提案する．これまでの BBSL 仕様ベースカバレッジでは，論理式とリテラルは構文として扱っていたため，仕様上で同じ記述が複数繰り返されていた場合でも，別々で数え上げていた．しかし，BBSL 仕様ベース複合条件カバレッジでは，各条件の論理的な意味を考慮し，同じ意味の条件が複数あった場合は除いて 1 つに数えあげる．まず，仕様 S 内のうち同値なリテラルを除いた列に対して，論理的に割り当てることが可能なブール列の集合 B_s を定義 4.10 に定義する．

定義 4.10. 仕様 S 内のリテラルから同値なものを除いた列を $Q_s = (l_1, l_2, ..., l_n)$ とする．ここで，$I_{BB^-} = \{i_{BB} \in I_{BB} \mid i_{BB}$ は仕様に書かれた前提条件を満たす $\}$ とすると，Q_s に論理的に割り当てることが可能なブール列の集合 B_s は次のように定義する．

$$B_s = \{v | Q_s(d) = v, d \in I_{BB^-}\}$$

表 4.3: "$p_2 = l_4$ or (l_5 and l_6)"に対するセンシティブな条件の組み合わせ

p_2 l_4 or (l_5 and l_6)	l_4	l_5	l_6	Sensitive
T	T	T	T	
T	T	T	F	✓
T	T	F	T	✓
T	T	F	F	✓
T	F	T	T	✓
F	F	T	F	✓
F	F	F	T	✓
F	F	F	F	✓

例 4.10. 図 4.5 の仕様の場合，4 つのリテラルが記述されているが，23 行目は 16 行目の否定であり，24 行目は 17 行目の否定であることから，同値のリテラルを除いた列は $Q_s = (l_1, l_2)$ と 2 つからなる．この時，論理的には全ての真理値を割り当て可能なため，B_s ={(T,T),(T,F),(F,T),(F,F)}となる．

この記法を用いて，BBSL 仕様ベース複合条件カバレッジ BC_{mc} を定義 4.11 に定義する．

定義 4.11. 仕様 S 内のリテラルから同値なものを除いた列を $Q_s = (l_1, l_2, ..., l_n)$ とする．また，Q_s に論理的に割り当てることが可能なブール列の集合を B_s，テストデータを Td とすると BBSL 仕様ベース複合条件カバレッジ BC_{mc} を以下に定義する．

$$BC_{mc}(Td, Q_s) = \frac{|\{Q_s(td)|td \in Td, Q_s(td) \in B_s\}|}{|B_s|}$$

例 4.11. 図 4.5 の場合，B_s ={(T,T),(T,F),(F,T),(F,F)}のため，BC_{mc} の分母は 4 となる．

以上で，BBSL で記述された仕様に対してどの程度テストされたかを示すカバレッジ，BBSL 仕様ベース判断カバレッジ BC_d，BBSL 仕様ベース条件カバレッジ BC_c，BBSL 仕様ベース条件判断カバレッジ BC_{cd}，BBSL 仕様ベース改良条件判断カバレッジ BC_{mcd}，BBSL 仕様ベース複合条件カバレッジ BC_{mc} の 5 つのカバレッジを提案した．これらのカバレッジ指標は，テストが機能仕様をどの程度効果的に反映しているかを評価するための枠組みを構築する．また，不足しているテストケースの条件を特定するための重要な知見を提供し，テストの網羅度を向上させることが期待される．体系的に BBSL 仕様ベースカバレッジを導入することで，テスト実行者は機能仕様の重要な側面を網羅的かつ徹底的に確認することが可能となる．

4.4 空間カバレッジ

空間カバレッジでは，画像上のオブジェクトに関する多様性を仕様とは独立して測定し，テスト終了基準の 1 つとして活用できる形式で定義することを試みる．ここでは，オブジェクトの多様性として，本研究の仕様やテストが焦点を当てているオブジェクトの位置とサイズに関する多様性を用いて，空間位置カバレッジと空間サイズカバレッジを提案

する．しかし，すべての理論的な画像上の位置やサイズの範囲に対して，自動運転システムの画像認識における現実的な位置やサイズは物理的な制約を伴い限定的であることが考えられる．そこで，まず，既存のデータセット上において，オブジェクトの位置やサイズの多様性を測定する．

まず，既存の 3 種類の自動運転システムにおける画像認識のデータセットを用いて計測をおこなう．この計測に用いたデータセットを表 4.4 に示す．arla のデータセットは，自動運転システムの開発と検証用途のシミュレータ Carla によって作成したデータセットであり，特定コース上で常に他車両が 1 台のみ前方カメラに写っている画像を 1059 枚用意したものである [41]．KITTI のデータセットは，自動運転システムの画像認識のためのグラウンドトゥルースラベル付きの既存のデータセットの一部であり，実際の街中を走行してデータをとっているため，一枚の画像に複数の自動車が存在しており，それぞれにラベルが与えられている [21]．Waymo のデータセットも同様に実際の街中を走行して取得された既存のラベル付きのデータセットの一部であるが，こちらは自動車のみでなく，歩行者と自転車についてもラベルが与えられている [25]．しかし，全ての画像に歩行者や自転車が存在してる訳ではない．

表 4.4: 位置関係，大小関係の分析に用いたデータセット

データセット名	対象オブジェクト	画像枚数	オブジェクトの数
Carla	自動車のみ	1059	1059
KITTI	自動車のみ	371	5053
Waymo	自動車, 歩行者, 自転車	1983	34846,6169,31*

* 左から自動車，歩行者，自転車の数

これらのデータセットの各オブジェクトに対して，位置と大小の分布を測定したものを図 4.6，図 4.7，及び，図 4.8 に示す．オブジェクトの位置に関しては，KITTI と Carla によるデータセットである図 4.6 は画像の左上を原点 $(0, 0)$ としており，waymo のデータセットである図 4.8 では左下を原点 $(0, 0)$ としている．

図 4.6 は，左側が Carla で作成したデータセットにおける自動車オブジェクトの位置の分布で，右側がデータセット KITTI における自動車オブジェクトの位置の分布であり，例として各データセットのうち 1 枚を透過して差し込んでいる．また，位置の分布は全て横軸が画像上の x 軸方向，縦軸が画像上の y 軸方向として，オブジェクトの左上座標の位置をプロットしている．左側の Carla で作成したデータセットを見たときに，車線に対して x 軸方向は多様であることが分かるが，y 軸方向にあたる自車との車間距離が偏っており，より車間距離が近い場合などを含んでいない．一方で，グラフ上の下部 (画像上で空が写っている部分) に関しては，そもそも車が空を飛ぶことは考えられないため，テストする必要のない位置に思える．右側の KITTI のデータセットにおいては，Carla の場合とは画像やカメラの規格が異なり，グラフ上の下部もテスト可能な位置である．実際に右下あたりは道路の形状によってはプロットされている．

図 4.7 は，左側が Carla で作成したデータセットにおける自動車オブジェクトの大小の分布で，右側がデータセット KITTI における自動車オブジェクトの大小の分布となる．大小の分布は全て横軸が幅，縦軸が高さとしてプロットしている．画像上に車両の全てが含まれる場合，車両の高さと幅の比率は一定であるため，左側のグラフは同じ比率で固まっている．しかし，前方カメラの場合，車両の横幅は場合によってカメラの画角からはみ出ることが多々あり，KITTI ではそのような画像を多く含むため，右側のグラフでは，縦軸の高さによらず，横軸の幅が多様にプロットされている．

図 4.8 は，上部がデータセット Waymo におけるオブジェクトの位置の分布で，下部が

オブジェクトの大小の分布となる．また，それぞれ，左から順に自動車，歩行者，自転車のオブジェクトについての分布となっている．waymo のデータセットにおいてもプロットされていない領域において，テストする必要のあってプロットされてない部分と，そもそも現実的にありえない位置や大きさのためテストする必要のない部分が存在していることが分かる．

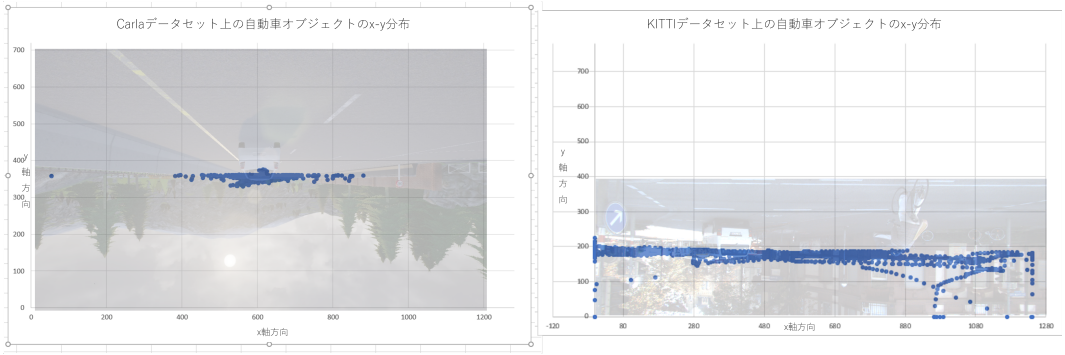

図 4.6: carla と KITTI のデータセットにおけるオブジェクトの位置の分布

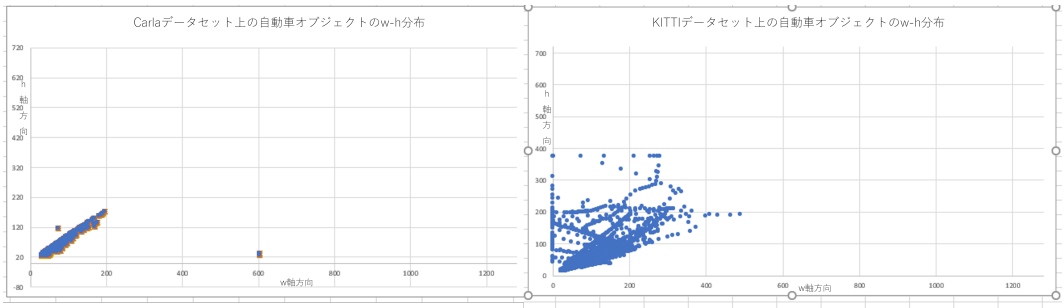

図 4.7: carla と KITTI のデータセットにおけるオブジェクトの大小の分布

以上より，どの実験結果からも，オブジェクトの位置や大小について，自動運転システムの走行環境という特性上，カメラや想定される交通状況によって，非現実的で考慮する必要のない位置や大きさが含まれていることが分かる．また，それらは位置と大きさそれぞれに個別で存在し，使用するカメラなどによっても変わるため，事前に汎用的なものを定義することは困難であることがわかる．そこで，ここで提案するそして，それぞれのカバレッジにおける位置の範囲や大きさの範囲は，テスト設計者が個別で与えることが可能なように柔軟性を持たせて定義することにする．

そこで，初めに，空間位置カバレッジの準備として，位置クラスという概念を定義 4.12 に定義する．この位置クラスを，テスト設計者が事前にテストしたい領域として定めるクラスとして使用し，カバレッジを定義する．この定義では，バウンディングボックス間の位置関係に基づいて条件を定義するために一部 BBSL の演算子を使用している．

定義 4.12. 以下の条件を満たすバウンディングボックスの集合 $R = \{r_1, r_2, ..., r_n\} \subseteq BB$ を位置クラスと呼ぶ．

- For all $a \in R.(w(PROJ_x(a)) \neq 0$ and $w(PROJ_y(a)) \neq 0)$
- For all $a, b \in R.(RAT(a \cap b, a) = 0)$

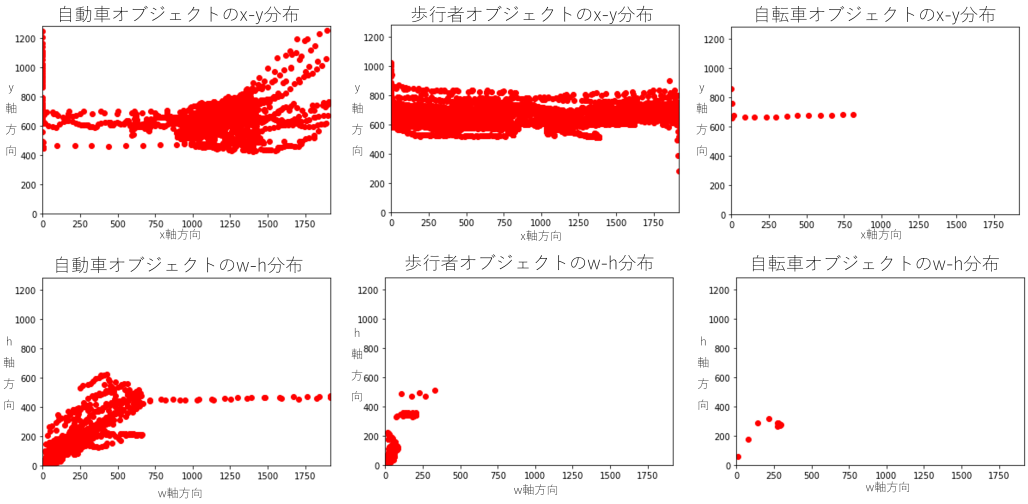

図 4.8: Waymo のデータセットにおける各オブジェクトの位置と大小の分布

つまり，位置クラスとは，互いに重なりあわず，どの辺の長さも 0 ではないバウンディングボックスからなる集合である．

例 4.12. 図 4.9 は，位置クラスの例を画像上に可視化したものである．図 4.9 に表現しているバウンディングボックス r_1 から r_{24} を要素に持つ $R = \{r_1, r_2, ..., r_{24}\}$ は位置クラスである．このように，設計者は位置クラスを定めることで，テストに関心のある領域に加えて，特に，注意深くテストすべき範囲などを，バウンディングボックスの大きさによって指定することができる．

図 4.9: 位置クラスの例 $R = \{r_1, r_2, ..., r_{24}\}$

次に，画像内の任意のオブジェクトを位置クラス R に割り当てる関数 POS を定義 4.13 に定義する．この定義でも，バウンディングボックス間の位置関係に基づいて条件を定義するために一部 BBSL の演算子を使用している．

定義 4.13. 位置クラス $R = \{r_1, r_2, ..., r_n\}$ とテストデータのグラウンドトゥルースに含まれるオブジェクトのバウンディングボックスの集合 T が与えられた時，任意のバウンディ

ングボックス $t \in T$ について，関数 $POS : T \to R \cup \{\perp\} (\perp \notin BB)$ を以下で定義する．

$$POS(t) = \begin{cases} & PROJ_{\underline{x}}(t) \subseteq PROJ_x(r_i) \\ & \text{and } PROJ_{\underline{y}}(t) \subseteq PROJ_y(r_i) \\ r_i & \text{and } not(PROJ_{\underline{y}}(t) = PROJ_{\overline{y}}(r_i)) \\ & \text{and } not(PROJ_{\underline{x}}(t) = PROJ_{\overline{x}}(r_i)) \\ & , i \in \{1, 2, ..., n\} \\ \perp & otherwise \end{cases}$$

つまり，画像の左上座標を原点 (0,0) とした場合，$POS(t)$ は，バウンディングボックス t を受け取って，その左上座標を含む R 内のバウンディングボックス (どのバウンディングボックスにも含まれなければ $\perp$) を返す関数である．

例 4.13. 図 4.10 は，図 4.9 と同じ画像と位置クラスに対して，オブジェクトのバウンディングボックス t1 と t2 を示した図である．先ほど図 4.9 で例にあげた位置クラス $R = \{r_1, r_2, ..., r_{24}\}$ の場合，図 4.10 で車両を囲む白いバウンディングボックス $t1$ と $t2$ を関数 POS で受け取ると，それぞれ $POS(t_1) = r_5$ と $POS(t_2) = r_1$ になる．

図 4.10: 位置クラス R に対する関数 POS の例

以上の R と POS を用いて，空間位置カバレッジを定義 4.14 に定義する．

定義 4.14. 任意のポジションクラス R とテストデータのグラウンドトゥルースに含まれるオブジェクトの任意のバウンディングボックスの集合 T について，空間位置カバレッジ SC_{pos} を以下で定義する．

$$SC_{pos}(T, R) = \frac{|\{POS(t)|t \in T, POS(t) \in R\}|}{|R|}$$

例 4.14. 先ほどと同様に図 4.9 及び図 4.10 で示したバウンディングを例にすると，$T = \{t_1, t_2\}$,$R = \{r_1, r_2, ..., r_{24}\}$ となり，この時の空間位置カバレッジは $SC_{pos}(T, R) = 2/24$ となる．

次に，もう一つの空間カバレッジとして，空間サイズカバレッジを提案する．空間サイズカバレッジの準備として，サイズクラスという概念を定義 4.15 に定義する．空間位置カバレッジにおける位置クラスと同様に，このサイズクラスでもテスト設計者が事前にテストしたいサイズを定めるクラスとして使用し，カバレッジを定義する．

定義 4.15. サイズクラスは，空でない 0 以上の実数の有限集合 $S = \{s_0, s_1, s_2, ..., s_n\} \subseteq \mathbb{R}_{\geq 0}$ で定義する．

例 4.15. $S_1 = \{20, 1000, 3000\}, S_2 = \{0, 5, 10, 15, 20\}$,
$S_3 = \{5, 10, 1000\}$ はどれもサイズクラスである．このように，設計者はサイズクラスに与える実数によって，テストに関心のあるサイズに加えて，特に，注意深くテストすべきサイズなどを，細かく指定することができる．

次に，画像内の任意のオブジェクトをサイズクラス S に割り当てる関数 SIZ を定義 4.16 に定義する．この定義でも，バウンディングボックス間の位置関係に基づいて条件を定義するために一部 BBSL の演算子を使用している．

定義 4.16. サイズクラス $S = \{s_0, s_1, s_2, ..., s_n\}, s_i < s_{i+1}$ とテストデータのグラウンドトゥルースに含まれるオブジェクトのバウンディングボックスの集合 T が与えられた時，任意のバウンディングボックス $t \in T$ について，関数 $SIZ : T \to S \cup \{\perp\} (\perp \notin \mathbb{R}_{\geq 0})$ を以下で定義する．

$$SIZ(t) = \begin{cases} & s_{i-1} \\ s_i & < w(PROJ_x(t)) \times w(PROJ_y(t)) \\ & \leq s_i \\ & , i \in \{1, 2, ..., n\} \\ \perp & otherwise \end{cases}$$

つまり，受け取ったバウンディングボックスの面積を計算し，その面積が s_0 以下または s_n より大きければ $\perp$ を，そうでなければその面積以上で最も近い実数を返す．

例 4.16. サイズクラス $S = \{20, 1000, 3000\}$, $t1 = ([0, 30], [200, 250]), t2 = ([50, 52], [210, 215])$ を例に考えると，それぞれ $SIZ(t_1) = 3000$ と $SIZ(t_2) = \perp$ となる．

以上の S と SIZ を用いて，空間サイズカバレッジを定義 4.17 に定義する．

定義 4.17. 任意のサイズクラス S とテストデータのグラウンドトゥルースに含まれるオブジェクトの任意のバウンディングボックスの集合 T について，空間サイズカバレッジ SC_{siz} を以下で定義する．

$$SC_{siz}(T, S) = \frac{|\{SIZ(t)|t \in T, SIZ(t) \in S)\}|}{|S| - 1}$$

例 4.17. 先ほど用いたサイズクラス $S = \{20, 1000, 3000\}$ と $t1 = ([0, 30], [200, 250])$, $t2 = ([50, 52], [210, 215])$ からなる $T = \{t_1, t_2\}$ を例に考えると，この時の空間サイズカバレッジは $SC_{siz}(T, S) = 1/2$ となる．

以上に定義した空間位置カバレッジと空間サイズカバレッジを用いて，暫定的な飽和判定を行えるようにする．ここでは，空間位置カバレッジと空間サイズカバレッジそれぞれにおいて飽和したかどうかを判定できるように定義する．これらの空間カバレッジを用いて，飽和したとみなす基準を定義 4.18 に定義する．

定義 4.18. R は位置クラス，S はサイズクラス，SC_{pos} は空間位置カバレッジ，SC_{siz} は空間サイズカバレッジ，T はテストデータのグラウンドトゥルースに含まれるオブジェクトの任意のバウンディングボックスの集合とする．ここで，任意の $t_i \in T$ について，$T_1 = \{t_1\}, T_2 = \{t_1, t_2\}$ とする $T_n = \{t_1, t_2, ..., t_n\}$ を与える．この時，ウィンドウサイズ $h \in \mathbb{N}$ を用いて，以下の条件を満たす場合に T が SC_{pos} において飽和していると呼ぶ．

$$\exists i \in \mathbb{N}.(SC_{pos}(T_i, R) - SC_{pos}(T_{i-h}, R) = 0)$$

同様に，ウィンドウサイズ $h \in \mathbb{N}$ を用いて，以下の条件を満たす場合に T が SC_{siz} において飽和していると呼ぶ．

$$\exists i \in \mathbb{N}.(SC_{siz}(T_i, S) - SC_{siz}(T_{i-h}, S) = 0)$$

以降，この h をウィンドウサイズと呼ぶ.

例 4.18. $r_1 = ([0, 40], [200, 260])$，$r_2 = ([50, 70], [210, 275])$，$r_3 = ([80, 100], [220, 290])$ からなる位置クラス $R = \{r_1, r_2, r_3\}$ と，サイズクラス $S = \{20, 1000, 3000\}$ を用いて考える．$t_1 = ([0, 30], [200, 250])$，$t_2 = ([50, 55], [210, 215])$，$t_3 = ([60, 80], [220, 270])$ からなるバウンディングボックスの集合 $T = \{t_1, t_2, t_3\}$ に対して，ウィンドウサイズ $h = 1$ とすると，空間位置カバレッジについては，$SC_{pos}(T_2, R) = SC_{pos}(T_1, R) = 2/3$ かつ $SC_{pos}(T_3, R) = 2/3$ より，$SC_{pos}(T_3, R) - SC_{pos}(T_2, R) = 0$ であるため，T は SC_{pos} で飽和していると見なされる．同様に，空間サイズカバレッジについては，$SC_{siz}(T_2, S) = 2/2$ かつ $SC_{siz}(T_3, S) = 2/2$ より，$SC_{siz}(T_3, S) - SC_{siz}(T_2, S) = 0$ であるため，T は SC_{siz} で飽和していると見なされる．

テストケースを順次実行している時に，カバレッジが一定数のケースにわたって全く変化しない場合に飽和状態と見なす．ここでのウィンドウサイズ h は，何ケースの間カバレッジが変化しない場合に，飽和状態と判断するかを定める．ここでは，オブジェクトの位置とサイズによって飽和度を測定して，この 2 種類が飽和したときに充分にテストしきったとして，テストの暫定的な停止条件とするが，オブジェクトの種類や，オブジェクト以外の環境，道路状況等既存の ODD のカバレッジを取り入れることで，より効果的にテストの状態を測定できる可能性がある．このように，空間カバレッジにより，テスト実行者は目的の位置とサイズを位置クラスとサイズクラスとして定義し，これらの側面がどの程度充分にテストされたかを定量的に測定できる．ウィンドウサイズを適切に調整することで，これらのカバレッジは，テストプロセスが十分にテストしきったかどうかを判断するための指標として機能する．

第5章　評価

5.1　既存のOEDRの仕様をBBSLで記述

ここでは，BBSLの表現力を確認するために，既存の自動運転システムの設計フレームワークを参考に，複数のシナリオについてBBSLでOEDRタスクの仕様を記述する．具体的には，NHTSAが提案した自動運転システムの設計フレームワーク [2] を用いる．この中から，SAEレベル3の渋滞時の自動運転,level 3 conditional automated traffic jam drive (L3TJD) についての仕様を対象とする．表5.1に，NHTSAのペーパーで例示されている，L3TJDについてのオブジェクトとイベントに対する自動運転システムの反応についてのリストを示す．例えば，一番上のLead vehicle deceleratingは，前方の車両が速度を落とし始めた状態のことをさしており，その時，自車は先行車両との適切な車間距離を維持しながら追従する動作か，減速か，停止すべきであることが定められている．このように，NHTSAでは，開発する自動運転システムに対して，このように遭遇するイベントを示したうえで，そのイベント上でとりうる自動運転システムの反応をまとめている．NHTSAや既存のフレームワークでは，このようにリストアップした上で，具体的なテストを作成するために，各々のレスポンスの条件を定めるには至っていない．そこで，BBSLで記述する仕様では，各イベント毎に定められているレスポンスをcaseとして記述する機能として，それらの機能の条件をBBSLで記述する．

表5.1のOEDRについて，各イベント内の自動運転システムのとりうる反応を条件を考えながらBBSLケースとして仕様として記述した結果として，BBSLで記述した結果の行数と仕様内のケース数をまとめたものを表5.2に示す．実際に記述した仕様はすべてgithub[42] 上の"Examples"ディレクトリに画像ファイルとして公開している．ここで，行数はBBSLで記述した時の各仕様の行数を示し，ケース数はケースブロックの数を示している．仕様内のケースは表5.1で示した自動運転システムの反応に対応したうえで，何の反応も必要ない状況もケースとして記述しているため，基本的にケース数は表5.1で記述されている反応数より1つ多い．最も行数の多い仕様でも69行程度であり，充分手動で定義できるレベルの分量であることが分かる．

表 5.1: NHTSA のペーパーで例示している L3TJD についてのオブジェクトとイベントに対する自動運転システムの反応 [2]

Event	Response
Lead vehicle decelerating	Follow vehicle, decelerate, stop
Lead vehicle stopped	Decelerate, stop
Lead vehicle accelerating	Accelerate, follow vehicle
Lead vehicle turning	Decelerate, stop
Vehicle changing lanes	Yield, decelerate, follow vehicle
Vehicle cutting in	Yield, decelerate, stop, follow vehicle
Vehicle entering roadway	Follow vehicle, decelerate, stop
Opposing vehicle encroaching	Decelerate, stop, shift within lane, shift outside of lane
Adjacent vehicle encroaching	Yield, decelerate, stop
Lead vehicle cutting out	Accelerate, decelerate, stop
Pedestrian crossing road – inside crosswalk	Yield, decelerate, stop
Pedestrian crossing road – outside of crosswalk	Yield, decelerate, stop
Pedalcyclist riding in lane	Yield, follow
Pedalcyclist riding in dedicated lane	Shift within lane
Pedalcyclist crossing road – inside crosswalk	Yield, decelerate, stop
Pedalcyclist crossing road – outside crosswalk	Yield, decelerate, stop
Debris static in lane	Decelerate, stop
Dynamic object in lane	Decelerate, stop

表 5.2: 表 5.1 の OEDR を BBSL で記述した結果

Event	行数	ケース数
Lead vehicle decelerating	42	4
Lead vehicle stopped	30	3
Lead vehicle accelerating	30	3
Lead vehicle turning	38	3
Vehicle changing lanes	50	3
Vehicle cutting in	55	4
Vehicle entering roadway	55	4
Opposing vehicle encroaching	69	5
Adjacent vehicle encroaching	59	4
Lead vehicle cutting out	45	4
Pedestrian crossing road – inside crosswalk	65	4
Pedestrian crossing road – outside of crosswalk	65	4
Pedalcyclist riding in lane	33	2
Pedalcyclist riding in dedicated lane	29	2
Pedalcyclist crossing road – inside crosswalk	65	4
Pedalcyclist crossing road – outside crosswalk	65	4
Debris static in lane	44	3
Dynamic object in lane	45	3

5.2 BBSLと既存手法との位置関係の記述に関する比較

ここでは，画像内のオブジェクトの位置関係を記述することができる既存の手法とBBSLを比較し，それらの違いを評価する．具体的に比較対象として用いた既存手法は，画像認識システムの評価基準として一般的に使用されるIoU[11][20]，空間データベース管理システムのクエリなどに使用されるbinary topological relationship(BTR)[34][35]，自動運転システム内でも経路探索アルゴリズムの設計などに利用される幾何学的な記述[5]である．

初めに，画像認識システムで広く使用されている評価指標であるIoUをBBSLを比較する．IoUは厳密には計算指標であるが，重なりに基づいてオブジェクト間の位置関係を間接的には説明ができる．BBSLとIoUの比較結果を表5.3に示す．"Ratio"は2つのバウンディングボックス間のサイズの比率を表しており，これはIoUで本来計算する対象であり，BBSLでも記述可能である．例えば，AとBの2つのバウンディングボックスについて，RAT(A ∩ B, A ∪ B)と記述することでIoUと同等の記述が可能である．そして，BBSLもIoUも2つのバウンディングボックスのサイズの大小関係を記述することができる．例えば，画像全体を表すバウンディングボックスUを用いることで，UとAのIoUの値と，UとBのIoUの値を比較することで間接的にサイズの大小を記述することができるただし，BBSLは，IoUでは表現できない頂点を包含しているかどうかや，方向を含めた位置関係の記述に優れている．一方で，IoUの値を計算機上で計算することは非常にシンプルで簡単ですが，BBSLでは独自のパーサー等を使用しない限り，計算機上で計算することはできず，現状は自動計算するための汎用的なツールは提供されていない，これらの点がBBSLとIoUの主な違いとなる．

表 5.3: BBSLとIoUの比較

Evaluation Item	BBSL	IoU
Ratio	can describe	can describe
Positional relationship	can describe	cannot describe
Inclusion relationship	can describe	can describe
Magnitude relationship	can describe	can describe
Distance	can describe	cannot describe
Division into the x-axis and y-axis	can divide	cannot divide
Application to object detection system evaluation	not yet fully established	well-established

次に，地理情報システムや3D空間モデリングにおいて利用されるBTRとBBSLを比較する．BBSLとBTRの比較結果をまとめた結果を表5.4に示す．"A contains B"から"B on A"までの項目は，BTR上で定義している関係の名前を用いており，節2.7で説明した図2.8に示す関係と同じである．これらはすべてBBSLでも記述することができる．具体的に，"A contains B"と等価な条件をBBSLで記述した例を図5.1に示す．これは，バウンディングボックスの各始点，終点が一致していないことを$PROJ$関数を用いて記述し，BはAに包含していることをx軸，y軸それぞれの区間の包含関係で記述している．次に．"A covers B"と等価な条件をBBSLで記述した例を図5.2に示す．この場合は，各境界と接触する場合も許容される関係のため，各区間に対して包含関係$\subseteq$のみで記述することができる．次に，"A touch B"と等価な条件をBBSLで記述した例を図5.3に示す．この例では，AとBが内部が重複していないことをRAT関数を用いて，AとBの重複部分の面積が0であることで記述し，各点の比較でいずれかど接触していることを記述してい

る．次に，”A overlapbdyintersect B ”と等価な条件を BBSL で記述した例を図 5.4 に示す．この例では，A と B が部分的に重なっていることを，必ず共通部分の面積が 0 でないことで記述し，かつ，それぞれに独立した領域があることを，A も B も互いに包含していない条件で記述している．次に，”A overlapbdydisjoint B ”と等価な条件を BBSL で記述した例を図 5.5 に示す．この例では，A と B について，x 軸の区間も y 軸の区間も等しくなく，重複があることを記述した上で，内部が重ならないことを共通部分の面積が 0 であることを記述している．次に，"A equal B"と等価な条件を BBSL で記述した例を図 5.6 に示す．この例では，A と B は x 軸の区間も y 軸の区間も等しいという条件を用いて，A と B が大きさ，位置ともに完全に一致していることを記述している．次に，"A disjoint B" と等価な条件を BBSL で記述した例を図 5.7 に示す．この例では，4 方向それぞれに分離している条件を用いることで，A と B がいずれかの方向に分離している条件を記述している．次に，"B on A"と等価な条件を BBSL で記述した例を図 5.8 に示す．この例では，関数 w を用いて，B が内部の領域を持たない条件を記述し，さらに，A のいずれかの境界の上にはみ出さずに乗っていることを，y 軸，x 軸それぞれにおいて場合分けをして条件を記述している．加えて，BBSL では，BTR では記述できない比率に関する記述 (Ratio) や，距離に関する記述 (Distance)，そして，大小関係に関する記述"Magnitude relationship" を記述することができる．ただし，BBSL とは異なり，BTR では多次元空間への拡張が定義されているため，例えば 3 次元の区間上での位置関係を記述するのは容易である．現状，BBSL では 3 次元空間においてどのように記述できるかは未解決である．これらの点が BBSL と BTR の主な違いとなる．

表 5.4: BBSL と BTR の比較

Evaluation Item	BBSL	BTR
A contains B	can describe	can describe
A covers B	can describe	can describe
A touche B	can describe	can describe
A overlapbdyintersect B	can describe	can describe
A overlapbdydisjoint B	can describe	can describe
A equal B	can describe	can describe
A disjoint B	can describe	can describe
B on A	can describe	can describe
Ratio	can describe	cannot describe
Distance	can describe	cannot describe
Magnitude relationship	can describe	cannot describe
Extension over multiple dimensions	not established	established

最後に，表 5.4 に，特に抽象化レベルの観点から，BBSL と幾何学的記述の違いを示す．幾何学的記述では，具体的な実数値を使用して正確で詳細な表現が可能になるため，高精度が求められる記述に非常に適している．ただし，その反面複雑さをもたらし，広範囲にわたる詳細なデータ入力が必要となる．対照的に，BBSL はより抽象的なレベルで記述が可能であり，特定の数値を必要とせずに空間関係を記述できる．この抽象化は，自動運転システムの OEDR における抽象度の高い仕様など，正確な数値精度よりも一般性と柔軟性が重要なシナリオで有効である．これらの点が BBSL と幾何学的な記述の主な違いとなる．

```
case "A contains B"
    let A : bb = getA(),
      B : bb = getB() in
         not(PROJ_ȳ(A) = PROJ_ȳ(B))
         and not(PROJ_y(A) = PROJ_y(B))
         and not(PROJ_x̄(A) = PROJ_x̄(B))
         and not(PROJ_x(A) = PROJ_x(B))
         and PROJ_y(B) ⊆ PROJ_y(A)
         and PROJ_x(B) ⊆ PROJ_x(A)
endcase
```

図 5.1: "A contains B"と等価な BBSL の条件の例

```
case "A covers B"
    let A : bb = getA(),
      B : bb = getB() in
         PROJ_y(B) ⊆ PROJ_y(A)
         and PROJ_x(B) ⊆ PROJ_x(A)
endcase
```

図 5.2: "A covers B"と等価な BBSL の条件の例

```
case "A touch B"
    let A : bb = getA(),
      B : bb = getB() in
         RAT(A∩B, A∪B) = 0
         and (PROJ_ȳ(A) = PROJ_ȳ(B)
         or PROJ_y(A) = PROJ_y(B)
         or PROJ_x̄(A) = PROJ_x̄(B)
         or PROJ_x(A) = PROJ_x(B))
endcase
```

図 5.3: "A touch B"と等価な BBSL の条件の例

```
case "A overlapbdyintersect B"
    let A : bb = getA(),
      B : bb = getB() in
         not(PROJ_y(B) ⊆ PROJ_y(A)
         and PROJ_x(B) ⊆ PROJ_x(A))
         and not(PROJ_y(A) ⊆ PROJ_y(B)
         and PROJ_x(A) ⊆ PROJ_x(B))
         and not(RAT(A∩B, A∪B) = 0)
endcase
```

図 5.4: "A overlapbdyintersect B"と等価な BBSL の条件の例

```
case "A overlapbdydisjoint B"
    let A : bb = getA(),
      B : bb = getB() in
         A ≈ B
         and RAT(A∩B, A∪B) = 0
endcase
```

図 5.5: "A overlapbdydisjoint B"と等価な BBSL の条件の例

```
case "A equal B"
    let A : bb = getA(),
      B : bb = getB() in
         PROJ_y(A) = PROJ_y(B)
         and PROJ_x(A) = PROJ_x(B)
endcase
```

図 5.6: "A equal B"と等価な BBSL の条件の例

```
case "A disjoint B"
    let A : bb = getA(),
      B : bb = getB() in
         PROJ_y(A) < PROJ_y(B)
         or PROJ_y(B) < PROJ_y(A)
         or PROJ_x(A) < PROJ_x(B)
         or PROJ_x(B) < PROJ_x(A)
endcase
```

図 5.7: "A disjoint B"と等価な BBSL の条件の例

```
case "B on A"
    let A : bb = getA(),
      B : bb = getB() in
         w(PROJ_y(B))=0
         and (PROJ_y(A) = PROJ_y(B)
         or PROJ_ȳ(A) = PROJ_ȳ(B))
         and PROJ_x(B) ⊆ PROJ_x(A)
         or w(PROJ_x(B))=0
         and (PROJ_x(A) = PROJ_x(B)
         or PROJ_x̄(A) = PROJ_x̄(B))
         and PROJ_y(B) ⊆ PROJ_y(A)
endcase
```

図 5.8: "B on A"と等価な BBSL の条件の例

まとめると，BBSL は抽象度と表現力に関して既存の手法と明確な違いがある．画像内の位置関係，距離，大きさを正確に記述できるため，自動運転システムの OEDR タスクの形式仕様に特に適している．ただし，計算を自動化させるためにはパーサーの実装が必要であるなど，現状ではいくつかの制限が存在している．

表 5.5: BBSL と幾何学的な記述の比較

Evaluation Item	BBSL	Geometric Description
Description using specific values	worse	better
Description without specific values	better	worse

5.3 BBSL を用いた機能テストの実施

提案した機能テストの性能を評価するため，自動運転システム用の画像認識システム，2 次元のバウンディングボックスを持つグラウンドトゥルースデータセット，および，BBSL で記述された仕様を準備して，テストを行う．テスト対象の自動運転システム用の画像認識システムとして，4 つのシステム (SUT1，SUT2，SUT3，および，SUT4) を準備した．これらは，YOLOv3，および，YOLOv8 に基づく既存のモデルを使用して準備した．また，グラウンドトゥルースデータセットには，KITTI のデータセットの一部を利用して作成した．BBSL で記述された形式仕様は，4 種類の機能仕様 (S_1,S_2,S_3,S_4) を準備した．ここでは，比較として，章で提案した機能テストに加えて，評価基準にはしきい値 0.6 の $IoU_{0.6}$ と 0.8 の $IoU_{0.8}$ を用いる．以降で，これらの要素についての詳細を説明する．

まず，KITTI のデータセットを 2 種類のテストデータセット Td_1 と $Td_2(Td_1 \cap Td_2 = \emptyset)$ として使用し，グラウンドトゥルースデータ GT を準備した．表 5.6 は，テストデータセット Td_1 と Td_2 の枚数と車両オブジェクトの数を示している．それぞれ自動運転システムの正面カメラから撮影された 349 枚と 1300 枚の画像で構成されている．これらの画像には，それぞれ 1 台以上の車両が含まれており，各データセット内の車両の数はそれぞ

れ 2736 台と 5644 台である．提案テストのテストケース数は定義上，テストの画像枚数に該当するが，本論文での実験に用いる仕様は単純化のために，全て画像上の 1 つの車両に対する位置関係で定義している．そのため，以降特に記述がない限り，実験上は各画像上の車両オブジェクト毎に仕様を対応させており，実験上でのテストケース数は車両オブジェクトの数と一致している．また，画像は 8 ビット PNG ファイルとして提供されており，画像サイズは今回実験に使用したサイズはすべて 1242×375 であり，アノテーションのクラスは Car,Van,Truck,Pedestrian,Person,Cyclist,Tram,Misc となっており，ここでは，Car,Van,Truck をすべて車両としている，グラウンドトゥルースラベルは，各車両を囲む 2 次元のバウンディングボックスで定義されている．データセット Td_1 と Td_2 は，Zenodo 上で公開されている [43]．Td_1 の画像は image1.zip ファイルで入手でき，対応するグラウンドトゥルースラベルは label1.zip ファイルにある．同様に，Td2 の画像は image2.zip ファイルに保存され，そのグラウンドトゥルースラベルは label2.zip ファイルにある．これらのデータセットは，Creative Commons Attribution-NonCommercialShareAlike 3.0 ライセンスの下で公開されており，同じ条件の下で非営利目的での使用と共有が許可されている．

表 5.6: 準備したテストデータセット

名前	画像の枚数	車両オブジェクトの数
Td_1	349	2736
Td_2	1300	5644

次に，テスト対象の画像認識システムを 4 種類 $SUT1$,$SUT2$,$SUT3$,$SUT4$ 準備した．これらは，すべて DNN ネットワークとして darknet53 を用いている．Darknet53 は,YOLO シリーズの物体検出アルゴリズムで使用されるバックボーンネットワークである．このネットワークは，元々YOLOv3 のために設計されており，53 層の畳み込みニューラルネットワーク（CNN）で構成されている．Darknet53 は，軽量でありながら高い計算効率を持ち，特徴抽出の性能が優れているため，物体検出タスクにおいて広く利用されている．採用している検出アルゴリズムは，$SUT1$ と $SUT2$ には YOLOv3[44] を，$SUT3$ と $SUT4$ には YOLOv8[45] を用いている．YOLOv3 は，物体検出の分野で広く利用されているアルゴリズムであり，リアルタイムでの処理が可能な点が特徴である．YOLOv8 は，YOLO シリーズの中でも比較的に新しいバージョンであり，一般に従来の YOLO アルゴリズムに比べて精度と効率性が向上していると言われている．具体的には，$SUT1$ では，yolov3-kitti.weights (`https://drive.google.com/file/d/1BRJDDCMRXdQdQs6-x-3PmlzcEuT9wxJV/view`) という重みファイルを用いている．これは，An の GitHub プロジェクト [46] で提供されている KITTI のデータセットを使用してトレーニングされた重みファイルである．また，$SUT2$ では，yolov3.weights(`https://github.com/patrick013/Object-Detection---Yolov3/blob/master/model/yolov3.weights`) という重みファイルを用いており，Yolov3 から提供されている Microsoft Common Objects in Context(COCO) データセット [23] を使用してトレーニングされた重みファイルである．さらに，$SUT3$ では，vehicle_kitti_v0_best.pt というモデルを用いており，これは，Didi の GitHub プロジェクト [47] で提供されており，KITTI データセットでの車両検出用に特別にトレーニングされ, さまざまな検出シナリオに最適化されているモデルである．一方，$SUT4$ では，YOLOv8n.pt モデルというモデルを用いている．これは，YOLOv8 で提供されている COCO データセットでトレーニングされたモデルである．

さらに，このテストでは BBSL で記述したシンプルな仕様を S_1, S_2, S_3,S_4 の 4 種類用いる．仕様 S_1 は，既に図 3.7 で示した仕様と同様のものであり，自車と他車との y 軸方向の距

離，つまり車間距離によって，自動運転システムが停止しなければならない条件と，停止する必要がない条件を定義した仕様である．仕様 S_2 の仕様を図 5.9 に示す．この仕様は，自車と他車との x 軸方向の位置関係，具体的には，自車の前進方向に他車が位置しているかによって，自動運転システムが停止しなければならない条件と，停止する必要がない条件を定義した仕様である．仕様 S_3 は，既に図 4.5 で示した仕様と同様のものであり，仕様 $S1$ と仕様 $S2$ を組み合わせて，x 軸，y 軸どちらの位置関係も用いてケース"stop"とケース"NOT stop"を定義した仕様である．仕様 S_4 の仕様を図 5.10 に示す．この仕様は，仕様 S_1 と仕様 S_2 の条件を組み合わせたうえで，"x_ystop"，"ysafe_xwarning"，"xsafe_ywarning"，"not warning"の 4 種類のケースを定義した仕様であり，"Not stop"のケースを y 座標と x 座標の関係によってさらに細かく分けたものとなる．これらの各仕様については，節 3 で定義した BBSL のセマンティックを使用して，python で計算できるように実装を行った．S_1 から S_4 は，すべて特定の 1 つの車両オブジェクトで条件を記述しているため，前述したとおり，複数の車両が含まれる画像であっても各車両 1 つ 1 つに対してテストケースを定めている．そのため，各テストデータセット Td_1 と Td_2 におけるテストケース数はそれぞれ 2736,5644 となり，これは，それぞれの車両オブジェクトの数に相当する．

テストを実行するために，もう 1 つ定めなければいけない要素がある．それは，BBSL で記述された仕様における外部関数ブロック ($\langle exfunc\_block \rangle$) 内で定義される外部関数を用意することである．車両を表すバウンディングボックスは，グラウンドトゥルースデータや，画像認識の出力から得れるため，ここで準備する必要がある外部関数は，"stoppingDistance()"と"directionAreaDistance()"である．今回は，どちらの外部関数も定数として与えた．

以上に準備した項目を組み合わせて，22 種類のテストを定義した．その 22 種類のテストの内訳を表 5.7 に示す．値はすべてにおいて画像の左上を原点 (0,0) とした座標の値であり，画像のサイズは 1242 × 375 のものを使用している．外部関数における"sD()"は"stoppingDistance()"の略称であり，"dA()"は"directionAreaDistance()"である．これらのテストを実行するためのスクリプトは，GItHub 上に公開している [48]．BBSL で記述された仕様を python コードに変換できる汎用的なパーサーは現在未実装であるため，仕様 $S1$ から $S4$ に該当するスクリプトが，"bbsl_test_sample"ディレクトリ下の，それぞれ"bbsl_specification1"から"bbsl_specification4"に配置してある．SUT_1 と SUT_3 はどちらも KITTI データセットに基いてトレーニングされたモデルで，SUT_1 は YOLOv3 を使用し，SUT_3 は YOLOv8 を使用する．一方で，SUT_2 と SUT_4 は COCO データセットに基いてトレーニングされたモデルであり，SUT_2 は YOLOv3 を使用し，SUT_4 は YOLOv8 を使用している．したがって，$ID1$ や $ID1^*$ のように,IDX と IDX^* といったテスト ID は，同じテスト条件の上でテスト対象のシステムが YOLOv3 の場合が IDX，YOLOv8 の場合が IDX^* となっている．

表 5.7 に示した 22 種類のテストを実施した結果を表 5.8 に示す．ここでの，$IoU_{0.6}$ は，対象の車両のグラウンドトゥルースのバウンディングボックスと SUT によって出力されたバウンディングボックスの間で計算され，IoU 値が 0.6 以上の場合は T，0.6 を下回る場合は F を割り当てている．$IoU_{0.8}$ は，同様のバウンディングボックスに対して，IoU 値が 0.8 以上の場合は T，0.8 を下回る場合は F を割り当てている．対して最も右の列の，「(T/T+F) in BBSL」は，定義 4.3 で示した機能テストであり，仕様を介してグラウンドトゥルースのバウンディングボックスを使用して計算されたケースが SUT によって出力されたバウンディング ボックスを使用して計算されたケースと一致する場合に T を割り当て，ケースが異なる場合に F を割り当てている．すべてのテストにおいて，提案手法は IoU ベースの手法よりも T の判定が多くなっている．これは，自車両から遠い車両は IoU 値が低くなる傾向があるため，多くの場合 F となるが，このような遠くの車両は，自動運転システムの安全性に影響を与える可能性は低いという意味で仕様上あまり重要ではな

いためである．この結果の詳細は，6.3 節でさらに詳しく考察ともに掘り下げる．さらに，$IoU_{0.8}$ の評価のみに着目した時，YOLOv8 はすべてのテストにおいて一貫して YOLOv3 よりも T が多く優れていることが分かるが，提案手法の機能テストにおける結果は必ずしもこの YOLOv8 の方が良いことを示しているわけではない．例えば，$ID1$ と $ID1^*$ を比較した場合，$IoU_{0.8}$ では，YOLOv3 でトレーニングした SUT をテストした $ID1$ では，52.3%であり，YOLOv8 でトレーニングした SUT をテストした $ID1^*$ では，61.0%となり精度は向上している．一方で，提案した機能テストでは，$ID1$ では，92.4%であり，$ID1^*$ では，88.8%となり，結果は下がっている．これは，$IoU_{0.8}$ によって画像認識の精度の向上を示している場合でも，その画像認識を用いる自動運転システムの仕様によっては，安全性に関しては低下している可能性があることを示唆している．

表 5.7: 準備した 22 種類のテスト

Test ID	S	外部関数	Td	SUT
$ID1$	S_1	sD()=[275,375]	Td_1	SUT_1
$ID1^*$	S_1	sD()=[275,375]	Td_1	SUT_3
$ID2$	S_1	sD()=[275,375]	Td_1	SUT_2
$ID2^*$	S_1	sD()=[275,375]	Td_1	SUT_4
$ID3$	S_1	sD()=[250,375]	Td_1	SUT_1
$ID3^*$	S_1	sD()=[250,375]	Td_1	SUT_3
$ID4$	S_1	sD()=[300,375]	Td_1	SUT_1
$ID4^*$	S_1	sD()=[300,375]	Td_1	SUT_3
$ID5$	S_2	dA()=[420,821]	Td_1	SUT_1
$ID5^*$	S_2	dA()=[420,821]	Td_1	SUT_3
$ID6$	S_3	sD()=[275,375], dA()=[420,821]	Td_1	SUT_1
$ID6^*$	S_3	sD()=[275,375], dA()=[420,821]	Td_1	SUT_3
$ID7$	S_4	sD()=[275,375], dA()=[420,821]	Td_1	SUT_1
$ID7^*$	S_4	sD()=[275,375], dA()=[420,821]	Td_1	SUT_3
$ID8$	S_3	sD()=[275,375], dA()=[420,821]	Td_1	SUT_2
$ID8^*$	S_3	sD()=[275,375], dA()=[420,821]	Td_1	SUT_4
$ID9$	S_4	sD()=[275,375], dA()=[420,821]	Td_1	SUT_2
$ID9^*$	S_4	sD()=[275,375], dA()=[420,821]	Td_1	SUT_4
$ID10$	S_1	sD()=[275,375]	Td_2	SUT_1
$ID10^*$	S_1	sD()=[275,375]	Td_2	SUT_3
$ID11$	S_2	dA()=[420,821]	Td_2	SUT_1
$ID11^*$	S_2	dA()=[420,821]	Td_2	SUT_3

```
1  exfunction
2      //Judge the existence of the vehicle.
           True if it exists.
3      VehicleExists():bool
4      //Calculate the bounding box that
           surrounds the vehicle.
5      Vehicle():bb
6      //Calculate the interval that
           represents the range to be stopped
           .
7      directionAreaDistance():interval
8  endexfunction
9
10 precondition
11     [VehicleExists() = true]
12 endprecondition
13
14 case stop
15     let Vehicle : bb = lVehicle(),
16     directionAreaDistance : interval =
           directionAreaDistance() in
17         PROJ_x(Vehicle) ≈
               directionAreaDistance
18 endcase
19
20 case NOT stop
21     let Vehicle : bb = Vehicle(),
22     directionAreaDistance : interval =
           directionAreaDistance() in
23         not(PROJ_x(Vehicle) ≈
               directionAreaDistance)
24 endcase
```

図 5.9: 他車の x 軸方向の位置関係で停止条件を定義した仕様 S_2

```
1  exfunction
2      vehicleExists():bool
3      vehicle():bb
4      directionAreaDistance():interval
5      stoppingDistance():interval
6  endexfunction
7
8  precondition
9      [vehicleExists() = true]
10 endprecondition
11
12 case x_ystop
13     let vehicle : bb = vehicle(),
14     directionAreaDistance : interval =
           directionAreaDistance(),
15     stoppingDistance : interval =
           stoppingDistance() in
16         PROJ_x(vehicle) ≈
               directionAreaDistance
17         and PROJ_y(vehicle) ≈
               stoppingDistance
18 endcase
19
20 case ysafe_xwarning
21     let directionAreaDistance : interval =
           directionAreaDistance(),
22     stoppingDistance : interval =
           stoppingDistance() in
23         PROJ_x(vehicle) ≈
               directionAreaDistance
24         and not(PROJ_y(vehicle) ≈
               stoppingDistance)
25 endcase
26
27 case xsafe_ywarning
28     let directionAreaDistance : interval =
           directionAreaDistance(),
29     stoppingDistance : interval =
           stoppingDistance() in
30         not(PROJ_x(vehicle) ≈
               directionAreaDistance)
31         and PROJ_y(vehicle) ≈
               stoppingDistance
32 endcase
33
34 case NOT warning
35     let vehicle : bb = vehicle(),
36     directionAreaDistance : interval =
           directionAreaDistance() in
37         not(PROJ_x(vehicle) ≈
               directionAreaDistance) and
38         not (PROJ_y(vehicle) ≈
               stoppingDistance)
39 endcase
```

図 5.10: 仕様 $S3$ のケースを分割して 4 つのケースで構成される仕様 S_4

表 5.8: 表 5.7 に示した 22 種類のテスト結果

Test ID	(T/T+F) of $IoU_{0.6}$	(T/T+F) of $IoU_{0.8}$	(T/T+F) in BBSL
$ID1$	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2527/(2527+209) = 92.4\%$
$ID1^*$	$2041/(2041+695) = 74.6\%$	$1668/(1668+1068) = 61.0\%$	$2429/(2429+307) = 88.8\%$
$ID2$	$1221/(1221+1515) = 44.6\%$	$791/(791+1945) = 28.9\%$	$1929/(1929+807) = 70.5\%$
$ID2^*$	$1450/(1450+1286) = 53.0\%$	$921/(921+1815) = 33.7\%$	$2312/(2312+424) = 84.5\%$
$ID3$	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2500/(2500+236) = 91.4\%$
$ID3^*$	$2041/(2041+695) = 74.6\%$	$1668/(1668+1068) = 61.0\%$	$2414/(2414+322) = 88.2\%$
$ID4$	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2524/(2524+212) = 92.3\%$
$ID4^*$	$2041/(2041+695) = 74.6\%$	$1668/(1668+1068) = 61.0\%$	$2444/(2444+292) = 89.3\%$
$ID5$	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2591/(2591+145) = 94.7\%$
$ID5^*$	$2041/(2041+695) = 74.6\%$	$1668/(1668+1068) = 61.0\%$	$2565/(2565+171) = 93.8\%$
$ID6$	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2604/(2604+132) = 95.2\%$
$ID6^*$	$2041/(2041+695) = 74.6\%$	$1668/(1668+1068) = 61.0\%$	$2520/(2520+216) = 92.1\%$
$ID7$	$2180/(2180+556) = 79.7\%$	$1432/(1432+1304) = 52.3\%$	$2502/(2502+234) = 91.4\%$
$ID7^*$	$2041/(2041+695) = 74.6\%$	$1668/(1668+1068) = 61.0\%$	$2417/(2417+319) = 88.3\%$
$ID8$	$1221/(1221+1515) = 44.6\%$	$791/(791+1945) = 28.9\%$	$2065/(2065+671) = 75.5\%$
$ID8^*$	$1450/(1450+1286) = 53.0\%$	$921/(921+1815) = 33.7\%$	$2406/(2406+330) = 87.9\%$
$ID9$	$1221/(1221+1515) = 44.6\%$	$791/(791+1945) = 28.9\%$	$1867/(1867+869) = 68.2\%$
$ID9^*$	$1450/(1450+1286) = 53.0\%$	$921/(921+1815) = 33.7\%$	$2256/(2256+480) = 82.5\%$
$ID10$	$4842/(4842+802) = 85.8\%$	$3182/(3182+2462) = 56.4\%$	$5299/(5299+345) = 93.9\%$
$ID10^*$	$4565/(4565+1079) = 80.9\%$	$3722/(3722+1922) = 65.9\%$	$5199/(5199+445) = 92.1\%$
$ID11$	$4842/(4842+802) = 85.8\%$	$3182/(3182+2462) = 56.4\%$	$5314/(5314+330) = 94.2\%$
$ID11^*$	$4565/(4565+1079) = 80.9\%$	$3722/(3722+1922) = 65.9\%$	$5183/(5183+461) = 91.8\%$

5.4 BBSL 仕様ベースカバレッジの比較

ここでは，既存のデータセット上で，2 種類の仕様を用意して，提案した 5 つの BBSL 仕様ベースカバレッジ BC_d，BC_c，BC_{cd}，BC_{mcd}，BC_{mc} の測定を行う．初めに，2 種類の仕様 S_3 と S_5 を準備する．仕様 S_3 は，既に図 4.5 で示した仕様と同様の比較的シンプルな仕様であり，仕様 S_5 は，この測定のために新たに与える比較的大きな仕様で多くの条件とケースを含む．図 5.12 に仕様 S_5 を示す．この仕様は，画像上に車両が存在するかを判定する vehicleExists と対象車両のバウンディングボックスを返す vehicle の他に 3 つのバウンディングボックス (leftZone，rightZone，directionArea) と 2 つの区間 (decelerationDistance, stoppingDistance) を返す外部関数が宣言されている．これらのバウンディングボックスと区間を画像上に図示したものを図 5.11 に示す．rightZone は右に車線変更するための充分なスペースを表すバウンディングボックスで，leftZone は左に車線変更するための充分なスペースを表すバウンディングボックス，そして，directionArea は自車の領域をあらわすバウンディングボックスとして導入している．また，stoppingDistance は停止する必要のある車間距離をあらわす区間，decelerationDistance は減速の必要のある車間距離をあらわす区間として導入している，仕様 S_5 では，このようなバウンディングボックスと区間を用いて対象車両の位置関係を記述し,「停止すべきケース」,「減速，右に車線変更，左に車線変更のいずれかがとれるケース」,「減速，左に車線変更のどちらかがとれる (右に車線変更できない) ケース」,「減速，右に車線変更のどちらかがとれる（左に車線変更できない) ケース)」,「減速のみができるケース」,「右への車線変更のみができるケース」,「左への車線変更のみできるケース」,「左右への車線変更のみできる (減速はできない) ケース」の 8 種類のケースを定義している．これらの仕様 S_3 と S_5 において，5 つの BBSL 仕様ベースカバレッジ BC_d，BC_c，BC_{cd}，BC_{mcd}，BC_{mc} の分母を示したものを表 5.9 に示す．仕様 S_3 における BBSL 仕様ベースカバレッジの分母に関しては節 4.3 で例として既に示しているのでここで詳細は割愛する．仕様 S_5 は，8 種類のケースを定義した仕様であるため，$BC_d(\_, S_5)$ の分母はケース数と同じ 8 となる．また，リテラルの数は，ケース stop では 23 行目と 25 行目の 2 つ，以降の 7 種類のケースでは，各 5 つずつリテラルがあるため，この仕様のリテラルの総数 L_{S_5} は，$2 + 7 \times 5 = 37$ である．そのため，各リテラルの true,false の数で定義される $BC_c(\_, L_{s_5})$ の分母は 37× 2=74 となる．さらに，$BC_d(\_, S_5)$ の分母と $BC_c(\_, L_{s_5})$ の分母の和で定義される $BC_{cd}(\_, S_5, L_{s_5})$ の分母は，$8 + 74 = 82$ となる．ここで，仕様 S_5 における条件がどのように構成されているかを見ると．S_5 の各ケースは，図 5.11 に示された各バウンディングボックスと区間 (合計 5 種) に対して他車両が重なっているか重なっていないかの組み合わせで定義されていることが分かる．したがって，同値なものを除いたリテラルの数 $|Q_{s_5}|$ は 5 である．この 5 つのリテラルに論理的に割り当てれる真理値の組み合わせ $|B_{s_5}|$ は，2 の 8 乗の 32 に対して，12 行目から 18 行目にある前提条件の制約の影響で，理論的に存在しない 4 つを除いて 28 となる．具体的には，leftZone と rightZone と重なり，directionArea には重ならない位置は，前提条件よりとることができない．leftZone と rightZone と重なり，directionArea には重ならない位置の時，残りの stoppingDistance と decelerationDistance 重なるかどうかの 4 種類の組み合わせが存在するため，$BC_{mc}(\_, Q_{s_5})$ の分母は $32 - 4 = 28$ となる．さらに，仕様 S_5 におけるセンシティブな条件の組み合わせをすべて数え上げた結果，総数は 81 であるため，$BC_{mcd}(\_, P_{s_5}, V_{s_5})$ の分母は 81 となる．

この測定では 5.3 節で用いた Table 5.6 上の KITTI のデータセット $Td1$,$Td2$ をベースにして，8 種類のデータセットを作成する．仕様 S_3 と共に用いる 4 種類のデータセット Td_{1a}, Td_{1b}, Td_{1c},Td_{1d} は，$Td1$ の非常に小さなサブセットであり，それぞれのテストケース数は 7,5,8,4 となる．対して，仕様 S_3 と共に用いる 4 種類のデータセット Td_1, $(Td_1 + Td_2)$, $(Td_1 + Td_2)^*$,$(Td_1 + Td_2)^{**}$ は比較的テストケースが多く，それぞれのテストケース数は，

2736,8380,25140,75420 となる．データセット Td_1 は，5.3 節で用いた Td_1 と同じデータセットである．データセット $(Td_1 + Td_2)$ は，Td_1 の後ろに Td_2 を加えたデータセットである．そして，データセット $(Td_1 + Td_2)^*$ と $(Td_1 + Td_2)^{**}$ は，データセット $(Td_1 + Td_2)$ と同様の画像セットであるが，仕様 S_5 に対して，与える外部関数の値を変えることでテストケース数をかさましさせたデータセットとなる．

以上の仕様とデータセットを用いて，BBSL 仕様ベースカバレッジを測定した結果を表 5.10 に示す．Td_{1a} は，BC_c は 100%だが，BC_d が 100%でないデータセットであることが分かる．また，Td_{1b} は，BC_{mcd} は 100%だが，BC_{mc} が 100%でないデータセットであることが分かる．データセット Td_{1d} に対する結果が示すように，仕様 S_3 では，わずか 4 つのテストケースで全ての BBSL 仕様ベースカバレッジで 100%を達成させることができる．一方で，8 つのケースを含む仕様 S_5 では，25140 のテストケースを持つデータセット $(Td_1 + Td_2)^*$ において，BC_d でさえ 100%にはならない．75420 のテストケースを持つデータセット $(Td_1 + Td_2)^{**}$ では，BC_d は 100%となっているが，それでも，BC_{mcd} や BC_{mc} は 100%にならない．これらの結果は，提案された BBSL 仕様カバレッジの差異を示すとともに，テストケースを手当たり次第に増加させても，必ずしもこれらのカバレッジが増加するわけではないことが分かる．

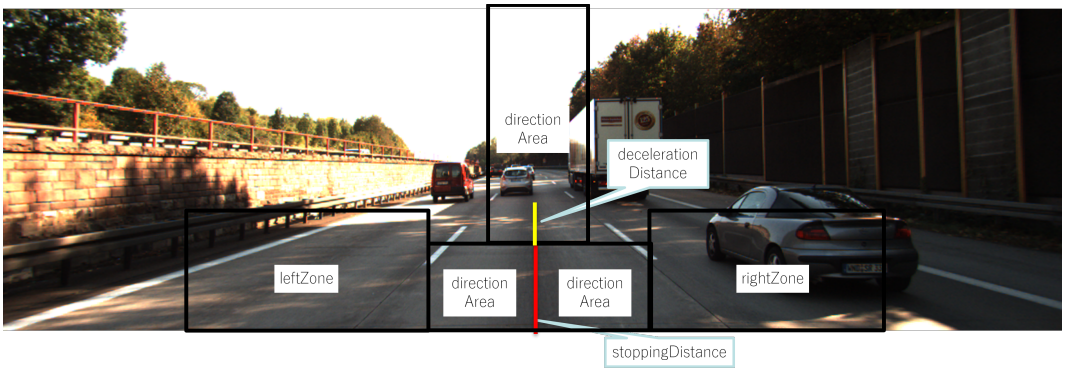

図 5.11: 仕様 S_5 で記述される他車の位置関係

表 5.9: 仕様 S_3，S_5 に対する BBSL 仕様ベースカバレッジの分母

仕様	BC_d	BC_c	BC_{cd}	BC_{mcd}	BC_{mc}
S_3	/2	/8	/10	/6	/4
S_5	/8	/74	/82	/81	/28

```
exfunction
   vehicleExists():bool
   vehicle():bb
   stoppingDistance():interval
   directionArea():setBB
   decelerationDistance():interval
   leftZone():bb
   rightZone():bb
endexfunction

precondition
   [vehicleExists() = true
         and PROJ_ȳ(decelerationDistance()) > PROJ_ȳ(leftZone())
         and PROJ_ȳ(decelerationDistance()) > PROJ_ȳ(rjghtZone())
         and PROJ_y(stoppingDistance()) = PROJ_y(leftZone())
         and PROJ_y(stoppingDistance()) = PROJ_y(rjghtZone())
         and PROJ_ȳ(stoppingDistance()) = PROJ_y(decelerationDistance())
endprecondition

case stop
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() in
         PROJ_y(vehicle) ≈ stoppingDistance
         and exists x ∈ directionArea.(vehicle ≈ x)
endcase

case deceleration,front−right,front−left
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         not(PROJ_y(vehicle) ≈ stoppingDistance)
         and exists x ∈ directionArea.(vehicle ≈ x)
         and PROJ_y(vehicle) ≈ decelerationDistance
         and not(vehicle≈leftZone)
         and not(vehicle≈rightZone)
endcase

case deceleration,front−left
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         not(PROJ_y(vehicle) ≈ stoppingDistance)
         and PROJ_y(vehicle) ≈ decelerationDistance
         and exists x ∈ directionArea.(vehicle ≈ x)
         and not(vehicle≈leftZone)
         and vehicle≈rightZone
endcase

case deceleration,front−right
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         not(PROJ_y(vehicle) ≈ stoppingDistance)
         and PROJ_y(vehicle) ≈ decelerationDistance
         and exists x ∈ directionArea.(vehicle ≈ x)
         and vehicle≈leftZone
         and not(vehicle≈rightZone)
endcase


case deceleration
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         not(PROJ_y(vehicle) ≈ stoppingDistance)
         and PROJ_y(vehicle) ≈ decelerationDistance
         and exists x ∈ directionArea.(vehicle ≈ x)
         and vehicle≈leftZon
         and vehicle≈rightZone)
endcase

case front−right
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         ((not(PROJ_y(vehicle) ≈ stoppingDistance)
         and not(PROJ_y(vehicle) ≈ decelerationDistance))
         or not(exists x ∈ directionArea.(vehicle ≈ x)))
         and vehicle≈leftZone
         and not(vehicle≈rightZone)
endcase

case front−left
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         ((not(PROJ_y(vehicle) ≈ stoppingDistance)
         and not(PROJ_y(vehicle) ≈ decelerationDistance))
         or not(exists x ∈ directionArea.(vehicle ≈ x)))
         and not(vehicle≈leftZone)
         and vehicle≈rightZone
endcase

case front−right, front−left
   let vehicle : bb = vehicle(),
      stoppingDistance : interval = stoppingDistance(),
      directionArea : setBB = directionArea() ,
      decelerationDistance : interval = decelerationDistance(),
      rightZone:bb = rightZone(),
      leftZone:bb = leftZone:bb in
         ((not(PROJ_y(vehicle) ≈ stoppingDistance)
         and not(PROJ_y(vehicle) ≈ decelerationDistance))
         or not(exists x ∈ directionArea.(vehicle ≈ x)))
         and not(vehicle≈leftZone)
         and not(vehicle≈rightZone)
endcase
```

図 5.12: 他車の位置によって定義される自車の 8 ケースの応答の仕様 S_5

表 5.10: BBSL 仕様ベースカバレッジの測定結果

仕様	テストデータセット	BC_d	BC_c	BC_{cd}	BC_{mcd}	BC_{mc}
S_3	Td_{1a}	1/2	8/8	9/10	4/6	3/4
S_3	Td_{1b}	2/2	8/8	10/10	6/6	3/4
S_3	Td_{1c}	2/2	8/8	10/10	4/6	3/4
S_3	Td_{1d}	2/2	8/8	10/10	6/6	4/4
S_5	Td_1	7/8	74/74	81/82	43/81	13/28
S_5	$(Td_1 + Td_2)$	7/8	74/74	81/82	45/81	14/28
S_5	$(Td_1 + Td_2)^*$	7/8	74/74	81/82	55/81	18/28
S_5	$(Td_1 + Td_2)^{**}$	8/8	74/74	82/82	58/81	19/28

5.5　既存のデータセットにおける空間カバレッジの測定

ここでは，既存のデータセット上で，提案した 2 つの空間カバレッジ SC_{pos} と SC_{siz} の測定を行う．5.3 節で用いた Table 5.6 上の KITTI のデータセット $Td1$,$Td2$ をベースにして，5 種類のデータセットを作成する．1 つ目のデータセット Td_1 と 2 つ目のデータセット Td_2 は，5.3 節で用いた Td_1，$Td2$ と同じデータセットである．3 つ目のデータセット $(Td_1 + Td_2)$ は，Td_1 の後ろに Td_2 を加えたデータセットである．4 つ目のデータセット Td_1^* は Td_1 から車両オブジェクトの数が 20000 個になるまで無作為に復元抽出して作成したデータセットであり，5 つ目のデータセット Td_2^* も同様に，Td_2 から車両オブジェクトの数が 20000 個になるまで無作為に復元抽出して作成したデータセットである．

次に，空間カバレッジ SC_{pos} と SC_{siz} のために必要な位置クラス R とサイズクラス S を準備する．この測定で用いる位置クラス R は，図 5.13 に示すように画像全体を隙間なくうめるバウンディングボックスの集合 $\{r1, r2, r3, r4, ...\}$ で与える．ここでは，画像の左上座標を原点 (0,0) とし，x 軸方向も y 軸方向も共に均等に 100 分割した合計 10000 個のバウンディングボックスの集合で位置クラス R を与える．一方，サイズクラス S は，$S = \{46.575x | 0 \leq x \leq 100000\}$ で与えた．ここで，実数 46.57 とは，画像全体のサイズ 1242×375=465750 を 100,00 で割った数に相当する．つまり，この測定で用いる空間カバレッジ SC_{pos} と SC_{siz} はどちらとも分母 10,000 で設定している．

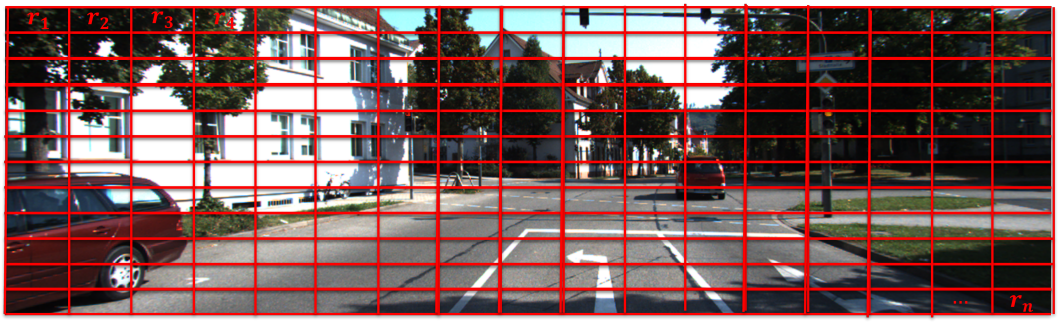

図 5.13: 測定に用いた位置クラス R のイメージ

データセット Td_1 における 2736 台の車両オブジェクトのグラウンドトゥルースに基づくバウンディングボックスの集合を $T1$ とする．図 5.14 の左に示すのが $SC_{pos}(T_1, R)$ の変移で，右に示すのが $SC_{siz}(T_1, S)$ の変移である．最終的な $SC_{pos}(T_1, R)$ の値は，6.15%であり，$SC_{siz}(T_1, S)$ の値は，6.44%となった．さらに，このデータセットにおいて，$SC_{pos}(T_1, R)$ に変化のないテストケースは連続で最大 128 件続き，$SC_{siz}(T_1, S)$ に変化のないテストケースは最大 231 件存在した．言い換えると，仮にこのテストケースを SC_{pos}，SC_{siz} について，飽和していないとみなすために必要なウィンドウサイズはそれぞれ少なくとも 129 以上，232 以上で設定する必要がある．

次に，データセット Td_2 における 5644 台の車両オブジェクトのグラウンドトゥルースに基づくバウンディングボックスの集合を $T2$ とする．図 5.15 の左に示すのが $SC_{pos}(T_2, R)$ の変移で，右に示すのが $SC_{siz}(T_2, S)$ の変移である．最終的な $SC_{pos}(T_2, R)$ の値は，10.11%であり，$SC_{siz}(T_2, S)$ の値は，7.89%となった．さらに，このデータセットにおいて，$SC_{pos}(T_2, R)$ に変化のないテストケースは連続で最大 194 件続き，$SC_{siz}(T_2, S)$ に変化のないテストケースは最大 371 件存在した．言い換えると，仮にこのテストケースを SC_{pos}，SC_{siz} について，飽和していないとみなすために必要なウィンドウサイズはそれぞれ少なくとも 195 以上，272 以上で設定する必要がある．

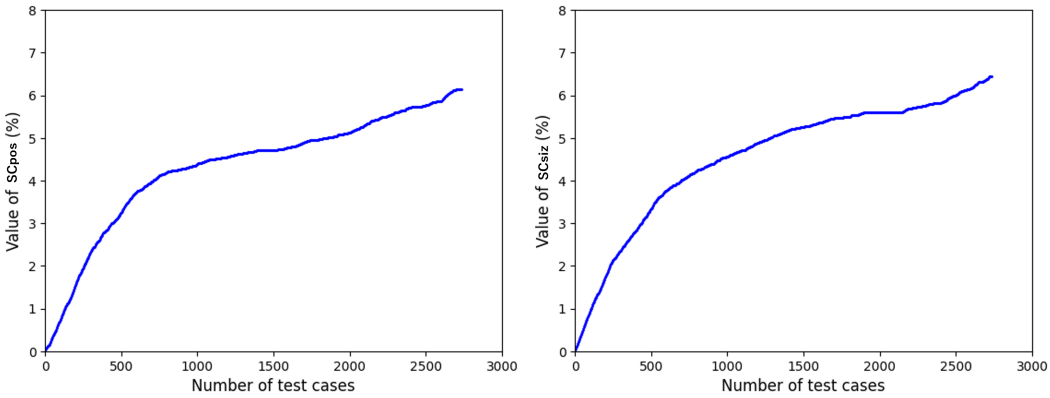

図 5.14: データセット Td_1 上の各空間カバレッジの変移

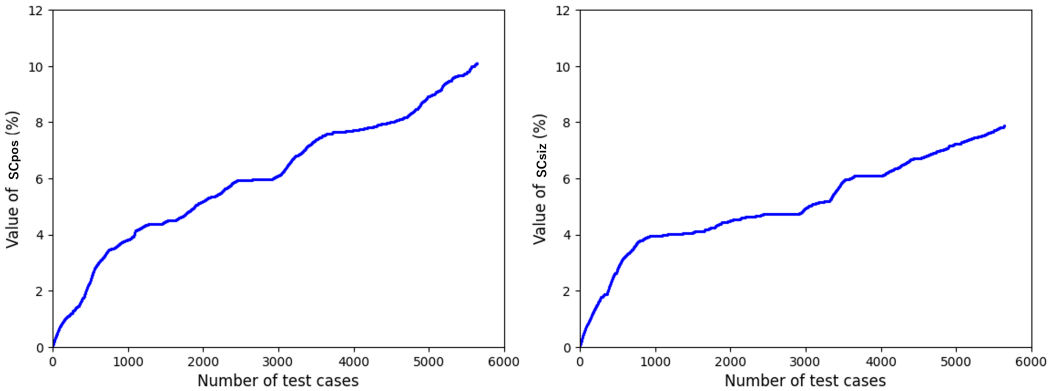

図 5.15: データセット Td_2 上の各空間カバレッジの変移

次に，データセット $(Td_1 + Td_2)$ における 8380 台の車両オブジェクトのグラウンドトゥルースに基づくバウンディングボックスの集合を $T3$ とする．図 5.16 の左に示すのが $SC_{pos}(T_3, R)$ の変移で，右に示すのが $SC_{siz}(T_3, S)$ の変移である．最終的な $SC_{pos}(T_3, R)$ の値は，11.74%であり，$SC_{siz}(T_3, S)$ の値は，9.81%となった．さらに，このデータセットにおいて，$SC_{pos}(T_3, R)$ に変化のないテストケースは連続で最大 361 件続き，$SC_{siz}(T_3, S)$ に変化のないテストケースは最大 841 件存在した．言い換えると，仮にこのテストケースを SC_{pos}，SC_{siz} について，飽和していないとみなすために必要なウィンドウサイズはそれぞれ少なくとも 362 以上，842 以上で設定する必要がある．

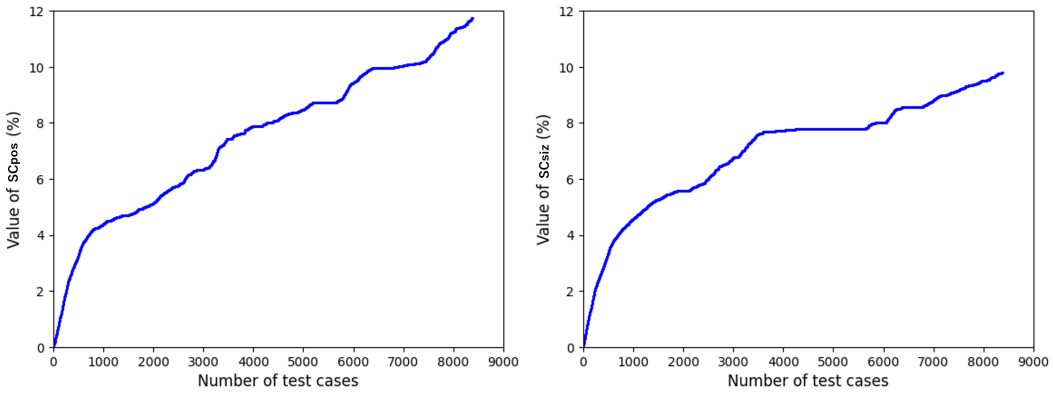

図 5.16: データセット $(Td_1 + Td_2)$ 上の各空間カバレッジの変移

さらに，データセット Td_1^* における 20,000 台の車両オブジェクトのグラウンドトゥルースに基づくバウンディングボックスの集合を $T4$ とする．図 5.17 の左に示すのが $SC_{pos}(T_4, R)$ の変移で，右に示すのが $SC_{siz}(T_4, S)$ の変移である．最終的な $SC_{pos}(T_4, R)$ の値は，6.15%であり，$SC_{siz}(T_4, S)$ の値は，6.44%となった．これは，Td_1 に含むすべてのバウンディングボックスの集合 $T1$ の場合と同値であり，実際，図 5.17 では 12,319 番目のテストケースまでで Td_1 のすべてのテストデータを少なくとも 1 回テストできた状態になっており，以降 SC_{pos}，SC_{siz} は横ばいになっている．

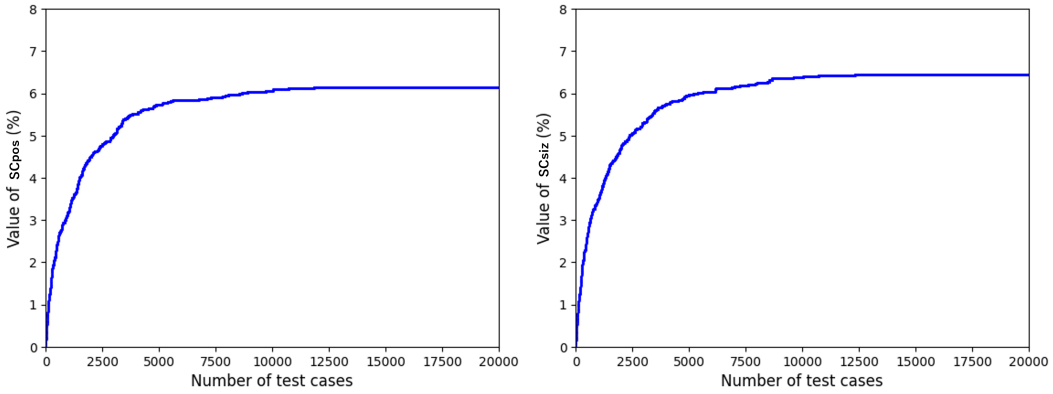

図 5.17: データセット Td_1^* 上の各空間カバレッジの変移

最後に，データセット Td_2^* における 20,000 台の車両オブジェクトのグラウンドトゥルースに基づくバウンディングボックスの集合を $T5$ とする．図 5.18 の左に示すのが $SC_{pos}(T_5, R)$

の変移で，右に示すのが $SC_{siz}(T_5, S)$ の変移である．最終的な $SC_{pos}(T_5, R)$ の値は，10.05% であり，$SC_{siz}(T_5, S)$ の値は，7.83%となった．Td_1^* の場合と異なり，Td_2^* の母集団である Td_2 の全てのデータがテストされたわけではないため，$SC_{pos}(T_5, R)$ と $SC_{pos}(T_5, R)$ の最終的な値は，Td_2 に対する値よりも低くなっている．

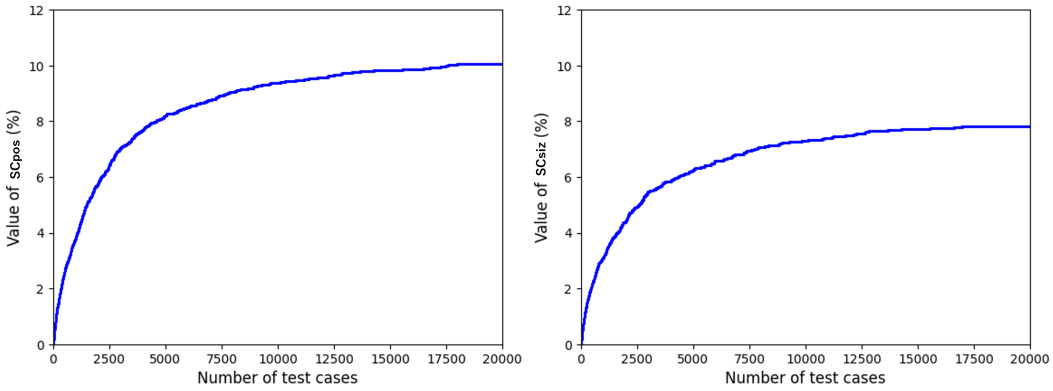

図 5.18: データセット Td_2^* 上の各空間カバレッジの変移

以上のデータセットにおける SC_{pos} と SC_{siz} の測定により，これらの空間カバレッジがそれぞれ異なるテストクライテリアとして機能していることがわかる．また，図 5.14，図 5.15，図 5.16 では，両方の空間カバレッジが非常に偏った曲線を示しており，飽和判定が困難であるのに対し，図 5.17，図 5.18 では，どちらの空間カバレッジも概ね飽和曲線を示していることが分かる．このことから，データセットによっては空間カバレッジで飽和判定をおこなうことができるが，既存のテストデータセットに対して，汎用的にできるわけではなく，データセットに対して何かしらの制限が必要であることがわかる．

第6章　考察

6.1　BBSLの表現力

ここでは，5.1 節と 5.2 節で BBSL を評価した結果を用いて，BBSL が自動運転システムの OEDR 仕様の記述に特に適している部分を考察する．本研究では，車両や歩行者などのオブジェクトと停止区間などの特定領域の相対的な位置関係によって，自動運転システムの機能を定義できると推測し，OEDR の仕様を画像上の位置関係で形式的に記述するための BBSL を提案した．5.1 節では，実際に書くべきイベントや書き分けるべき自動運転システムののレスポンスを表 5.1 を参照して決めた．結果として，表 5.2 に示すように，画像上の特定領域に対するオブジェクトの相対的な位置関係による条件のみを用いて，全てのレスポンスを書き分けて記述できた．しかし，BBSL がカバーする範囲は画像上の相対位置関係に限定されてるため，BBSL が画像から判断できる機能仕様全体を包括的に記述できているわけではないことも明らかである．例えば，図 6.1 の画像は，道路上に絵として描かれた歩行者のトリックアートである [7]．人間は画像情報のみからこの画像の歩行者が絵であることが分かり，停止する必要がないことを判断することができるが，この場合，位置によって判断しているのではく，BBSL では抽象化した形状や色などを含む質感により判断している．このように，位置以外の画像情報によって定義される自動運転システムの機能は存在するため，BBSL は画像情報で自動運転システムの機能を包括的に記述できるわけではないが，画像上のオブジェクトの相対的な位置を用いて自動運転システムの機能を書き分けることが可能である．

さらに，例として表 5.2 の中から"lead vehicle cutting out"イベントにおける反応を実際に記述した仕様を図 6.2 に示す．この仕様は，加速，減速，停止，特に反応の必要のない状況の 4 種類に関して，それぞれ"accelerate"，"decelerate"，"stop"，"not respond"というケースで定義している．図 6.2 の 29 行目や 37 行目に記述した特定の方向への位置関係は，5.2 節の表 5.3 で示したように，IoU で表現することはできない．一方で，図 6.2 の 20 行目や 28 行目に記述した，どの自車の進行方向に侵入しているかという割合を用いた位置関係に関しては，5.2 節の表 5.4 で示したように，BTR で表現することはできない．このような車両がある領域にどの程度侵入しているかや，どの方向に位置しているかといった要素は，自動運転システムの反応を決める際に重要な要素となる．したがって，自動運転システムの OEDR の仕様を画像ベースに記述するために，これら両方の側面を記述できる BBSL の表現力は適していると言える．

BBSL で記述された仕様に関するもう 1 つの特筆すべき利点は，前提条件を満たすあらゆる走行環境についてどう反応すべきか網羅的に記述することが可能であるという点である．例えば，図 6.2 の仕様では，画像上に車両が存在している限り，どんな画像であってもどのケースにあてはまるか記述されている．5.2 節の表 5.5 に示したように，幾何学的な記述でこのように網羅的な仕様を実現させることは非常に困難である．そして，このようにあらゆる画像に対してケースつまり機能を定義できる仕様であることは，任意の入力画像に対する正しい応答の決定を容易にするため，自動運転システムの画像認識を機能テストする場合に重要な要素となる．

図 6.1: 道路上に歩行者が描かれたトリックアート [7]

結論として,BBSL の 1 つ目の利点は，位置関係を記述する際の表現力が IoU や BTR を上回っていることである．BBSL は，位置と大きさに基づく関係の両方を含む複雑な空間関係を記述できる．そのため，自動運転環境での物体の動作と動きを記述するために重要な画像の情報を詳細かつ正確に表現することを可能とする．BBSL の 2 つ目の利点は，画像認識タスクの仕様の定義に適している点である．BBSL は，自動運転システムのシナリオ毎に抽象化された位置関係の記述をおこなえるため，幅広いシナリオをカバーでき，幾何学的手法のみを使用する場合よりも柔軟性と包括性が向上している．ただし，BBSL は，主に自動運転システムの物体検出モジュール内の画像認識システムに焦点を当てているため，特に 2 次元のバウンディングボックスとそれらの位置関係のみを扱っている．これは，画像認識に特化して設計されたため，画像認識タスクには適用できたが，実際の自動運転システムにおいて，認識モジュールは LiDAR センサーを使用して 3D オブジェクトを処理し，これらのオブジェクトの軌跡をも考慮して計算される機構を含んでいる [49]．自動運転システム全体のコンテキストと DNN を用いたサブモジュールの実装の乖離は画像認識のみの問題点ではないため，BBSL が画像認識のみであるという適用範囲の狭さは，現状の欠点といる．このような高度な物体検出モジュール全体を考慮した仕様の記述には，BBSL を拡張して 3 次元のバウンディングボックスと軌跡情報に関する情報を記述できる必要がある．このように，現状の BBSL は画像認識システムのみを対象としているが，今後 BBSL を拡張すると，より複雑で現実的な自動運転シナリオへの適用性が大幅に向上し，多様なセンサー入力から処理される動的なオブジェクトの検出に対しても機能テストを実施できることが期待できる．

```
exfunction
    //Judge the existence of the lead vehicle. true if it exitst.
    leadVehicleExists():bool
    //Return the bounding box that surrounds the lead vehicle.
    leadVehicle():bb
    //Return the bounding box that represents the range to reduce speed.
    deceleratingArea():bb
    //Return a set of bounding boxes covering the range of traveling lane.
    travelingLane():setBB
endexfunction

precondition
    [leadVehicleExists() = true]
endprecondition

case decelerate
    let leadVehicle : bb = leadVehicle(),
    deceleratingArea : bb = deceleratingArea(),
    travelingLane : setBB = travelingLane() in
        RAT(travelingLane∩{leadVehicle}, {leadVehicle})> 0.3 and
        PROJ_y(leadVehicle) ≈PROJ_y(deceleratingArea)
endcase

case accelerate
    let leadVehicle : bb = leadVehicle(),
    deceleratingArea : bb = deceleratingArea(),
    travelingLane : setBB = travelingLane() in
        RAT(travelingLane∩{leadVehicle}, {leadVehicle})> 0.3 and
        PROJ_y(leadVehicle) >PROJ_y(deceleratingArea)
endcase

case stop
    let leadVehicle : bb = leadVehicle(),
    deceleratingArea : bb = deceleratingArea(),
    travelingLane : setBB = travelingLane() in
        RAT(travelingLane∩{leadVehicle}, {leadVehicle})> 0.3 and
        PROJ_y(leadVehicle) <PROJ_y(deceleratingArea)
endcase

case NOT respond
    let leadVehicle : bb = leadVehicle(),
    deceleratingArea : bb = deceleratingArea(),
    travelingLane : setBB = travelingLane() in
        not(RAT(travelingLane∩{leadVehicle}, {leadVehicle})> 0.3)
endcase
```

図 6.2: "lead vehicle cutting out"イベントにおける BBSL で記述した OEDR 仕様

6.2 仕様の品質

ここでは，本論文で提案した仕様が，2.5 節で挙げた IEEE 830 における 8 つの品質特性をどの程度考えられているかについて考察する．まず，表 5.2 の中から"Lead vehicle stopped"イベントにおけるリアクションを実際に記述した仕様を図 6.3 に示す．この仕様は，減速，停止，特に反応の必要のない状況の 3 種類に関して，それぞれ"decelerate"，"stop"，"NOT respond"というケースで定義している．画像情報における車両の下端の y 軸方向の位置のみを見ているため，車両の位置に対応する定義されている機能を抽象的な図で表すと，図 6.4 のようになる．よって，leadVehicle が存在する場合，どんな画像であっても何かしらのケースが定義されているため網羅的であり，2 つ以上のリアクションに対応する画像はないため排他的で，どのリアクションにも必ず該当する画像は存在するため非冗長な仕様である．そのため，3.5 節で定義した 3 種類の性質を持たす仕様になっている．つまり，提案手法により，IEEE 830 で定義されている完全性に関しては，非冗長な仕様を満たすことにより意味を持たない記述がないことを保証することで寄与しており，一貫性に関しては，排反的な仕様を満たすことにより，停止する必要があり，かつ，停止する必要のない状況のような矛盾する条件を排除していることで寄与している．さらに，無曖昧性に関しても，排反的な仕様を満たすことにより，各位置関係が停止などの一意的な意味しか持たないことを保証することで寄与しつつ，BBSL という形式仕様記述言語で仕様が作成されていることによっても，無曖昧性の寄与に大きく貢献をしている．追跡可能性に関しては，5.3 節で実施したように，作成した機能仕様上のどの機能要件に基づいたテストケースであるかが対応づいた形式で仕様に基づいてテストすることができる点で貢献している．

一方で，このような自動運転システムの OEDR の仕様上では，各機能の優先度は考慮することがはできるが，現状では，複数の仕様間における仕様レベルの順序付けに関する品質特性について考慮できてはいない．例えば，図 6.3 の仕様には，"stop"，"decelerate"，"NOT respond"に該当するオブジェクトの位置の y 軸方向上の連続性が存在している．つまり，"stop"に該当する車両の下辺より，"decelerate"に該当する車両の下辺の方が常に画像上の上に位置しており，さらに，"NOT respond"に該当する車両の下辺はそれ以外の画像の車両の下辺より画像上の上に位置している．このような連続性は，"stop"，"decelerate"のように機能的に優先度が事なるレスポンスを持つ仕様において存在しており，条件によって BBSL で記述することが可能であるが，IEEE 830 で定義されている複数仕様間におけるどの仕様の要求を優先すべきかを記述する方法について未解決である．

次に，仕様上の網羅性，排他性，非冗長性などを対象として，検証容易性について考察する．BBSL では，型や各演算子について全て定義がされており，さらに，BBSL の仕様そのものも画像情報を受け取りレスポンスを返す関数として定義されている．これらの定義は，集合論などの数学的概念を用いて記述されており，おそらく矛盾なく定義されているが，coq[50] などの定理証明システムのような厳密な形式体系で形式化されているわけではないため，仕様そのものの正当性や性質が厳密に証明されているわけではない．そのため，現状では，形式仕様記述における検査可能性の保証は困難である．実際に，BBSL における区間やバウンディングボックスは実数により定義しており，これらの位置関係の多くは実数の比較により定義されていることから，これらの性質の証明は停止しない可能性がある [51]．ただし，BBSL で記述する区間やバウンティンボックスが画像上の特定の領域を表すことと，実際に使われる画像の多くがラスターファイル，つまり有限個のピクセルから構成されるデータ形式であることを考えると，BBSL で定義する実数を有理数で置き換えても問題ない可能性がある．そのため，実用上は有理数による定義に置き換えることで，決定可能性が保証されると考えられる．

最後に，仕様上の網羅性，排他性，非冗長性の証明プロセスの自動化の可能性についても考察する．coq のような対話的証明では人間が計算機に対して対話的に指示をしながら

証明をするが，その一部の自動化のために，SAT ソルバーや SMT ソルバーを使うことが考えられる [52]．BBSL における区間やバウンディングボックスは実数により定義しており，さらに，定義 3.13 では割り算が使用されるため線型算術でもなく，BBSL で使用される forall や exists などの量化子が定義上無限のバウンディングボックスの集合に対して用いられるため，自動化は非常に困難である．ただし，検査可能性の場合と同様に，画像上が有限個のピクセルから構成されるデータ形式であることから BBSL で定義する実数の一部を整数に置き換えても問題ない可能性がある．実用上は，定義 3.13 で求められる RAT の返り値としての実数以外の実数を整数で置き換えることで，全てのバウンディングボックスの集合は有限となり，理論上，量化子のついた集合は全て数え上げることができ，バウンディングボックスの集合の量化子は全て除去できる可能性がある．このような工夫を行うことによって，自動化の可能性はあると考えられる．

```
1  exfunction
2      //Judge the existence of the lead vehicle. true if it
           exitst.
3      leadVehicleExists():bool
4      //Return the bounding box that surrounds the lead
           vehicle.
5      leadVehicle():bb
6      //Return the bounding box that represents the
           range to reduce speed.
7      deceleratingArea():bb
8  endexfunction
9
10 precondition
11     [leadVehicleExists() = true]
12 endprecondition
13
14 case decelerate
15     let leadVehicle : bb = leadVehicle(),
16     deceleratingArea : bb = deceleratingArea() in
17         PROJ_y(leadVehicle) ≈PROJ_y(
               deceleratingArea)
18 endcase
19
20 case stop
21     let leadVehicle : bb = leadVehicle(),
22     deceleratingArea : bb = deceleratingArea() in
23         PROJ_y(leadVehicle) <PROJ_y(
               deceleratingArea)
24 endcase
25
26 case NOT respond
27     let leadVehicle : bb = leadVehicle(),
28     deceleratingArea : bb = deceleratingArea() in
29         PROJ_y(leadVehicle) >PROJ_y(
               deceleratingArea)
30 endcase
```

図 6.3: "Lead vehicle stopped"イベントにおける BBSL で記述した OEDR 仕様

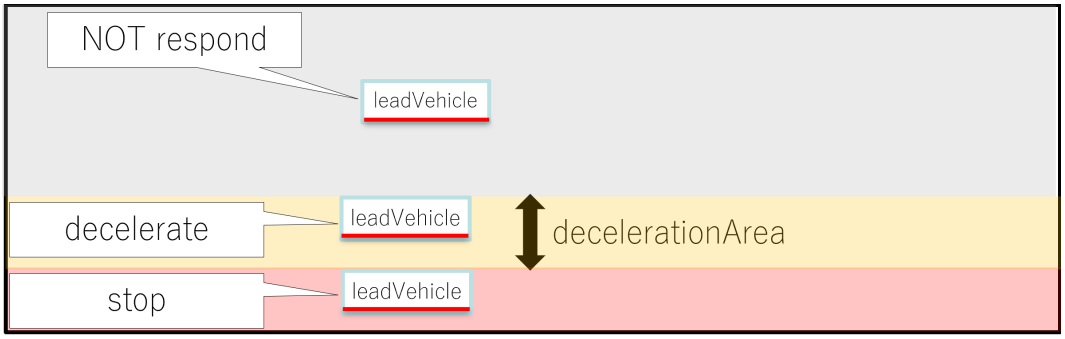

図 6.4: 図 6.3 の仕様で定義される相対位置に対するレスポンスを可視化した画像

6.3 BBSLによる機能テストの有効性

ここでは，5.3節の評価で示した結果を用いて，従来の画像認識のテストとBBSLによって可能になる機能テストとの違いから，新たに評価できるようになった要素を特定し，その有効性を考察する．提案手法は，仕様に基づいて自動運転システムの画像認識をテストし，従来のIoUベースの性能評価では考慮することが難しかった側面を仕様と照らして判定し，自動運転システムの誤動作につながる不具合を検出することを目的としている．特に，IoUによってfalseとして判定されるテストケースは，提案した機能テストと乖離がある可能性がある．従来のIoUベースの評価と提案手法の評価の違いを明確にするために，まず，それぞれの混同行列を定義する．次に，例を示し，これらの行列に反映されている個々のケース間の違いを分析する．

まず，画像認識に用いられる従来の混合行列を定義する．ここでは，2.4節で説明したtrue Positive（TP），False Positive（FP），False Negative（FN），およびTrue Negative（TN）の概念を用いる．車を検出する画像認識において，TPは，グラウンドトゥルースのバウンディングボックスが正しく検出された(IoUが決められた値以上に計算される)場合を示す．一方，FPは，そもそもグラウンドトゥルースのバウンディングボックスが存在しないのに，誤って検出された場合を示す．FNは，グラウンドトゥルースのバウンディングボックスが正しく検出されたなかった(IoUが決められた値未満に計算される)場合を示す．TNについては，バウンディングボックスの存在しない領域を正しく存在しないものとみなせた場合に該当しますが，物体検出の文脈において，任意の画像内には検出されるべきでない領域が無数に存在するため，定義されない．このような定義のもとで，表5.8のテストID1に対応する$IoU_{0.6}$の混合行列を表6.1に示す．TPが2180，FNが556であり，前述したとおりTNは未定義となる．また，このテストにおいては，全ての画像において常に車両のグラウンドトゥルースのバウンディングボックスが存在するため，FPに該当するテストケースも存在しない．

表 6.1: 表5.8のテストID1における$IoU_{0.6}$の混合行列

	Actually Positive	Actually Negative
Predicted Positive	2180	-
Predicted Negative	556	-

次に，表5.8のテストに用いた仕様のうち，仕様S_1，S_2，S_3，つまり，停止すべきケースと停止すべきでないケースが定義されているし仕様も対して，混合行列を定義する．提案テストにおいて，仕様S_4のように機能仕様は2値であるとは限らないため，一般に混合行列は定義できない．ここでは，仕様S_1，S_2，S_3において自動運転システムが正しく停止できるかを検出対象とし，従来の画像認識と同様にTP,FP,TN,FNとして定める．従って，グラウンドトゥルースのバウンディングボックスが仕様上のケース"stop"に該当し，予測結果のバウンディングボックスもケース"stop"に該当する場合をTPとする．一方，グラウンドトゥルースのバウンディングボックスが仕様上のケース"stop"に該当するが，予測結果のバウンディングボックスがケース"NOT stop"に該当する場合をFPとする．さらに，グラウンドトゥルースのバウンディングボックスが仕様上のケース"NOT stop"に該当し，予測結果のバウンディングボックスもケース"NOT stop"に該当する場合をTNとし，グラウンドトゥルースのバウンディングボックスが仕様上のケース"NOT stop"に該当するが，予測結果のバウンディングボックスがケース"stop"に該当する場合をFNとする．このような定義のもとで，表5.8のテストID1に対応する提案テストの混合行列を表6.1に示す．TPが438，FPが131，TNが78，FNが2089となる．以上のテストID1に

おける 2 つの混合行列表 6.1 と表 6.1 を組み合わせてまとめたものを表 6.3 に示す．このようにテスト ID1 のテストケースを 2 つの混同行列により①から⑧に分類した．①から⑧の右側に示す数値が該当するテストケース数である．

表 6.2: 表 5.8 のテスト ID1 における提案テストの混合行列

	Actually Positive	Actually Negative
Predicted Positive	438	131
Predicted Negative	78	2089

表 6.3: 表 5.8 のテスト ID1 における提案テストの混合行列と $IoU_{0.6}$ の混同行列の比較

		BBSL を用いる提案テスト			
		TP	TN	FP	FN
$IoU_{0.6}$	TP	①404	②1744	⑤4	⑥28
	FN	③34	④345	⑦74	⑧103

ここから，表 6.3 に示したテスト ID1 における 8 種類のテストケースの分類に焦点を当てて，例に基づき提案テストの有効性を論じる．まず，表 6.3 の①に該当するテストケースを取り上げる．図 6.5 に示す画像の赤い車両が提案テストと $IoU_{0.6}$ の両方とも TP となる①に該当するテストケースの 1 つであり，推論結果のバウンディングボックスも示されている．この赤い車両は自車に非常に近いため，図 3.7 の仕様 S_1 を基に期待値はケース"stop"となる．推論結果のバウンディングボックスは，x 軸方向に多少のズレが見られるが，赤い車両に対して僅かなズレであるため，$IoU_{0.6}$ において TP である．一方，x 軸方向のズレはあるが，y 軸方向，つまり，前方車との車間距離に関してはほとんどズレがないため，検出結果もケース"stop"に該当し，仕様に基づいて違反していないことが分かる．従って，提案テストが TP となることは妥当であることが分かる．

図 6.5: 表 6.3 の①に該当するテストケースの例

次に，表 6.3 の②に該当するテストケースを取り上げる．図 6.6 に示す画像の白い車両が提案テストでは TN となり $IoU_{0.6}$ では TP となる①に該当するテストケースの 1 つであり，推論結果のバウンディングボックスも示されている．この白い車両は自車から充分に離れた位置にあるため，図 3.7 の仕様 S_1 を基に期待値はケース"NOT stop"となる．推論

結果のバウンディングボックスは，ほとんどズレなく検出いるため，$IoU_{0.6}$ において TP である．当然，このようにほとんどズレがない場合は，仕様と照らし合わせても検出結果はケース"stop"に該当し，仕様に違反していないことが分かる．このように，ズレなく検出できた場合は，$IoU_{0.6}$ でも仕様ベースでも正しい検出であるものとして判定される．

図 6.6: 表 6.3 の②に該当するテストケースの例

次に，表 6.3 の③に該当するテストケースを取り上げる．図 6.7 に示す画像の赤いバウンディングボックスで囲まれた黒い車両が提案テストでは TP となり $IoU_{0.6}$ では FN となる③に該当するテストケースの 1 つであり，推論結果のバウンディングボックスは紫色で示されている．この黒い車両は自車に非常に近いため，図 3.7 の仕様 S_1 を基に期待値はケース"stop"となる．グラウンドトゥルースではこの車両の実際の高さを考慮して赤いバウンディングボックスが与えられるが，画像上では前方部しか写っていないため，推論できているの実際の車の高さよりはるかに低くなっている．そのため，$IoU_{0.6}$ は非常に低くなり，FN となっている．一方で，この車両の高さがズレて検出されていても，この車両との距離間はほとんどズレなく検出できているため，推論結果を仕様と照らし合わせてもケース"stop"に該当する．従って，提案テストが TP となることは妥当であることが分かる．

図 6.7: 表 6.3 の③に該当するテストケースの例

次に，表 6.3 の④に該当するテストケースを取り上げる．図 6.8 に示す画像の白い車両と建物に隠れている赤いバウンディングボックスで囲まれた車両が提案テストでは TN となり $IoU_{0.6}$ では FN となる④に該当するテストケースの 1 つであり，推論結果のバウンディングボックスは下の画像に示されている．この車両は自車から充分に離れた位置にあるため，図 3.7 の仕様 S_1 を基に期待値はケース"NOT stop"となる．この画像を見ると，画像認識は対象の車両を認識できずに，手前の車両を認識してしまっている．当然，$IoU_{0.6}$ は非常に低くなり，FN となっている．一方で，赤い車両を白い車両と認識しても，自動

運転システムは他車と充分な距離があるという事実を正しく認識できている．そのため，仕様に違反しておらず，提案テストが TP となることは妥当であることが分かる．

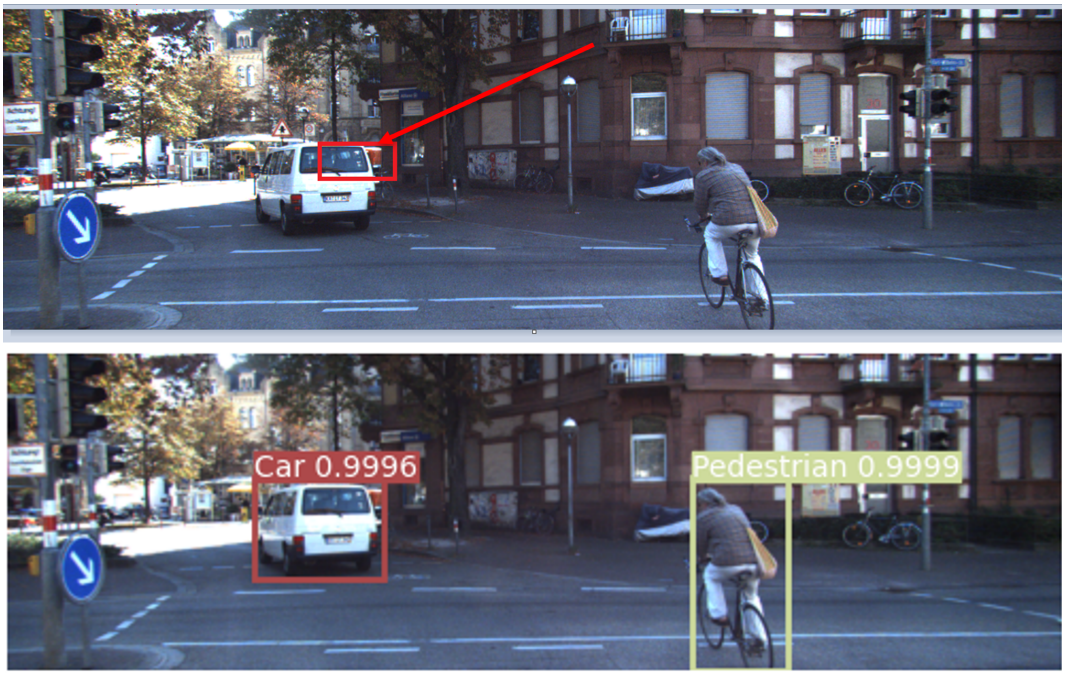

図 6.8: 表 6.3 の④に該当するテストケースの例

次に，表 6.3 の⑤に該当するテストケースを取り上げる．図 6.9 に示す画像の右端にある赤い車両が提案テストでは FP となり $IoU_{0.6}$ では TP となる⑤に該当するテストケースの 1 つであり，推論結果のバウンディングボックスは紫色で示されている．この車両は自車から充分に離れた位置にあるため，図 3.7 の仕様 S_1 を基に期待値はケース"NOT stop"となる．推論結果のバウンディングボックスは，y 軸方向に多少のズレが見られるが，赤い車両に対して僅かなズレであるため，$IoU_{0.6}$ において TP である．一方で，これは y 軸方向の下向きのズレ，つまり，自車との距離間をまちがって認識している．実際，推論結果は仕様に照らし合わせて"stop"となっており，停止する必要のない環境を，停止すべき環境に間違って認識している．そのため，ADS の安全性を低下させるケースとして，提案テストが FP となることは妥当であることが分かる．

このように，IoU ベースの判定とは異なり，提案テストは判断基準に上下左右のズレを反映させることができる，そのため，図 6.7 の黒い車両のように推論結果のズレの程度が大きい場合でも，自車の反応に影響を及ぼさないものを TP とすることができ，一方で，図 6.9 の赤い車両のようにズレの程度がわずかでも，自車の反応に影響を及ぼすものを厳しく検出し，FP と判定することができる．これは，提案テストの重要な利点である．

次に，表 6.3 の⑥に該当するテストケースを取り上げる．図 6.10 に示す上の画像の赤いバウンディングボックスで囲まれた車両が提案テストでは FN となり $IoU_{0.6}$ では TP となる⑥に該当するテストケースの 1 つであり，推論結果は下の画像の同じ位置にあるバウンディングボックスである．この黒い車両は自車にかろうじて近いとみなせる距離 (テスト時に与えた stoppingDistance に僅かに重複する距離) のため，図 3.7 の仕様 S_1 を基に期待値はケース"stop"となる．推論結果のバウンディングボックスはほとんどズレがないため，$IoU_{0.6}$ において TP である．しかし，推論結果のバウンディングボックスは僅かに実際よりも奥に認識しているため，仕様の停止条件の境界を越え，停止すべき環境を停止す

図 6.9: 表 6.3 の⑤に該当するテストケースの例

る必要のない環境と認識しているとみなされ提案テストではFNと判定する．このように，提案テストは仕様のケース間の境界にあたる環境に対して非常に厳しく機能する．

図 6.10: 表 6.3 の⑥に該当するテストケースの例

次に，表 6.3 の⑦に該当するテストケースを取り上げる．図 6.11 に示す画像の建物に部分的に隠れているオレンジ色の車両が提案テストではFPとなり $IoU_{0.6}$ ではFNとなる⑦に該当するテストケースの 1 つであり，図 6.8 の場合と異なり，白い車両の推論結果のバウンディングボックスはオレンジ色の車両と全く重なっていないため，オレンジ色の車両に対して何も認識されなかったとみなします．このオレンジ色の車両は自車から充分に離れた位置にあるため，図 3.7 の仕様 S_1 を基に期待値はケース"NOT stop"となる．対象のオレンジ色の車両を全く認識できていないため，$IoU_{0.6}$ は 0 となり FN となる．一方で，存在していた車両に対して，推論結果ではその車両が存在しない環境として認識しているこのケースは，仕様 S_1 において前提条件を満たさない状況となっている．このように，提案した機能テストは，仕様の前提条件を満たす画像のみでテストを行っても，推論結果が

仕様の範囲外として認識されることがある．

図 6.11: 表 6.3 の⑦に該当するテストケースの例

最後に，表 6.3 の⑧に該当するテストケースを取り上げる．図 6.12 に示す左端にわずかに見切れている黒い車両が提案テストと $IoU_{0.6}$ の両方とも FN となる⑧に該当するテストケースの 1 つであり，図 6.11 と同様に，この車両に対して何も認識されなかったとみなします．この黒い車両は自車に非常に近い位置にあるため，図 3.7 の仕様 S_1 を基に期待値はケース"stop"となる．対象の黒色の車両を全く認識できていないため，$IoU_{0.6}$ は 0 となり FN となる．このテストケースは，図 6.11 と同様に，推論結果が仕様の範囲外 (前提条件を満たさないケース) を示す．図 6.11 の場合は，期待値のケースが"NOT stop"であるため，検出できなくてもあまり安全性に問題はないようにみえる．しかし，図 6.12 の場合は，期待値のケースが"stop"であり，自車と非常に近い位置にある車両を検出しそこねているので，自動運転システムの安全性が損なわれている．このように，仕様の対象の画像をテストした時に，仕様外に検出された場合は，安全性を損なう場合も，あまり損なわない場合も含んでいることが分かる．そのため，提案テストでは，安全性を優先し，仕様外に検出された場合は F として判定するように定義している．

図 6.12: 表 6.3 の⑧に該当するテストケースの例

以上の考察により，提案テストは，指定された OEDR 要件に沿った安全性が重要な動作決定基準を考慮して妥当な信頼性評価を提供する上で大きな利点を示している．バウンディングボックスの検出精度に焦点を当てる IoU などの従来の評価方法とは異なり，提案手法は，自動運転システム全体のコンテキストを含む機能仕様にどの程度準拠できているかを重視している．提案手法の焦点は，仕様からのわずかな逸脱でも深刻な結果につながる可能性がある自動運転システムなどの大規模で安全性が重要なアプリケーションに画像認識システムを用いている場合に特に重要である．提案テストは，事故につながる可能

性のあるリスクを特定するだけでなく，画像認識システムが仕様で定義されている安全性要件に厳密に準拠していることを保証する．これにより，安全な自動運転システムの設計と開発に寄与するための，より包括的で安全性を重視した評価が実現できる．一方で，この機能テストの結果，仕様に準拠できてない場所が明らかになったときに，どのように修正するかについての指針にはなりにくいことはこの手法の欠点である．システム全体の要求と実装の乖離の大きい DNN ベースのシステムに対して，BBSL では，自動運転システム全体のコンテキストをベースに仕様の作成を目指したため，仕様が実際のシステムの構成要素や DNN の構造とは無関係であり，どのような学習データで再学習すべきか，センサーの精度の問題なのか，根本の原因を特定することが困難である．また，そもそも仕様違反の原因が多様で、テスト結果の解釈が難しい．ただし，提案する機能テストは，次節で考察する BBSL 仕様ベースカバレッジや，仕様そのものの情報から，その仕様違反の重要度などの分析をしやすいくする工夫がある．また，別の観点では，現状の提案された機能テストの実装は汎用化されてはいない．本研究で実施したすべてのテストは，対象の機能仕様と同等の条件を持つプログラムを，手動で Python により実装したものである．テストプロセスの効率を高めるには，BBSL で記述した仕様をテストプログラムに自動的に変換できるパーサーの開発が次のステップとして重要である．

6.4 BBSL 仕様ベースカバレッジの有効性

ここでは，5.4 節で BBSL 仕様ベースカバレッジを評価した結果を用いて，100%未満のカバレッジが重要である場合とあまり重要ではない場合について説明し，これらのカバレッジを基に，欠落しているケースの条件をどの程度推測可能にするかを考察する．さらに，テストデータの量を増やすだけでは十分なテスト精度が保証されないことを明確にすることで，提案したカバレッジを指標として用いる重要性を考察する．加えて，テストが包括的かどうかを判断する際にこれらのカバレッジがどのように有効であるかについて考察する．

初めに，BC_c が 100%であっても，BC_d が 100%でない場合は，仕様に記述した自動運転システムの反応の少なくとも 1 つ以上テストされておらず，非常に不適切なテストであることを考察する．図 6.13 は，Td_{1a} のテストデータに含まれるすべてのテストケースに対応する車両を示している．テストデータ Td_{1a} は，図 6.13 に示す車両 1 から 7 を対象とした 7 つのテストケースで構成されている，仕様 $S3$ を使用した表 5.10 に示す 4 種類のテストのうち，BC_c のみが 100%となるのはこのテストのみである．このテストは，進行方向に沿って車両がある条件とない条件，近くに車両がある条件とない条件という仕様 $S3$ に記述した全ての条件の真偽値をカバーしている．しかし，近くに車両があり，かつ，進行方向に位置しているケース"stop"に対応するテストケースがないため，BC_d は 100%ではない．つまり，自動運転システムが正しく停止するための環境を認識できるか否かは全くテストされていない，このように，BC_c が 100%であっても，BC_d が 100%でないテストは，仕様上，非常に致命的なケース漏れが存在していることを示している．

次に，BBSL 仕様ベースカバレッジが未テストのテストケースの分析をどのように容易にするかを考察する．図 6.14 は，Td_{1b} のテストデータに含まれるすべてのテストケースに対応する車両を示している．テストデータ Td_{1b} は，図 6.14 に示す車両 1 から 5 を対象とした 5 つのテストケースで構成されている．仕様 $S3$ を使用した表 5.10 に示す 4 種類のテストのうち，BC_{mc} のみが 100%でないのはこのテストのみである．BC_{mc} を 100%にするために必要なテストケースは，進行方向に位置しておらず，自車から遠い場所に位置している車両 (図 6.14 の※印が指すケース) のテストである．BC_{mcd} は，4.3 節で示したように，最優先すべき条件の組み合わせで定義されているため，センシティブな条件の組み合

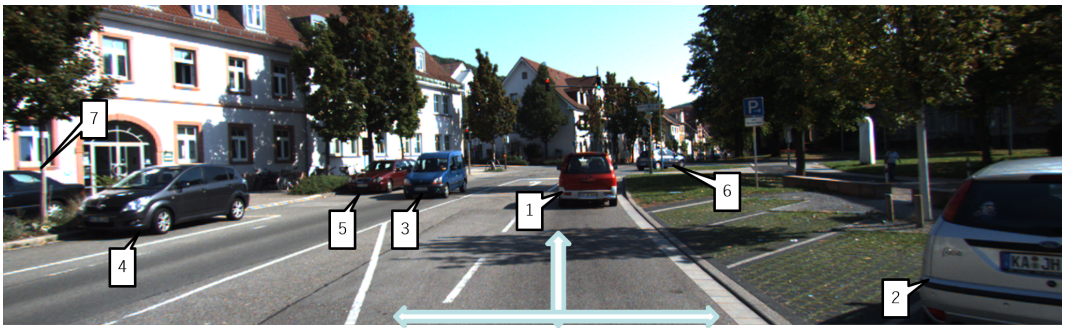

図 6.13: 表 5.10 におけるテストデータ Td_{1a} に含まれる全てのテストケース

わせであるこのケースが欠落していることは明らかである．このように，BBSL 仕様ベースカバレッジ内で値を比較し，未テストのテストケースの分析を容易にする．

図 6.14: 表 5.10 におけるテストデータ Td_{1b} に含まれる全てのテストケース

次に，BC_{mcd} が 100%でない場合，テストされていないコーナーケースが存在し，自動運転システムの誤動作につながる可能性があることを考察する．図 6.15 は，Td_{1c} のテストデータに含まれるすべてのテストケースに対応する車両を示している．テストデータ Td_{1c} は，図 6.15 に示す車両 1 から 8 を対象とした 8 つのテストケースで構成されている．このテストでは，BC_{mcd} も BC_{mc} も 100%ではない．BC_{mc} は，Td_{1b} の場合と同じ 75%だが，BC_{mcd} を満たすテストケースも未テストとなっている．このテストケースは，車両が自車の近くにいるが，進行方向には位置していないケースである．このケースに該当する車両に対して，進行方向に位置している車両と誤認してしまうと，自動運転システムが停止すべきでない状況で停止してしまい誤動作してしまう可能性がある．従って，このテストケースは，Td_{1b} でテストされなかったテストケースよりも優先度が高くなる，このように，BC_{mc} が同一であっても，BC_{mcd} が低いテストデータの方が，自動運転システムの誤動作につながるケースを見逃している可能性が高くなる．

次に，すべての BBSL 仕様ベースカバレッジを 100%にするための最小のテストデータについて考察する．図 6.16 は，Td_{1d} のテストデータに含まれるすべてのテストケースに対応する車両を示している．テストデータ Td_{1d} は，図 6.16 に示す車両 1 から 4 を対象とした 4 つのテストケースで構成されている．これらの 4 つのテストケースは仕様 $S3$ に対してすべての BBSL 仕様ベースカバレッジを 100%にするために必要な最小のテストデータを表している，4.3 節の定義により，任意の仕様において，BC_{mc} が 100%に達すると，ほかの全ての BBSL カバレジも 100%となる．しかし，仕様 $S3$，仕様 $S5$ を含む多くの仕様において，一般に BC_{mc} の分母は BC_c や BC_{mcd} よりも小さくなる．これは，BC_{mc} 以外

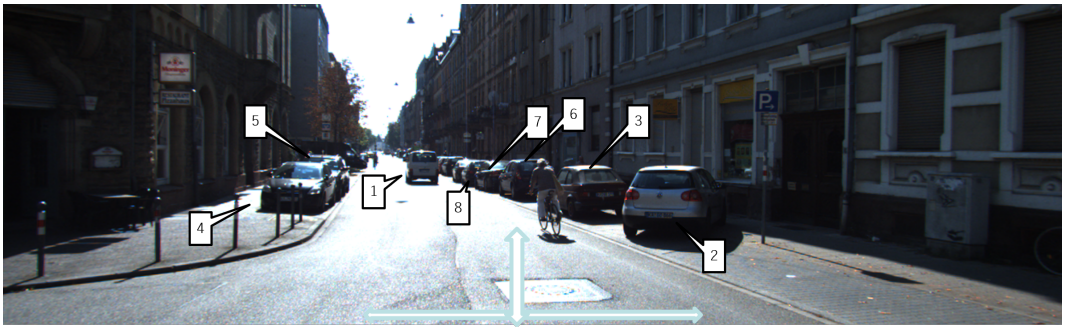

図 6.15: 表 5.10 におけるテストデータ Td_{1c} に含まれる全てのテストケース

の BBSL 仕様ベースカバレッジでは，同じ条件の重複を考慮していないためにおきる．繰り返しになるが，BC_{mc} 以外の BBSL 仕様ベースカバレッジでは，仕様上に何度も出てくる条件程テストの重要度が高いという観点から仕様の構文に重点を置いて定義している．結果として，BC_{mc} のみが，任意の仕様において分子が高々1 増加するように設計されている．このような設計であることから，任意の仕様において全ての BBSL 仕様ベースカバレッジを 100%にするための最小のテストケース数は，BC_{mc} の分母と同じである．

図 6.16: 表 5.10 におけるテストデータ Td_{1d} に含まれる全てのテストケース

仕様 S_3 のような小さな仕様では，すべての BBSL 仕様ベースカバレッジを 100%にすることは可能であるが，より大きな仕様では極めて困難になる．特に，テストデータの量を単純に増やすだけでは，これらのカバレッジを向上させるのにあまりにも非効率な場合がある．BBSL 仕様ベースカバレッジの利点は，仕様に応じて，テストされていないケースが致命的なものか，あるいは，起こりそうにない無視しても問題のないケースなのかを判断する助けになることである．これらのことを確認するため，比較的大きな仕様である仕様 S_5 における BBSL 仕様カバレッジについて考察する．表 5.10 における，2736 のテストケースを含む Td_1 の場合，BC_c のみが 100%となっている，そこから，テストケースを増やして 25140 にした $(Td_1 + Td_2)$ の場合でも，100%に達しているのは BC_c のみである．75429 のテストケースからなる $(Td_1 + Td_2)^*$ まで増やした場合に，初めて BC_d が 100%に達する．図 6.17 は，$(Td_1 + Td_2)^*$ において BC_d を 100%に増加させたテストケースを示している．このテストケースは，図 5.12 が示す仕様 S_5 における 71 行目から 83 行目のケース"deceleration"に該当するテストケースであり，自車が右にも左にも車線変更できず，しかし，減速する必要がある状況に対応している．つまり，対象の車両が"leftZone"，"rightZone"，"decelerationDistance"とは重なり，"stoppingDistance"とは重ならない状況を示している．図 6.17 を見ても分かるように，このような状況は他のケー

スに比べて実際の走行環境で発生頻度が低いため，$(Td_1 + Td_2)^*$ に至る全データセットで BC_d が 100%に達さなかったことが理解できる．このように，自動運転システムの応答の中には実際の走行環境上で非常に遭遇する機会の低い状況に対応するものを含んでおり，実際の走行データからテストデータを作成する場合，単にテストデータを増やすだけでは，重要度の高い BC_d ですら充分にカバーできない可能性がある．これは，自動運転ではない従来の車の人間による運転時のタスクにおいて，障害物に対してシビアな反応すべき状況は実際の走行環境上で遭遇する機会が決して高くはないことからも直観に従っている．BBSL 仕様ベースカバレッジはこのような実際の走行環境上で非常に遭遇する機会の低い状況を無視せずに，自動運転システム全体のコンテキストを考慮した指標を提供する．これは，BBSL 仕様ベースカバレッジの大きな利点である．表 5.10 に示すように，BC_{mcd} および BC_{mc} は，$(Td_1 + Td_2)^{**}$ の場合でも 100%には達しない．テストケース数をさらに増加させることでこれらのカバレッジをもう少し増加させることは可能ですが，100%のカバレッジを達成することは極めて困難であると考えられる．これは，実際の運転環境では，物理的な制限により，車両の位置に関する暗黙の，あるいは未知の制約が存在するためである．例えば，図 6.17 において，車両が"directionArea"と"stoppingDistance"とは重なっているが，"leftZone"と"rightZone"にはどちらにも重なっていない位置にある車を仮定する．このような位置に車が存在するには，非現実的なほど小さい車が存在するか，あるいは，自車に一部埋め込まれているような状況でない限りおこりえない．このような例があり得るため，BC_{mcd} と BC_{mc} は，BC_d とは異なり，常に 100%である必要はない．重要なことは，BC_{mcd} と BC_{mc} が 100%に達しない場合は，そのケースが実際の走行環境上で非常に遭遇する機会の低いが 0 ではなくテストすべきケースなのか，非現実的でテストしなくてよいケースなのかを判断することであり，BBSL 仕様ベースカバレッジはその判断を助ける指標として有効である．

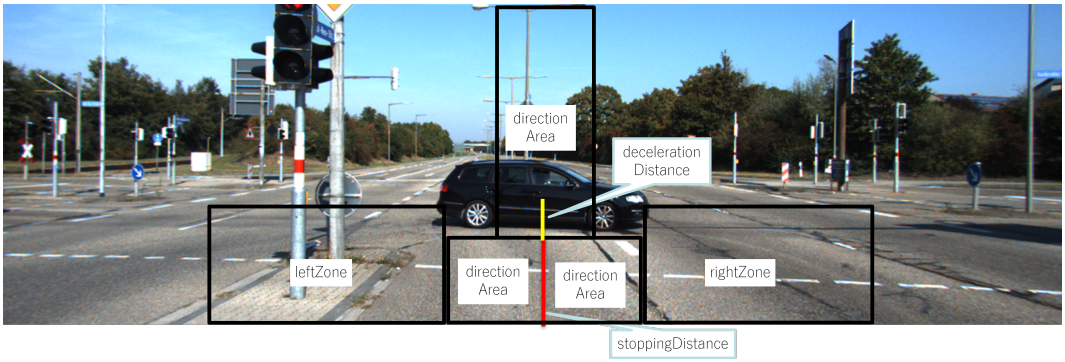

図 6.17: $(Td_1 + Td_2)^*$ において BC_d を 100%に増加させたテストケース

以上の考察により，BBSL 仕様ベースカバレッジの特徴と有効性を考察した．特に，実際の走行環境上で非常に遭遇する機会の低いが 0 ではなく，遭遇した場合に致命的な事故につながる可能性のある重要なケースを含め，テストデータが仕様を満たす範囲を特定できることは BBSL 仕様ベースカバレッジの利点である．実際には，100%のカバレッジを達成することが常に実現可能であるわけではなく，また，100%であることが必要であるとは限らないため，適切なカバレッジとその目標レベルを慎重に決定することが重要である．実際の走行環境上で非常に遭遇する機会の低いシナリオの中には，自動運転システムの安全性を確保するために重要なものもあるが，非現実的またはそれほど重要でないものも含まれていた．したがって，徹底的なテストと実際の制約の間でバランスを取る必要がある．本論文で提案した BBSL 仕様ベースカバレッジは，仕様に基づいてテストされてい

ないシナリオを特定するための貴重なツールである．従来の車両の種類や道路の構成情報などを用いたODDベースのカバレッジとは異なり，BBSL仕様ベースカバレッジは，仕様内の関心のある位置関係がどの程度テストされているかについて，より詳細な評価を実現する．また，このように特定の位置関係のカバレッジの範囲を評価できるこの特徴は，仕様記述にBBSLを使用することの大きな利点でもある．一方で，これらのカバレッジを100%にするためのテストケースは仕様によって大きく異なり，場合によっては明らかに少ないテストケースで簡単に100%になる場合も存在しているという点で，絶対的な指標としての運用が困難であることはBBSL仕様ベースカバレッジの欠点となる．そのため，次節で考察する空間カバレッジと補完して使用することで，この欠点を軽減する工夫がある．また，これらのカバレッジを満たすために必要なテストデータの生成方法が未解決であることは，依然として大きな課題である．カバレッジ基準を満たすテストケースを手動で作成することは，手間がかかり，非現実的である．この制限に対処するために，今後の研究では，BBSL仕様ベースカバレッジを満たすテストケースを生成する手法に重点を置く必要がある．具体的には，Carla[41]のような自動運転システム用のシミュレータ使用することで，各カバレッジが該当するテストケースが実環境上で物理的に起こりうるか，起こる場合にどのようなケースが不足したカバレッジの一例になるか判断する方法が考えられる．このように，カバレッジを考慮しながらテストケースを準備することができれば，テストプロセスの効率と有効性が大幅に向上する．結果として，重要なシナリオが徹底的にテストされることで自動運転システムの安全性がより向上する可能性がある．

6.5 飽和判定を行うための適切なデータセットとウィンドウサイズ

ここでは，5.5節で空間カバレッジを評価した結果を用いて，飽和判定を行うための制限と，その効果について考察する．具体的には，空間カバレッジを用いて飽和度を決定するのに，時間的に独立したデータセットを使用する必要があることを論じる．また，ウィンドウサイズを大きく設定することに応じて，飽和時の空間カバレッジが理論上の最大値に近づくことを示す．

図5.14，図5.15，図5.16が示すように，このテストデータに対する2つの空間カバレッジはの測定結果は非常に偏った曲線を示しており，飽和判定が困難なグラフとなった．これらのテストデータ内のオブジェクトの位置とサイズにみられる偏りには2つの理由があると考えられる．1つ目の理由は，KITTIを含む既存の自動運転システムの画像データセットは，特定のシナリオに沿って実際車を走行させて取得された時間的に連続した画像で構成されているためである．この場合，画像上のオブジェクトは時間順に相互に依存している．例えば，図6.18は，KITTIのデータセット上で連番になっている4枚の画像データと，その時の車両をバウンディングボックスで表した図である．図6.18に示すように，自車の前に車両が存在する場合，前方の車両の位置とサイズは後続の画像でもしばらくほとんど変化しない場合が多々ある．2つ目の理由は，画像内の地理状況により，1枚の画像内で特定の位置にオブジェクトが集中することが多いためである．例えば，図6.19は，KITTIのデータセットのうちのある1枚の画像データと，その時の車両の一部をバウンディングボックスで表した図である．図6.19に示すように，左右に複数の車両が並んで駐車されている場合，それらの位置に偏りが生じる．このうち1つ目の時間的な依存を排除するため元画像から復元抽出して再構築したデータセットがTd_1^*とTd_2^*であり，その結果は，図5.17と図5.18が示すように概ね飽和曲線を示している，

1つ目の時間的な依存を排除しても，地理状況により依存するため完全な飽和曲線では

図 6.18: データセット上でオブジェクトの位置やサイズが時間順に依存する例

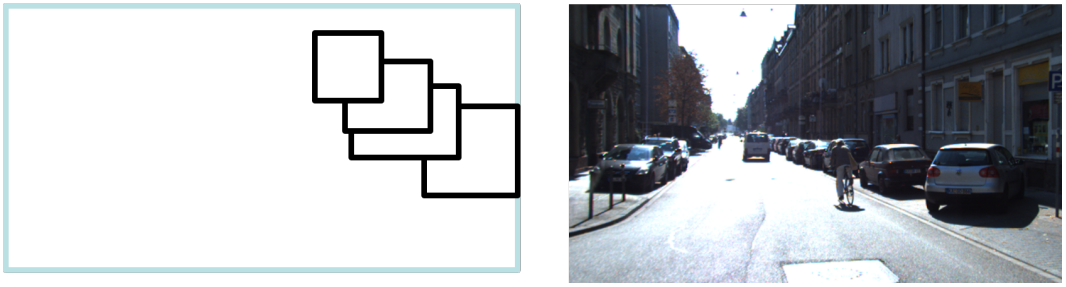

図 6.19: データセット上でオブジェクトの位置が地理状況により依存する例

ないが，ウィンドウサイズを適切に指定することでこの問題を軽減することができる．表 6.4 は，データセット Td_1^* に対するウィンドウサイズ h の値ごとの SC_{pos} と SC_{siz} 上の飽和タイミングを示している．一方，表 6.5 は，データセット Td_2^* に対する各飽和タイミングを示している．Td_1^* における SC_{pos} の理論上の最大値は，母集団の Td_1 における SC_{pos} の最大値である 6.15%となる．そして，ウィンドウサイズを大きくする程飽和した時の SC_{pos} は理論値の 6.15%に近づく．これは，SC_{siz} についても同様である．また．Td_2^* においても，ウィンドウサイズを大きくする程飽和した時の SC_{pos} や SC_{siz} は理論上の最大値に近づく．重要なことは，飽和するために理論値に近いカバレッジで停止することと，そのために必要なデータセットの数はトレードオフにあることである．また，開発する自動運転システムの想定されるシナリオにもよるため，ウィンドウサイズを置き，柔軟に飽和タイミングを指定することが可能なように定義している．

表 6.4: データセット Td_1^* に対するウィンドウサイズ h の値ごとの SC_{pos} と SC_{siz} 上の飽和タイミング

h	SC_{pos} 上の飽和タイミング (何番目のテストケースか)	SC_{siz} 上の飽和タイミング (何番目のテストケースか)
200	5814	4473
500	6114	9176
800	11586	9476
1000	11786	13320

このように空間カバレッジは，テスト設計者が関心のある領域内でどの程度充分にテス

表 6.5: データセット Td_2^* に対するウィンドウサイズ h の値ごとの SC_{pos} と SC_{siz} 上の飽和タイミング

h	SC_{pos} 上の飽和タイミング (何番目のテストケースか)	SC_{siz} 上の飽和タイミング (何番目のテストケースか)
200	5321	8101
500	15516	10727
800	18796	15116
1000	18996	15316

トされたかを表す指標としての利点がある．SC_{pos} と SC_{siz} はそれぞれデータセット内のオブジェクトの位置とサイズが充分にテストされているかを示すテスト基準を表す．そして，ウィンドウサイズ h を設定することで，これらの基準を使用して自由に飽和状態を判定することができる．ただし，これらは既存のテストデータセットに関して汎用的に利用することができないという欠点も存在している．前述したように，テストに使用するデータセットが時間に依存しないように注意する必要がある．また，もう一つ触れておくべき事項として，本研究の測定した結果の SC_{pos} と SC_{siz} の最大値が絶対値として非常に小さい点があげられる．例えば，データセット Td_1^* に対する SC_{pos} は理論値の最大値でも高々 6.15%である．これは，今回の測定で位置クラス R とサイズクラス S の両方を画像全体の範囲に適用したためである．一般的に自動運転システムのデータセットにおいて母集団を定義することは困難であるが，位置クラスとサイズクラスを使用して対象の領域やサイズを指定することで，実際はより効果的に空間カバレッジを使用できることが期待できる．

第7章　関連研究

7.1　形式仕様記述言語

形式仕様記述言語とは，設計対象の機能や性質を数理論理学 [53] などの数学的記法に基づいた言語で表現し，その言語体系を利用して開発プロセスにおいて不具合を検出したり，一貫した設計を実現させる手段の総称である．仕様を形式的に記述することで，設計段階での曖昧さを排除し，システム全体の信頼性や安全性を向上させることに寄与する．形式仕様記述言語は形式手法の一部を構成しており，システムの仕様を数学的な記号や理論に基づいて厳密に記述するためのツールとして用いられる．この言語は，システム設計における厳密性と正確性を保証するだけでなく，開発プロセスにおける検証や検査を自動化する手段としても利用される．

形式仕様記述言語としてZ記法は，1980年代初頭から存在しており．集合論と一階述語論理を基盤とする形式仕様記述言語である [32]．システムの関数や状態を「スキーマ」と呼ばれる表形式で記述することで，仕様の構造を視覚的に整理し，複雑なシステムの設計において有効である．ソフトウェアだけに限らずハードウェアシステムの設計や文書化を目的として用いられる場合もある [54]．例えば，マイクロプロセッサの命令セットの仕様記述に使用された例がある [55]．Z記法は一般的には，実行可能な仕様ではないが，実行可能にするための研究は存在している [56]．また，Z記法は優れた型チェックツールが存在しており，代表的なものに，Zのリファレンスマニュアル [57] をもとにした，fuzz type checker がある [58]．

VDM(Vienna development method) は，1970年代の前半にIBMのウィーン研究所によって，プログラミング言語PL/Iの仕様記述や，コンパイラの検証のために開発された形式手法全般を指す言葉である [59]．VDMを適用するためには，対象の仕様を形式仕様記述言語で記述する必要があるのだが，このときによく使用される形式仕様記述言語として，大きく3種類の異なる形式のシステムをサポートする言語が存在している．1種類目は，VDM-SL(Specification Launguage) と呼ばれる形式仕様記述言語であり，逐次的なプログラムの機能仕様におけるモジュールの構造を記述するための言語である．1996年に，ISOの支援を受けて標準化もされている [60]．2種類目は，VDM++と呼ばれる形式仕様記述言語であり，並列かつリアルタイムで動作を行うオブジェクト指向向けのプログラムの機能仕様を記述するための言語である．[61] この言語は，VDM-SLを基にして，クラスやオブジェクトの概念を取り入れて拡張されており，多くの場合，クラス単位で構造を記述する．そして，3種類目は，VDM-RT(Real-Time) 言語や，VICE (VDM++ In Constrained Environments) と呼ばれる形式仕様記述言語で，リアルタイム組み込みシステムや分散システムを記述するための言語である [62]．この言語は，VDM++に対して，時間的な振る舞いの記述ができるように，また，スレッドの概念を導入し，並行プロセスを記述可能にするために拡張された言語である．これらのVDM関連の形式仕様記述言語は，状態遷移や機能仕様を数学的に表現することで，設計プロセス全体の一貫性を確保し，不具合の早期検出を可能にする．VDMの特徴は，システムの状態をモデル化することに焦点を当てており，集合や関数，論理式を用いている点である．これにより，設計段階での仕様の正

確性を数学的に検証し，実装段階でのエラーを防ぐことを可能としている．

このように，形式仕様記述言語は，システムの設計における曖昧さを排除し，信頼性を向上させることに寄与するが，VDM-SL や Z 記法は，状態遷移やデータ操作の記述には優れているものの，画像認識のような空間的な仕様を記述するには適していない．この点が，本研究で提案した BBSL との違いである．BBSL は，物体間の位置関係や空間的な仕様を正確に記述するために設計された形式仕様記述言語であり，従来の形式仕様記述言語が対応できなかった画像認識の特性を考慮している．

7.2 画像上の位置関係の技術

画像上の位置関係を形式的に記述するための先行研究として，地図データベースにおける特定の地形を検索するためのクエリ言語が存在する．この研究では，まず 1 次元のトポロジーを定義し，その後これを多次元の位置関係に拡張する手法が提案されている [34]．トポロジーベースの記述言語は，地図データベースや CAD データといった空間データを処理するために設計されており，特定の地理的特徴を効果的に検索・管理することを可能にする．この技術は，オブジェクト間の空間的な位置関係を厳密に記述し，それをもとにデータベース操作を行うための理論的な基盤を提供している．

また，別の既存研究では，画像表現において，スケール不変特徴変換（Scale-Invariant Feature Transform, SIFT）という手法がある)[63]．SIFT は，画像から局所的な特徴を多数抽出し，それを幾何学的に表現することで，部分重複検索や画像認識を行う技術である．具体的には，画像内の特徴点を検出し，それらの特徴量をスケール不変な形で記述することで，異なる解像度や視点で撮影された画像間でも対応関係を正確に特定することが可能としている．この技術は，主に画像検索や物体認識といった分野で広く利用されており，画像間の類似性を幾何学的に評価する枠組みを提供している．さらに，SIFT を応用した研究として，より大規模な部分重複画像検索のために幾何学的な表現を利用したマッチング検証手法が提案されている [64]．この研究では，画像特徴点の対応関係を幾何学的な制約を用いて検証することで，高い精度の画像検索を実現している．この手法は，従来の単純な特徴マッチングに比べて，背景ノイズや一部の特徴の欠落に対しても堅牢であり，画像間の部分的な一致を効率よく特定することが可能である．

これらの研究は，画像上のオブジェクトや特徴を形式的かつ幾何学的に記述するための枠組みを提供している点で共通しているが，それらが対象とする分野は自動運転システム以外の特定の応用分野に限定されている．地図データベースや画像検索において，これらの技術は重要な役割を果たしている一方で，自動運転システムの画像認識には，その検出対象であるバウンディングボックスの方向などを伴って位置関係が重要な要素のため，そのような位置関係にに特化した空間関係の記述技術を構築する必要性がある．

7.3 画像認識のテスト

画像認識におけるテストの一般的な手法として，DNN を使用して推論されたラベルと，それに対応するグラウンドトゥルースラベルを照合する方法が挙げられる．グラウンドトゥルースラベルとは，対象の画像に含まれる物体や位置関係を正確に示す基準となるデータであり，通常は手動でラベル付けされることで準備される [65][66]．このようなラベルは，自動運転システムをはじめとする多くの画像認識タスクで性能評価を行うための基礎となるものである．性能評価では，グラウンドトゥルースラベルと推論結果のバウンディングボックスの位置や形状がどの程度一致しているかを測定することが求められる．その際，

Jaccard インデックスとしても知られる Intersection over Union（IoU）が一般的に用いられる [11]．IoU は，グラウンドトゥルースと推論結果のバウンディングボックスの共通部分と，それらの全体部分の比率として定義される．この指標は，物体検出の精度を評価するための標準的な手法として広く採用されており，閾値を設定して一致の基準を決めることで，その true，false が定められている．自動運転システムの画像認識においても，IoU は性能評価の指標として用いられている [20]．例えば，PASCAL VOC Challenge などでは，この手法を使用して物体検出アルゴリズムの性能を比較してきた．しかし，IoU のような単純なズレによる閾値に基づく評価は，必ずしも自動運転システムの安全要件を十分に反映しているとは言い難い．本研究では，このような仕様に基づく性能評価を提案し，従来の IoU に基づく単純な評価では反映できなかった仕様の要件を反映させるためのアプローチを提案している．

一般的な画像認識のテストとは異なり，自動運転システムにおける画像認識のテストでは，DNN モデルの入力空間が非常に広大であることから，すべての入力画像に対応する正しい出力を得るためのオラクルを定義することは重要な課題となっている．従来のソフトウェアテストでは，テスト対象システムの期待される出力を明確に定義できる場合も多いが，自動運転システムの画像認識においては，入力の多様性や環境要因の影響により，一般的なオラクルを直接構築することが難しい．この課題に対する 1 つの有効なアプローチとして，メタモルフィックテストが提案されている [67]．メタモルフィックテストは，テストオラクルが存在しない場合でもシステムのテストを実現するために導入された手法である．この手法では，入力と出力の絶対的な関係を定義するのではなく，入力間に存在する一般化可能な関係，すなわちメタモルフィック関係を利用してシステムの挙動を間接的に検証する．自動運転システムにおいては，画像やフレーム間に存在するさまざまなメタモルフィック関係が研究されており，これに基づいて認識モジュールのテストが行われている．たとえば，画像認識におけるメタモルフィック関係の 1 つとして，元の画像で検出された物体が合成画像でも同様に検出されるべきであるという関係がある [68]．このような関係は，画像を変形や変換することで新たなテストケースを生成し，テストデータを増大させ，それに基づいて画像認識のテストの充分性を増加させるために使用される．また，LiDAR ベースの物体検出においては，関心領域（Region of Interest, ROI）の外側に存在するノイズポイントが ROI 内の物体検出に影響を与えない，というメタモルフィック関係が利用されている [69]．これにより，センサーデータのノイズ耐性や検出精度を効率的に評価することが可能となる．さらに，自動運転システムの動的なシナリオを考慮し，シナリオ内の異なるフレーム間でのオブジェクト検出に関するメタモルフィック関係も提案されている．具体的には，あるフレームで検出された物体がその後のフレームでも継続して検出されるべきである，という関係である [70]．この手法は，連続する画像フレーム間の整合性を検証することで，認識モジュールが動的な環境下でも一貫したパフォーマンスを維持できるかを評価するために用いられている．以上のように，メタモルフィックテストは，自動運転システムの認識モジュールのテストにおける重要な手法の 1 つであり，オラクルを直接構築することが困難な状況においてもシステムの信頼性を評価する手段を提供している．

7.4 テストカバレッジ

テストの充分性基準のためのカバレッジには，プログラムベースのカバレッジ，仕様ベースのカバレッジ，インターフェースベースのカバレッジなど測定に使用される対象によっていくつかの種類が存在する．プログラムベースのカバレッジの中でも主要なカバレッジの 1 つに，コードカバレッジがある [37][14]．コードカバレッジは，特定のテストスイート

の実行時にプログラムのソースコードがどの程度実行されるかを示す指標である．コードカバレッジには，ソースコード上のどの項目に着目するかで，ステートメントカバレッジや，ブランチカバレッジ，コンディションカバレッジなどの種類が存在する．その中でも，最も一般的に知られ，広く使用されるカバレッジは，改良条件判定カバレッジ (MC/DC) である．このカバレッジは，条件のあらゆる組み合わせを試すには量が多く現実的でない場合に，優先度が高い組み合わせのみに着目て与えられるカバレッジで，優先度をどのように与えるかによって，複数の定義が混在している [38][39][40].

仕様ベースのカバレッジは，仕様の種類によってそれぞれ独自に提案されており，代数的仕様に対するカバレッジ [71] や，Prolog に対するカバレッジ [72] があり，通常それらの仕様から効率よく自動的にテストケースを作成するために用いられる．

さらに，画像認識のような DNN モデルを使用するソフトウェアに対しても独自のカバレッジがいくつか提案されている．DNN モデルに対する主なテストカバレッジはニューロンカバレッジである [15]．これは，利用可能なすべてのテスト入力が DNN に渡された場合際に，すべてのニューロンに対する活性化されたニューロンの割合として定義される．また，MC/DC カバレッジに影響をうけて，DNN モデルの隣接する層のニューロン間の因果関係を用い 4 つのカバレッジを提案した研究がある [73]．これらの DNN モデルに対するカバレッジは，敵対的手法を使用して悪意を持って人工的にテスト入力を作成し，DNN モデルの堅牢性をテストする．したがって，これらのカバレッジのほとんどは，DNN の構造に着目するカバレッジである．しかしながら，いくつかの研究によりこのようなカバレッジが，通常受け取られる自然な入力を用いるテストに対して使用されることは不適切と言われている [17][18][74]．例えば，これらの研究のいくつか [17][74] では，こういったカバレッジとテストセット内の誤認識された入力の数との間に有効な相関関係がないことが示されている．したがって，本研究で提案するテストにこれらの DNN モデルに対するカバレッジは効果がない可能性が高い．

第8章　おわりに

8.1　まとめ

本研究では，仕様に基づいて自動運転システムの画像認識処理をテストするために，3種類の提案を行った．具体的には，BBSL と名付けた形式仕様記述言語，BBSL で記述した自動運転システムの機能仕様と機能テスト手法，そして，機能テストに効果的なテストカバレッジ基準となる．ここでは，以下に各提案の詳細と従来のどのような貢献をしたかについてまとめる．

まず，BBSL は，画像上のオブジェクトの相対位置を記述するために設計された言語であり，画像認識システムの機能仕様を記述することに貢献した．これは，既存の非形式的な仕様に存在する位置関係に関しての曖昧さを排除するために導入された．本研究では，NHTSA の安全フレームワークで概説されている 18 種類のイベントに関して，BBSL で OEDR の仕様を記述した．その結果，BBSL は高度な抽象度を維持しながら，高々65 行程度で自動運転システムの応答を表現できることを示した．さらに，BBSL は位置関係を記述することができる既存のいくつかの手法よりも優れた表現力を備えていることを示した．このように，自動運転システムの画像認識における機能仕様の記述に言語として適している要素を明確にした．

次に，BBSL を用いた機能テスト手法は，BBSL で記述した機能仕様によって，自動運転システムのコンテキストを考慮するのが難しい従来の IoU ベースとは異なる観点で不具合を検出することに貢献した．BBSL を用いた機能テストでは，画像認識システムが機能仕様に記述した要件に準拠しているかどうかを直接評価できるため，仕様に基づいた安全性テストが可能となった．実際に自動運転システム用途として学習された 4 つの画像認識システムを使用した実験により，提案手法では，IoU ベースの方法のみでは見逃されていたリスクを特定できることを実証した．

さらに，テストカバレッジ基準では，空間カバレッジと BBSL 仕様ベースカバレッジと呼ぶ 2 種類のカバレッジ基準を提案した．BBSL 仕様ベースカバレッジは，テストケースが BBSL 仕様で定義されている空間関係やその他の要件をどの程度カバーしているかを評価することに貢献した．評価のために，最大 75,420 のテストケースにわたってカバレッジを測定した結果，従来のカバレッジ方法では評価が困難だった仕様の網羅性を評価できることが示された．特に，安全性が最も重要である自動運転システムでは，重要なシナリオを見逃さないための有効なアプローチであることが確認された．この方法では，機能テストにおいて，事故に繋がる可能性のあるシナリオを見逃していないかの充分性を評価できるため，自動運転システムの安全性の向上に貢献している．一方で，このカバレッジは，具体的な仕様の分量や条件に依存するため，機能テストをどの時点で終了すべきかを明確に定めることは困難であったために，別のカバレッジ基準として，空間カバレッジを提案した．空間カバレッジは，仕様とは独立して，特定の位置とサイズのクラスに基づいて，テストデータ内のオブジェクトの位置とサイズがどの程度カバーされているかを定量的に評価する能力があり，オープンデータセットを用いた実験結果では，空間カバレッジ指標によって，テストケースが飽和していることを評価できることが示された．しかし，この

カバレッジによって飽和判定を行うためには，一部制限がある．具体的には，時系列画像データセットでの飽和の判断は困難であり，一般的な時系列データでテストする場合は時系列の従属性を減らし，ウィンドウサイズと呼ぶパラメータに適切な値を設定する必要がある．このような制限はあるものの，テスト設計者がテストデータに対して，暫定的に飽和したことを判定することができ，効率的な機能テストの実施に貢献した．

8.2 今後の課題

8.2.1 BBSL の拡張

本研究で提案した BBSL は正面カメラから取得された画像上の位置関係を記述するもので，自動運転システムの画像認識をテストするためのものである．そして，このテスト対象の画像認識は 2 次元のバウンディングボックスを出力として返すモジュールに適合している．しかしながら，多くの自動運転システムでは，カメラや LiDAR 機器を使用して，オブジェクトを 3 次元のバウンディングボックスとして扱っている [49]．例えば，自動運転システムを作るために必要なソフトウェアの機能を一式備えた OSS として，Autoware というソフトウェアがある [75]．図 8.1 に，Autoware の認識モジュール内のノード図を示す．これは，Autoware Foundation から公開されているドキュメント上のアーキテクチャのノード図から一部抜粋し，番号を降ったものである [76]．Autoware は ROS2 と呼ばれるミドルウェアの上で動作しており，ここでのノードとは ROS2 におけるプロセスの単位である，この認識モジュールのプロセス毎の出力を調査した結果を，表 8.1 に示す．図 8.1 上の番号を降ったノードが表 8.1 のノードに該当しており，角度付きとは，オブジェクトのバウンディングボックスに関して，2d の場合，xy 平面において各 xy 軸に平行な直線で構成されるバウンディングボックスならば×．3d の場合，xyz 空間において，各 xyz 軸に平行な直線で構成される bounding box ならば×，そうでなければ○となる．ノード 1 に該当する"tensorrt_yolox"が，本研究でテストしたカメラ画像を受け取り，検出したオブジェクトとして 2 次元のバウンディングボックスを返す画像認識処理である．表 8.1 に示すように，2 次元のバウンディングボックスが出力に用いられるのはこのノード 1 のみであり，それ以降は 3 次元のバウンディングボックスが用いられている．さらに言うと，オブジェクトの過去の位置を付加したバウンディングや，最終的には，未来の軌道を確率的に表したバウンディングとなる．本研究で取り組んだ画像認識の問題点は，この Autoware 上における他のノードや，認識モジュール全体が抱える問題でもあるため，このように，現状ノード 1 のみを対象とする本研究で提案した BBSL を拡張し，3 次元のバウンディングボックスや，確率付きのバウンディングボックスを扱えるようにすることは，これらのノードにも機能テストできる可能性がある点で意義がある．

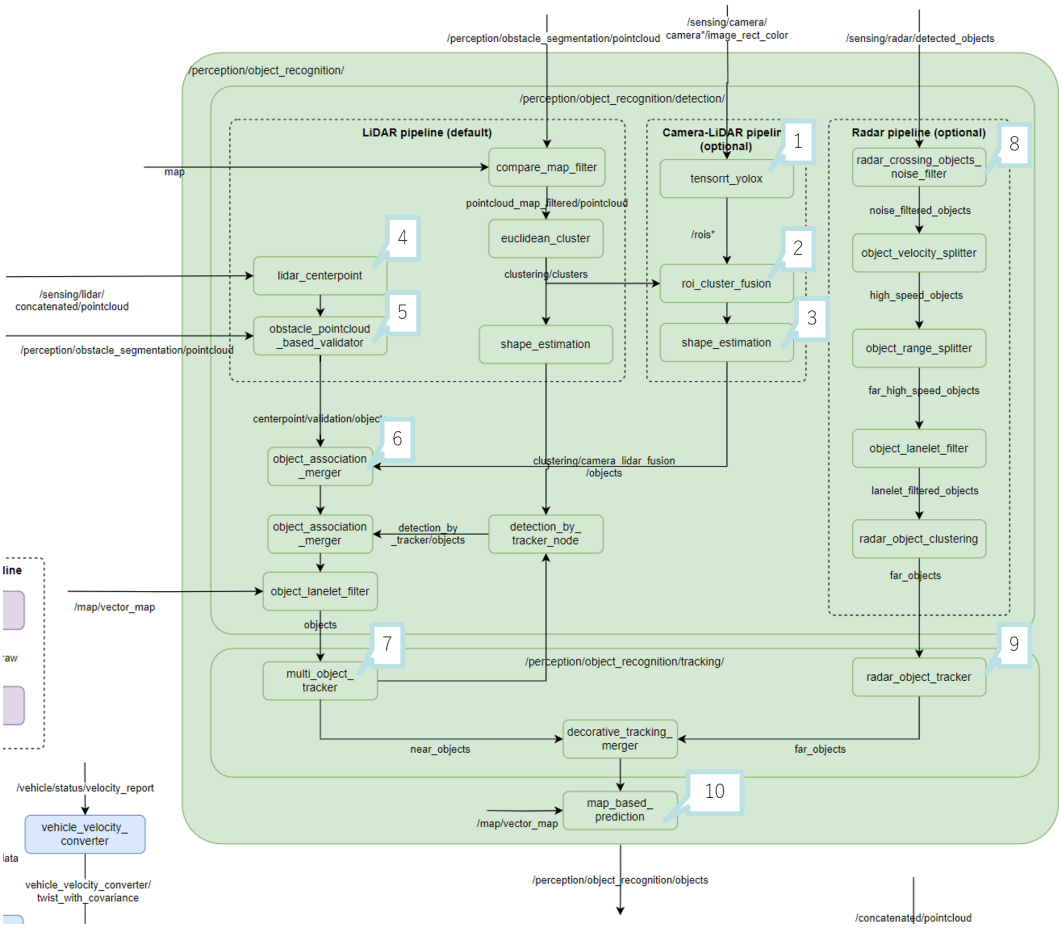

図 8.1: Autoware における認識モジュールのノード図

表 8.1: Perception モジュール内の各ノード毎のオブジェクトの出力形式

	オブジェクトの出力形式				
ノード	2dBB	3dBB	角度付き	過去の変移付き	予測 (確率) 付き
1	○	×	×	×	×
2	×	○	×	×	×
3	×	○	○	×	×
4	×	○	○	×	×
5	×	○	○	×	×
6	×	○	○	×	×
7	×	○	○	○	×
8	×	○	○	×	×
9	×	○	○	○	×
10	×	○	○	○	○

8.2.2　仕様における証明と表明

現状では BBSL の定義は，集合論などの数学的概念を用いて記述されているが，仕様そのものの正当性や性質を厳密に証明するには不十分である．そのため，まずは，BBSL 自体を Coq などの定理証明システムのような厳密な形式体系で形式化し，仕様そのものの正当性や性質を厳密に証明できる基盤を整備する必要がある．また，本論文で定義した 3 種類の性質（網羅性、排他性、非冗長性）のような仕様全般に与えられる性質とは異なり，複数の仕様との連携やその時の仕様間の優先度や，機能の一貫性のような特性などの必要性が分かったが，現状記述する方法に関して未解決である．このような表明を記述して，仕様を検査するための手法を構築することが求められる．

8.2.3　適切なテストケースの作成方法

本研究で提案した BBSL 仕様ベースカバレッジは，OEDR の仕様に基づいてテストされていないテストケースの発見に効果的であるが，その特定されたテストケースをどうのように作成するかに関しては現状未知である．また，一般の走行環境データから単に盲目的に増やした場合，100%であることが期待されるカバレッジ BC_d でさえ，網羅するデータを容易することが難しいことは既に示した通りである．そのため，BBSL 仕様カバレッジを用いて効率的にテストケースを作成できる手法を考える必要性がある．その場合，Carla[41] などの既存の自動運転システム向けシミュレーターを使用することが直観的には有効そうであるが，現状では未確認である．

参考文献

[1] On-Road Automated Driving (ORAD) Committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*, apr 2021.

[2] Eric Thorn, Shawn C. Kimmel, and Michelle Chaka. A framework for automated driving system testable cases and scenarios. 2018.

[3] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[4] Oracle Corporation. 5.1 introduction to spatial concepts. `https://docs.oracle.com/cd/E15817_01/appdev.111/e05682/sdo_intro.htm`, 2025.

[5] Geon Hwan Bae and Seon Bong Lee. A study on the evaluation method of highway driving assist system using monocular camera. *Applied Sciences*, 10.

[6] Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. Society for Industrial and Applied Mathematics, 2009.

[7] Preventable. In a rush at a school zone? seriously. `https://www.preventable.ca/in-a-rush-at-a-school-zone-seriously/` (Accessed: 2025-03-26).

[8] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access*, 8:58443–58469, 2020.

[9] S. Devi, P. Malarvezhi, R. Dayana, and K. Vadivukkarasi. A comprehensive survey on autonomous driving cars: A perspective view. *Wirel. Pers. Commun.*, 114(3):2121–2133, oct 2020.

[10] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[11] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[12] International Organization for Standardization. Iso 26262:2018 road vehicles — functional safety, 2018. Available at: `https://www.iso.org/standard/68383.html`.

[13] International Organization for Standardization. Iso/pas 21448:2022 road vehicles — safety of the intended functionality, 2022. Available at: `https://www.iso.org/standard/70939.html`.

[14] Xia Cai and Michael R. Lyu. The effect of code coverage on fault detection under different testing profiles. *SIGSOFT Softw. Eng. Notes*, 30(4):1–7, may 2005.

[15] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[16] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*, pages 120–131, 2018.

[17] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. Structural coverage criteria for neural networks could be misleading. In *2019 IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 89–92, 2019.

[18] Yizhen Dong, Peixin Zhang, Jingyi Wang, Shuang Liu, Jun Sun, Jianye Hao, Xinyu Wang, Li Wang, Jinsong Dong, and Ting Dai. An empirical study on correlation between coverage and robustness for deep neural networks. In *2020 25th International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 73–82, 2020.

[19] Federal Ministry for Economic Affairs and Energy of Germany. Home-pegasus-en.

[20] Mark Everingham and John Winn. The pascal visual object classes challenge 2012 (voc2012) development kit. *Pattern Anal. Stat. Model. Comput. Learn., Tech. Rep*, 2007:1–45, 2012.

[21] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[23] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[24] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[25] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[27] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[28] Laith Alzubaidi, Jinglan Zhang, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, José Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74, 2021.

[29] João Simões Farinha. In-vehicle object detection with yolo algorithm. Master's thesis, Universidade do Minho (Portugal), 2018.

[30] Ieee recommended practice for software requirements specifications. *IEEE Std 830-1998*, pages 1–40, 1998.

[31] Iso/iec/ieee international standard - systems and software engineering – life cycle processes –requirements engineering. *ISO/IEC/IEEE 29148:2011(E)*, pages 1–94, 2011.

[32] J Michael Spivey and @inproceedingsdeng2009imagenet, title=Imagenet: A large-scale hierarchical image database, author=Deng, Jia and Dong, Wei and Socher, Richard and Li, Li-Jia and Li, Kai and Fei-Fei, Li, booktitle=2009 IEEE conference on computer vision and pattern recognition, pages=248–255, year=2009, organization=Ieee Jean-Raymond Abrial. *The Z notation*, volume 29. Prentice Hall Hemel Hempstead, 1992.

[33] Nico Plat and Peter Gorm Larsen. An overview of the iso/vdm-sl standard. *ACM Sigplan Notices*, 27(8):76–82, 1992.

[34] Max J Egenhofer. A formal definition of binary topological relationships. In *Foundations of Data Organization and Algorithms: 3rd International Conference, FODO 1989 Paris, France, June 21–23, 1989 Proceedings 3*, pages 457–472. Springer, 1989.

[35] Sisi Zlatanova. Topological relationships and their use. In *Encyclopedia of GIS*, pages 1–21. Springer, 2016.

[36] Aditya P. Mathur. *Foundations of Software Testing: Fundamental Algorithms and Techniques*. Pearson Education India, 2013.

[37] Glenford J Myers, Corey Sandler, and Tom Badgett. *The art of software testing*. John Wiley & Sons, 2011.

[38] John Joseph Chilenski and Steven P Miller. Applicability of modified condition/decision coverage to software testing. *Software Engineering Journal*, 9(5):193–200, 1994.

[39] John Chilenski and LA Richey. Definition for a masking form of modified condition decision coverage (mcdc). *Boeing Report*, 1997.

[40] Sergiy A Vilkomir and Jonathan P Bowen. Reinforced condition/decision coverage (rc/dc): A new criterion for software testing. In *International Conference of B and Z Users*, pages 291–308. Springer, 2002.

[41] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[42] Uda Takuma and Tanaka Kento. Formalization of bbsl with coq. `https://github.com/utatatata/bbsl-coq`, 2022.

[43] Kento Tanaka. Annotated vehicle dataset for bbsl specification-based testing in autonomous driving systems (modified from kitti). zenodo.13910002, 10 2024.

[44] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.

[45] Glenn Jocher, Ayush Chaurasia, Jirka Borovec, NanoCode012, Christopher Harris, Alex Wong, Nikhil Mankrani, Linghao Zhang, Geng Yifeng, V Abhiram, Laughing, tkianai, Piotr Skalski, and Hieu Pham. ultralytics: YOLOv8. `https://github.com/ultralytics/ultralytics`, 2023.

[46] An Deng. Pytorch-yolov3-kitti. `https://github.com/packyan/PyTorch-YOLOv3-kitti`, 2018.

[47] Didi Ruhyadi. Vehicle detection with yolov8. `https://github.com/ruhyadi/vehicle-detection-yolov8`, 2023.

[48] Kento Tanaka. Bbsl testing tools. `https://github.com/IOKENTOI/BBSL-test`, 2023.

[49] You Li and Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[50] Adam Chlipala. *Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant*. MIT Press, 2022.

[51] Gilles Dowek. Non deterministic computation over the real numbers. *Submited to publication*, 2013.

[52] Adam Chlipala and George C Necula. Cooperative integration of an interactive proof assistant and an automated prover. In *Proceedings of the 6th International Workshop on Strategies in Automated Deduction*, 2006.

[53] Robert John Allen. *A formal approach to software architecture*. Carnegie Mellon University, 1997.

[54] J.P. Bowen. Formal specification in z as a design and documentation tool. In *Second IEE/BCS Conference: Software Engineering, 1988 Software Engineering 88.*, pages 164–168, 1988.

[55] Jonathan P. Bowen. Formal specification and documentation of microprocessor instruction sets. *Microprocessing and Microprogramming*, 21(1):223–230, 1987. Microcomputers: Usage, Methods and Structures.

[56] Peter T Breuer and Jonathan P Bowen. Towards correct executable semantics for z. In *Z User Workshop, Cambridge 1994: Proceedings of the Eighth Z User Meeting, Cambridge 29–30 June 1994*, pages 185–209. Springer, 1994.

[57] J. M. Spivey. *The Z notation: a reference manual*. Prentice-Hall, Inc., USA, 1989.

[58] J. M. Spivey. The fuzz type-checker for z. Technical report, University of Oxford, UK, 2008.

[59] Cliff B Jones. *Systematic software development using VDM*. Prentice-Hall International London, 1986.

[60] P. G. Larsen, B. S. Hansen, H. Brunn, N. Plat, H. Toetenel, D. J. Andrews, J. Dawes, G. Parkin, et al. Information technology – Programming languages, their environments and system software interfaces – Vienna Development Method – Specification Language – Part 1: Base language. Technical report, International Organization for Standardization (ISO), December 1996.

[61] John Fitzgerald, Peter Gorm Larsen, Paul Mukherjee, Nico Plat, and Marcel Verhoef. *Validated Designs For Object-oriented Systems*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2005.

[62] Marcel Verhoef, Peter Gorm Larsen, and Jozef Hooman. Modeling and validating distributed embedded real-time systems with vdm++. In *International Symposium on Formal Methods*, pages 147–162. Springer, 2006.

[63] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.

[64] Wengang Zhou, Houqiang Li, Yijuan Lu, and Qi Tian. Sift match verification by geometric coding for large-scale partial-duplicate web image search. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–18, 2013.

[65] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. *CoRR*, abs/1912.04838, 2019.

[66] Daniel Kondermann, Rahul Nair, Katrin Honauer, Karsten Krispin, Jonas Andrulis, Alexander Brock, Burkhard Güssefeld, Mohsen Rahimimoghaddam, Sabine Hofmann, Claus Brenner, and Bernd Jähne. The hci benchmark suite: Stereo and flow

ground truth with uncertainties for urban autonomous driving. In *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 19–28, 2016.

[67] Tsong Yueh Chen, S. C. Cheung, and Siu-Ming Yiu. Metamorphic testing: A new approach for generating next test cases. *CoRR*, abs/2002.12543, 2020.

[68] Jinyang Shao. Testing object detection for autonomous driving systems via 3d reconstruction. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 117–119, 2021.

[69] Zhi Quan Zhou and Liqun Sun. Metamorphic testing of driverless cars. *Commun. ACM*, 62(3):61–67, feb 2019.

[70] Manikandasriram Srinivasan Ramanagopal, Cyrus Anderson, Ram Vasudevan, and Matthew Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, oct 2018.

[71] Gilles Bernot, Marie Claude Gaudel, and Bruno Marre. Software testing based on formal specifications: a theory and a tool. *Software Engineering Journal*, 6(6):387–405, 1991.

[72] Richard Denney. Test-case generation from prolog-based specifications. *IEEE Software*, 8(2):49–57, 1991.

[73] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Structural test coverage criteria for deep neural networks. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 320–321, 2019.

[74] Junjie Chen, Ming Yan, Zan Wang, Yun Ah Kang, and Zhuo Wu. Deep neural network test coverage: How far are we? *ArXiv*, abs/2010.04946, 2020.

[75] Shinpei Kato, Shota Tokunaga, Yuya Maruyama, Seiya Maeda, Manato Hirabayashi, Yuki Kitsukawa, Abraham Monrroy, Tomohito Ando, Yusuke Fujii, and Takuya Azumi. Autoware on board: Enabling autonomous vehicles with embedded systems. In *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*, pages 287–296. IEEE, 2018.

[76] Autoware Foundation. Autoware architecture - node diagram. `https://autowarefoundation.github.io/autoware-documentation/main/design/autoware-architecture/node-diagram/`, 2025.

本研究に関する文献

[1] 田中健人, 青木利晃, 川上大介, 千田伸男, 河井達治, 冨田尭. 自動運転システムにおける画像を対象とした形式仕様記述言語 BBSL の提案. 研究報告ソフトウェア工学 (SE), 2020(8):1–8, 2020.(査読なし).

[2] Kento Tanaka, Toshiaki Aoki, Tatsuji Kawai, Takashi Tomita, Daisuke Kawakami, and Nobuo Chida. A Formal Specification Language Based on Positional Relationship Between Objects in Automated Driving Systems. In 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), pages 950–955, 2022.(査読あり).

[3] Kento Tanaka, Toshiaki Aoki, Tatsuji Kawai, Takashi Tomita, Daisuke Kawakami, and Nobuo Chida. Specification Based Testing of Object Detection for Automated Driving Systems via BBSL. In ENASE, pages 250–261, 2023.(査読あり).

[4] Kento Tanaka, Toshiaki Aoki, Takashi Tomita, Daisuke Kawakami, and Nobuo Chida. Specification-based Testing of the Image-recognition Performance of Automated Driving Systems. IEEE Access, 2024.(査読あり).