**進化型遠隔操作向け基盤技術開発**

1. **はじめに**

　　遠隔操作（Teleoperation）技術は、人間が危険な場所や手が届かない場所にアクセスし、操作することができるようになるための技術である。遠隔操作技術は、19世紀後半から20世紀初頭にかけて、主に無線通信技術の発展とともに始まった。Nikola Teslaは1898年にはじめて開発した無線で制御できる船が遠隔操作技術の基礎を築いたとされている（1）。当初は、軍事目的や危険な環境での作業を安全に行うために開発された。そのあと、コンピューター技術の発展とインターネットの普及により、遠隔操作ロボット技術は急速に進化し、より高度な制御が可能になり、様々な分野で活用されるようになった。また、高速な通信技術の進歩により、遠隔操作のレイテンシが大幅に短縮され、よりリアルタイムな操作が可能になった。

|  |
| --- |
|  |
| **Figure. 1 Schematic diagram of robotic teleoperation. Teleoperators communicate with mobile robots through communication channel such as satellite, radio links, and wired, and or wireless Internet connections. Teleoperators provide commands and receive feedback via a user interface.** |

　　遠隔操作技術は現在、多くの分野で活用されている。医療分野では、手術ロボットや内視鏡ロボットなどの遠隔操作ロボット技術を使用して、手術や診断を行っている。これにより、高い精度での手術や、患者への負担少ない診断が可能になっている。建築分野では、遠隔操作ロボット技術が高所作業や危険な環境下での作業に使用されている。例えば、遠隔操作クレーンや重機により、作業員が安全な場所から操作することができ、労働災害のリスクを軽減する。また、建物の点検やメンテナンス作業も、遠隔操作ドローンやクローラー型ロボットを用いることで効率化が図られている。農業分野では、遠隔操作ロボット技術が労働力不足の解消や生産性向上に寄与している。遠隔操作ロボットは、農地の耕作や収穫作業、植物の管理や害虫の監視を行うことができる。さらに、精密農業技術と組み合わせることで、資源の効率的な利用や環境への負荷軽減が期待されている。宇宙開発分野でも重要な役割を果たしている。国際宇宙ステーションにおいては、遠隔操作ロボットアームを利用して、宇宙船のドッキングや貨物の取り扱い、宇宙遊泳の支援などが行われている。

　　しかし、遠隔操作技術の発展とともに、以下のいくつか問題も出てきた。1） システムの複雑さ：　遠隔操作技術は、高度な技術・知識が必要であるため、遠隔地でも作業や操作に慣れていない人は、操作に失敗する可能性が高くなる。また、高齢化が進む社会においては、その普及に課題がある。2）　操作の精度や速度の制限：　現在の技術水準では、オペレーターの知覚情報や操作制御の精度に制限がある。たとえば、オペレーターが操作を行うためには、データを収集し、システムに反映するまでの時間がかかるので、オペレーターがリアルタイムで操作を行うことが困難な場合がある。また、手術ロボットは高精度な手術が可能だが、人間の手の感覚や判断力に比べると限界があるため、操作に必要な情報の表示や提示にもさまざまな課題がある。3） 安全性の確保：　遠隔操作においては、通信回線を介してシステムにアクセスするため、セキュリティの問題を考慮する必要がある。例えば、オペレーターが操作するシステムには、機密情報や重要な情報が含まれる場合がある。また、不正アクセスやハッキングなどの攻撃によって、システムが乗っ取られて、データが漏洩する可能性もある。さらに、遠隔操作技術を使用する場合、現地の法律や規制に従う必要がる。4） 遠隔地でのメンテナンス：　遠隔操作ロボット技術を使用する場合、オペレーターが遠隔地にいるため、ロボットのメンテナンスを行うことが困難である。例えば、遠隔操作技術のシステムには、カメラやセンサーなどのハードウェアとソフトウェアが含まれている。ハードウェアの故障が発生した場合、交換作業を行うためには、現地でのサポートが必要となる。また、ソフトウェアの拡張やアップデートを行うためには、遠隔地にいるオペレーターが、システムを一時停止してアップデートを行う必要がある。さらに、アップデートによってシステムが不安定になる可能性があるため、アップデートのタイミングや方法を慎重に検討する必要がある。

　　本開発では、上記の問題4「遠隔地でのメンテナンス」のソフトウェアアップデートに対し、モジュール更新機能を持つシステムを設計した。具体的には、システムにモジュール更新機能を実装し、遠隔地のメンテナンス担当者がアップデート用のモジュールをアップロードすることで、システムを更新することが可能となる。この遠隔操作システムより、メンテナンス担当者がどこでも、いつでも、ソフトウェアのアップデートが容易に行えるようになってきた。続いて、2章では要求分析を行い、ユースケースや要件定義が明確化される。3章ではシステムの設計が行われ、設計の概要、システム状態遷移図、実装の詳細が説明される。最後に4章では、システムの実装に対する検証が行われ、システムの正確性と信頼性が確認される。

1. **要求分析**

**2.1　ユースケース**

　　本開発において、荷物運搬ロボットのアーム運動における拘束機能について、操縦性を向上させることを検討された。荷物運搬ロボットアームは、工場や倉庫、物流センターなどで使用される自動化された機械であり、その主要な目的は、荷物や商品を運搬し、効率を向上させ、作業負荷を軽減することである。荷物運搬ロボットアームにおける運動拘束機能は、ロボットアームの動きや運動範囲を制御し、安全かつ効率的な荷物運搬を実現するための機能である。この機能は、遠隔地でのメンテナンスにおける動的ライブラリ更新機能と連携して、以下のようなユースケースを提供できる。

1. 運動範囲の制限：　ロボットアームの運動範囲を制限することで、荷物の安全な運搬を確保する。これにより、運搬中に荷物が落下するリスクや、周囲の環境に損傷を与えるリスクを最小限に抑えることができる。
2. オペレーターの操作負荷軽減：　運動拘束機能により、オペレーターはロボットアームをより簡単に制御できるようになる。
3. ソフトウェアアップデータによる機能強化：　遠隔地でのメンテナンスを行う際に、動的ライブラリ更新機能を用いて運動拘束機能をアップデートすることができる。これにより、新たな制御アルゴリズムや安全機能が追加されることで、運動拘束機能の性能が向上し、ロボットアームの操縦性がさらに向上する。
4. 状況に応じた運動拘束設定：　状況に応じて適切な運動拘束設定を選択することが可能となる。例えば、荷物の大きさや重量に応じて、運動範囲や速度を制限することができる。これにより、荷物の種類や状況に応じて最適な運搬が行える。さらに、ユーザーにとっても使いやすいシステムとなる。

**2.2　要件定義**

本遠隔操作システムでは、節2.1に示した各ユースケースに対し、遠隔操作技術を用いることで、システムの安全性と効率性の向上を目的とした更新対象機能の要求について検討し、以下の機能要件が必要と考えられる。ソフトウェアの部分更新機能を用いることで、ロボットシステムの柔軟性や安定性を向上させることを狙っている。運動拘束機能を有さない遠隔ロボットに対して、新たに運動拘束機能を配信して、ロボット側のS / Wの更新により機能性を向上させるための要件を定義している。具体的な要件は、以下のように挙げられる。

1. ロボットのソフトウェアの部分更新（更新失敗時のロールバック）
   1. 遠隔地にいる稼働中のロボットに、オペレーターから対象アプリのS / Wを指定して、S / Wを更新できること。
   2. オペレーターが前記S / W配信の正常に完了したことを確認できること。
   3. ロボットと通信できなくなった等、前記S / Wの配信に失敗した場合、前記S / Wの再配信できること。
   4. 更新失敗の時に、ロールバックを実行すること。ロールバックとは、前記S/Wの更新処理途中でうまくいかなかった場合、もしくは前記S/Wの無事に更新完了し、オペレーターが切り替えた後、異常が発生する場合に、更新前のS/Wを復帰できること。
2. 更新のチェック（ソフトの更新が正しく行えたかのPrecheck）
3. 運用中のロボット上に、システムの一時停止をしなくても、遠隔オペレーターによる現在S / WからアップデートしたS / Wへの切り替えを実施できること。
4. 前記切り替え処理中にロボットと疎通できなくなった場合、疎通再開時に切り替え前のS/Wで復帰できること。
5. 対象ロボットにおいて前記S/Wを削除できること。

これらの要件により、遠隔地からでも正確かつ安全にロボットのS / Wを更新することができ、運用中に問題が発生しても、迅速かつ的確に対応することができると考えらえる。

**3.　遠隔操作システムの設計**

**3.1　設計概要**

遠隔システムの設計において、まず、以上の要件定義を満たすために、システムのコンポーネント以下のように構成される。①オペレーター側のユーザーインタフェイス：　オペレーターが遠隔地にいる稼働中のロボットを管理・操縦するためのインタフェイスである。②通信モジュール：　オペレーターとロボット間の通信を担うモジュールである。③ロボットの管理モジュール：　ロボット側で動作するモジュールであり、オペレーターからの指示を受け取り、ロボットを管理・操縦するモジュールである。具体的なシステムの設計は以下のようになる。

オペレーターは、専用のユーザーインタフェイスを通じて、遠隔地にある稼働中のロボットに事前クラウドサーバーにアップロードしたS / Wモジュールをダウンロードすることを指示できる。このインタフェイスは、セキュアな通信プロトコルが使用されることで、信頼性と安全性の確保が期待されている。本システムは、MQTTプロトコルを使用している。モジュールのダウンロードが完了した後、オペレーターに完了シグナルを送信する機能を実装して、オペレーターはこのシグナルによって、モジュールのダウンロードが正常に完了したこと確認できる。そして、モジュールを適用するかどうかを判断し、適用する場合は、稼働中のロボットが一時停止することなく、モジュールが正常に動作できるかどうかを確認するためのPrecheck機能が望まれている。さらに、遠隔オペレーターによる現在モジュールからアップデータしたモジュールへの動的に切り替えを実施できることも期待されている。その要件を満たすため、本システムは、各更新対象となるモジュールが動的ライブラリを利用した。動的ライブラリは、実行時に必要に応じてロードされ、アプリによって使用される共有オブジェクトである。プログラムが起動する際に、必要な静的ライブラリとともに、OSが動的ライブラリをロードし、メモリにマップされる。動的ライブラリは、一度ロードされたら、再利用することができる。また、ライブラリの更新に伴い、再コンパイルの必要がないため、ライブラリの更新や切り替えも容易になる。本システムの動的ライブラリの実装詳細については、節3.3実装の詳細に説明する。さらに、アップデータしたモジュールへの切り替えを実施した後、システムの異常が発生した場合でも、切り替え前のモジュールで復帰したり、ロボットにおいてダウンロードしたモジュールを削除したりすることも本システムに実装した。

本遠隔システム設計の概要を説明したが、これだけでロボット側の管理モジュールの動作がまだ十分に理解できないため、次の節3.2に状態遷移図を用いてシステムの詳細を説明する。

**3.2　状態遷移**

|  |
| --- |
|  |
| **Figure 2. Transition diagram for a robot, consisting of 6 states: "Power OFF", "Power ON", "Downloading", "Waiting Precheck Request ", "Precheck", and "Error". The diagram shows the possible transitions between these states and the actions that trigger them. The system uses version control and rollback to minimize downtime and reduce the likelihood of recurring errors.** |

ロボットの状態遷移図はFig. 2に示したように、「Power OFF」、「Power ON」、「Downloading」、「Waiting Precheck Request」、「Precheck」、「Error」６つの状態が含まれている。ロボット最初は、「Power OFF」であり、この状態ではロボットは動作していない状態である。この状態から「Power On」の状態へしか遷移できない。「Power ON」状態は、ロボットが正常に動作している状態である。さらに、「Power On」状態は、「STOP」、「AUTO」、および「MANUAL」3つのサブ状態から構成されており、詳細について本報告に議論しない。「Power ON」状態から「Power OFF」、「Downloading」、「Error」のいずれかの状態へ遷移することができる。「Downloading」状態は、ロボットがネットワークからS / Wモジュールなどをダウンロードしている状態である。ロボットが「Power ON」の状態では、遠隔地にいるオペレーターから更新対象となるモジュールのダウンロード指示を受けることが可能であり、ロボット側がモジュールのダウンロードを開始すると、「Downloading」状態に遷移する。モジュールのダウンロード完了後、そのモジュールのHash値を計算して、データの完全性やセキュリティ性を確認する。計算したモジュールのハッシュ値をオペレーター側に送信して、ロボット側の状態を「Waiting Precheck Request」状態に遷移する。「Waiting Precheck Request」状態は、ロボットが正常に稼働しており、オペレーター側から動作確認の指示を待っている状態である。オペレーター側が受け取ったHash値を用いて、ロボットにダウンロードされたモジュールの完全性とセキュリティ性を確認した後、異常が確認できなければ、Precheckの要求をロボット側に送信する。「Waiting Precheck Request」から「Precheck」状態への遷移は、オペレーターによってPrecheck要求が発行され、ロボット側が要求を受信すると実行される。「Precheck」状態は、ロボットが稼働している状態で、ダウンロードしたモジュールが正しく動作するかどうかを確認するための作業である。この作業では、実機ロボット上でロボットの現動作に影響せず、新しい配置されたS / Wモジュールの試運転を行い、試運転から得られたデータ情報がオペレーターに送信し、その情報に基づいてオペレーターがモジュールの動作を確認することができる。Precheckが実行されると、実機ロボット上で新たな配置されたモジュールが動作し始めて、問題がある場合、オペレーター側にエラーメッセージを送信する。一方、モジュールが正常に動作する場合は、正常に完了した旨のメッセージがオペレーター側に送信され、ロボットが再び「稼働中」に遷移する。このようにして、オペレーターがPrecheckの結果を確認し、新たなモジュールへの切り替えを実行するかどうかを判断する。新しいモジュールへ切り替えた後、システムが予定どおり動いていなかったり、異常が発生したりするパターンがよくある。その際に、本システムでは、異常などが発生すると、ロールバックを自動的に実行することができる。このロールバック処理によって、ロボットの運用停止時間を最小限に抑えることができる。

**3.3　実装の詳細**

遠隔操作システムのモジュール更新機能を実際に検証するため、C++17をプログラミング言語として使用し、ROS（Robot Operating System）のTurtlesimパッケージを用いて、システムを構築した。Turetlesimは、ROSの教育用シミュレータで、ロボットの制御やアルゴリズムの検証によく利用されている。本遠隔システムのキーポイントは、S / Wモジュールの更新が動的に実行するため、更新対象となるモジュールが動的ライブラリを用いて、Turtlesimロボットを動かさせる。この部分の詳細については、4.1節に説明する。Turtlesimロボットを動かすためのメインスレッドと同時に、複数のサブスレッドも実行させた。最初のサブスレッドは、オペレーターからネットワークから新しいモジュールのダウンロード指示を送ると、サブスレッドを実行し、ダウンロードが完了後、サブスレッドを終了する。2つ目のサブスレッドは、更新対象となるS / Wモジュールの管理システムのフォルダー（以降、フォルダーと呼ぶ）のタイムスタンプを常時チェックし、フォルダーの「更新日時」が更新されると、新しいS / Wモジュールがフォルダー内に配置されたどうかを確認する。新しいモジュールが配置された場合、オペレーターからPrecheckの要求をロボット側に送信すると、3つ目のサブスレッドを実行する。このサブスレッドでは、新しいモジュールに対してPrecheckを行い、得られた情報をMQTT経由でオペレーター側に送信する。Precheckが終わると、3つ目のサブスレッドの実行を終了し、メインスレッドでTurtlesimロボットが新しいモジュールへの切り替えが可能となる。

|  |
| --- |
| テキスト  自動的に生成された説明**(*A*)** |
| **テキスト  自動的に生成された説明 (*B*)** |
| **Figure. 5 An example implementation code for dynamic libraries in C++ (A) and Python (B).** |

本システムのキーポイントは、Fig. 5Aに示したように、対象となるモジュールをライブラリ化して、Unix系システムで動的リンクをサポートするための”dlfcn.h”ヘッダーファイルを利用した。動的リンクは、プログラムが実行される際に必要な関数や変数などを、プログラムの実行中にリンクすることを指す。” dlfcn.h” は、Unix系OSでよく利用されており、動的リンクにより、アプリケーションの起動時間が短くなったり、メモリの使用効率が向上したりするなどのメリットがある。” dlfcn.h”は、動的リンクに関するいくつかの関数を提供している。dlopen()関数は、指定された共有ライブラリをメモリーにロードし、ロードされたライブラリのハンドルを返す。dlsym()関数は、ロード済みの共有ライブラリに対して、指定されたシンボル名（変数や関数）に対応するポインタを取得することによって、プログラムから共有ライブラリ内に定義された関数を呼び出すことができる。

Fig. 5Aに示したC++具体例は、loading\_tuyrtlesim\_lib()関数が、Turtlesimの共有ライブライをロードし、それを実行するための関数である。まず、ライブラリのパスを“lib\_path”に格納し、dlopen（）により、共有ライブラリをロードし、ハンドルを取得する。ライブラリが正常にロードされたら、dlsym()関数によって、\_Z9RunTurtlevという名前の関数（Turtlesimの実行関数）の取得し、reinterpret\_cast()で、そのアドレスをvoid(\*)()型の関数ポインタMoveturtleに変換する。そして、Moveturtle（）を呼び出すことで、共有ライブライ内のTurtlesim実行関数が実行された。最後に、dlclose（）を使って、共有ライブラリのハンドルを閉じてリソースを開放する。これで、動的ライブラリをロードしてTurtlesimの実行関数を呼び出すプロセスが完了した。この動的共有ライブラリ方法は、モジュール化されたシステムにおいて、Turtlesimの実行にかかわる部分を外部の共有ライブラリに切り分けることで、柔軟性とメンテナンス性を向上させる目的で使用されると考えられる。そして、Fig. 5Bに示したようにPythonのimportlibライブラリを使用することで、モジュールを動的にリンクすることができる。ここでは、Pythonの実装について省略した。

**4.　検証**

　　本研究では、ROS Turtlesimシミュレータ上で設計した遠隔操作システムを使用し、動的ライブラリの更新が実行できるかどうかを検証した。以下に、検証方法と結果について説明する。

検証方法：　①Turtlesimロボットが稼働中の状態に動的ライブラリが正しくダウンロードされることを確認する。②ダウロードした動的ライブラリに対して事前チェック機能を実行し、正常に動作するライブラリと異常を発生するライブラリ両方の試験を行った。③事前チェックに成功した動的ライブラリがシステムに適用されたため、ロボットの動作に反映されるかどうかの検証を行った。この検証では、ロボットがアップデートされたライブラリへ切り替えて正常に動作するか、および問題が発生した場合にシステムが適切に対応できることを確認した。

検証結果： 上記の検証方法① - ③に従い、本開発で設計・実装した遠隔操作型ロボットシステムのモジュール更新機能に対して検証を行った。検証した結果は、S / Wが正しくダウンロードされ、Precheck機能によって正常なライブラリと異常なライブラリの両方を検出できた。また、Precheckに成功したモジュールがシステムに適用された後、ロボットの動作に反映されることも確認できた。これにより、システムは迅速に新しい機能を取得・適用し、問題が発生した場合にも対処が容易になる。また、システムの柔軟性と拡張性が向上し、将来的な開発や改善が容易に実施できることが示された。

**5．まとめ**

本研究では、遠隔操作技術を用いた荷物運搬ロボットのシステムに対し、安全性と効率性の向上を目的としたS /　Wモジュール更新機能の開発を目指し、ROS Turtlesimシミュレータ上で遠隔操作ロボットシステムの設計と実装を行った。本研究のキーポイントは、遠隔操作システムにおけるS / Wモジュールが安全に更新するために、アップデートされる対象となるモジュールのPrecheckとロールバックプロセスに関する仕組みを検討した。実装および検証の結果、本システムは正確かつ効率的に動的ライブラリの更新を行い、問題が発生した場合でも適切に対応できたことを確認した。これにより、システムは迅速に新しい機能を取得・適用し、将来的な開発や改善が容易に実施できることが示された。

　　今後の課題として、遠隔地でのソフトウェアアップデートには、セキュリティ上のリスクがあるため、アップデートの際に、十分なセキュリティ対策を講じる必要がある。これらの問題に対して、今後の改善が求められると考えられる。