GUI要素の階層構造構築・比較による視覚的回帰テスト差分検出方法の実証評価

研究コース5 GUI-Testチーム

研究員　：塚越 章弘（株式会社日本科学技術研修所)
主査　　：石川 冬樹（国立情報学研究所）
副主査　：栗田 太郎（ソニー株式会社）
　　徳本 晋（株式会社富士通研究所）

2021年2月26日

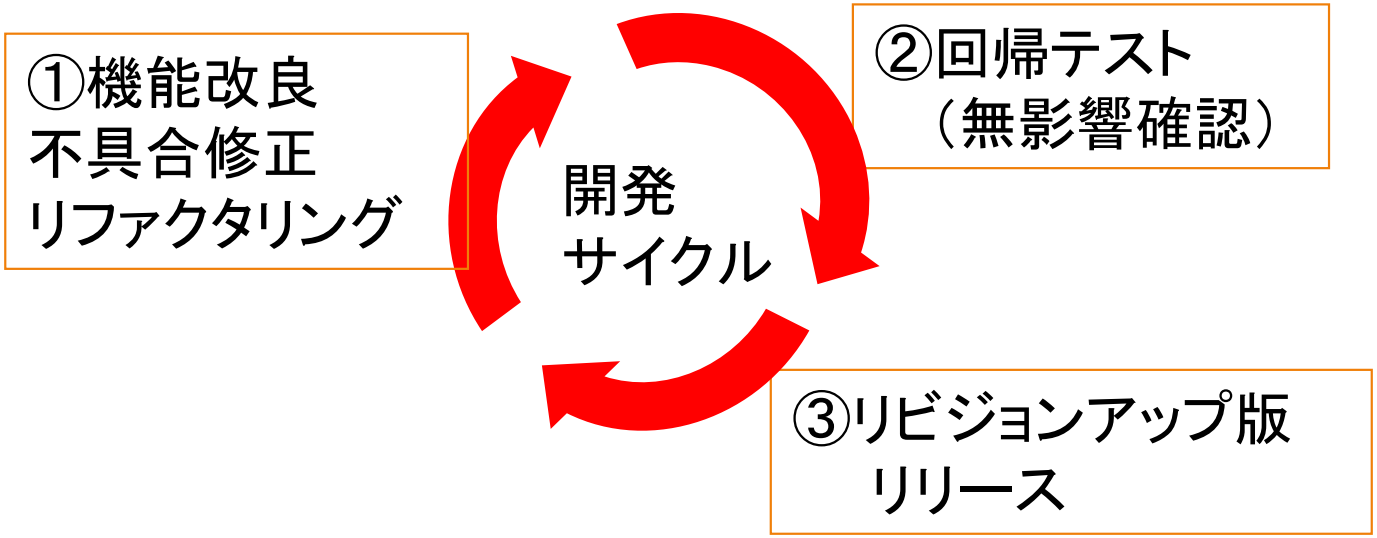
◆はじめに･研究の背景

◆GUI回帰テスト全体の概要

◆GUI回帰テスト 無影響確認

はじめに・研究の背景

・アプリケーションソフトの機能改良・不具合修正のため
短期間にリビジョンアップを行ったり、
保守性を高めるためにソースコードのリファクタリングを行ったりすることがある
→ 回帰テストによる、無影響確認は欠かせない
→ 無影響を確認することで、安心して修正できる！

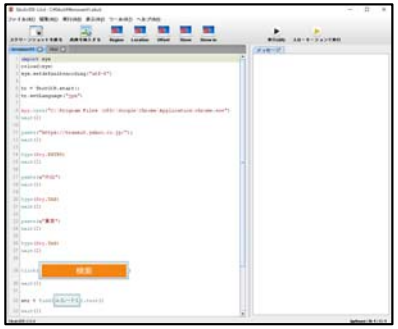


・改良が入った新機能部分は手作業でテストせざるを得ないが、
無影響確認はなるべく自動化で済ませたい（効率良くテストをこなしたい）

・Web系のアプリはDOMというデータ構造を持っているため、自動テスト化しやすいが、
GUIアプリはそのような構造を持たない為、自動テスト化しにくい

GUI回帰テスト全体の概要

・スクリプト作成



スクリプト

GUIを動かす元

・スクリプトでアプリを
自動実行させて，
画面キャプチャ

アプリVer1

確認したい
画面

・スクリプトでアプリを
自動実行させて，
画面キャプチャ

アプリVer2

確認したい
画面

アプリVer3

…

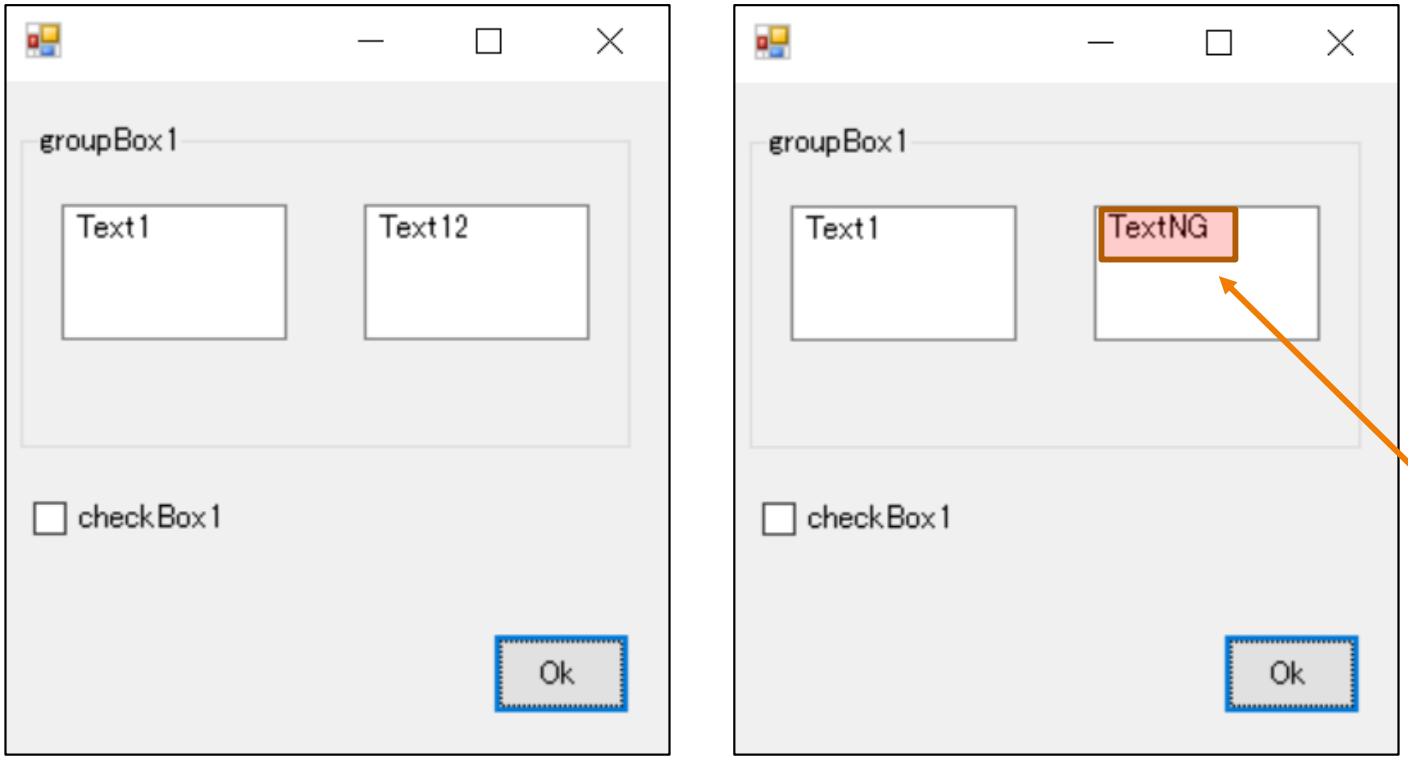
…

視覚的回帰テスト（VRT:Visual Regression Test）
による**画像差分比較**によって，無影響確認を行う

本研究では、画像の差分比較に焦点を当てている

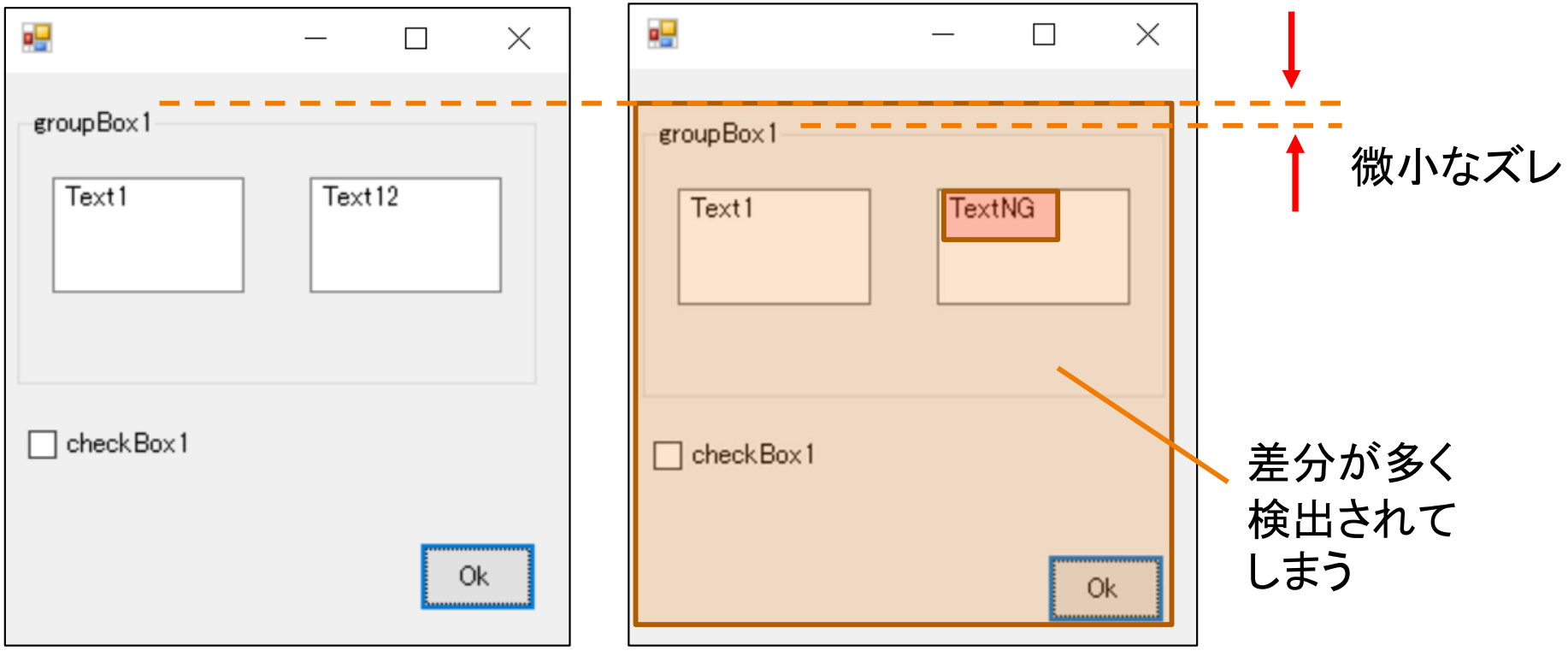
GUI回帰テスト 無影響確認

画面内にズレが入り込んでいない場合、単純な差分比較により
不具合個所を検知できる



GUI回帰テスト 無影響確認

GUIアプリ画面の要素に、微小なズレが存在した場合でも
差分比較できるようにしたい



要素が全体的に下にズレている場合、差分が多く発生してしまう。
（本来検出したい不具合個所を見逃してしまう）

GUI回帰テスト 無影響確認

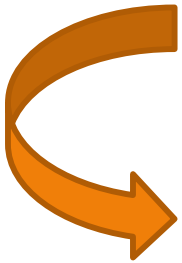
微小なズレが入り込む要因

1. 画面キャプチャ取得時の精度によるズレ

2. GUI要素を動的に配置している領域（動的領域）のズレ

3. データベース連携やレジストリを多数扱うアプリケーションの場合、
   実行時の状態が異なることに起因するズレ
   （実行環境の状態を毎回同じ状態に設定することは、コストが掛かる）

4. Windows UpdateやライブラリのUpdate等により
   外観が微妙に変化してしまうことに起因するズレ



微小なズレが存在している場合でも、
本来検出したい不具合個所を
特定できるようにしたい

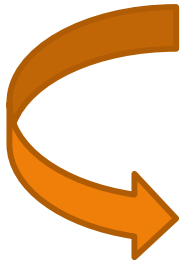
◆解決方法を模索

◆差分検出の処理概要

◆処理の詳細

解決方法を模索

微小なズレが存在している場合でも、
本来検出したい不具合個所を
特定できるようにしたい



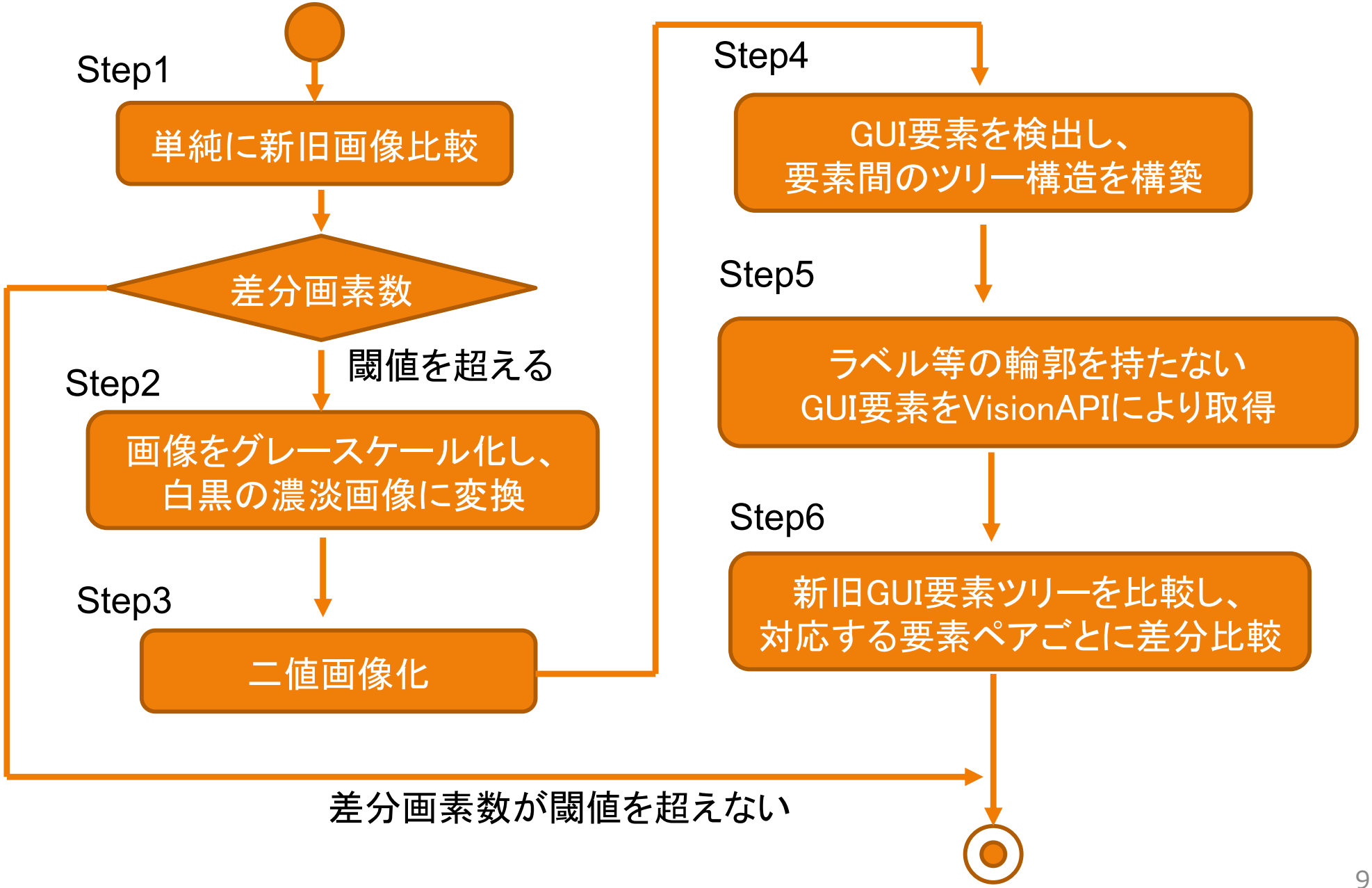
・先行論文(* ReBDiff)から実装ヒントを得る
　→　画像内のGUI要素をエッジ検出して、
　　対応する要素ごとに差分比較を行う

・画像を扱うライブラリとして OpenCV を利用
　→　GUI要素のエッジ，階層構造を取得

・ Google Cloud Vision APIを利用
　→　ラベルのように輪郭を持たないGUI要素を検出

(*) **Region-based Detection** of Essential Differences in Image-based Visual Regression Testing
ReBDiff

差分検出の処理概要



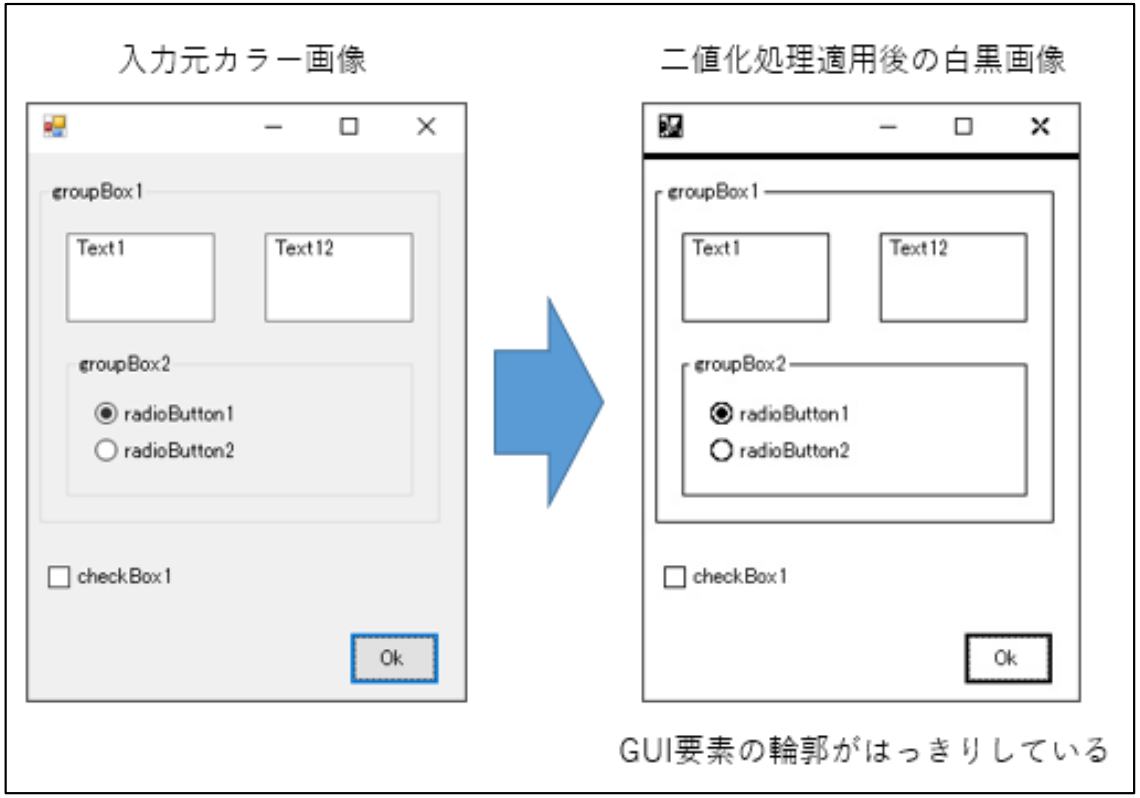
処理の詳細

Step3　二値画像化

ある画素値が閾値よりも大きければ白，そうでなければ黒のように処理
→　オブジェクトの輪郭を捉えることが出来るようになる

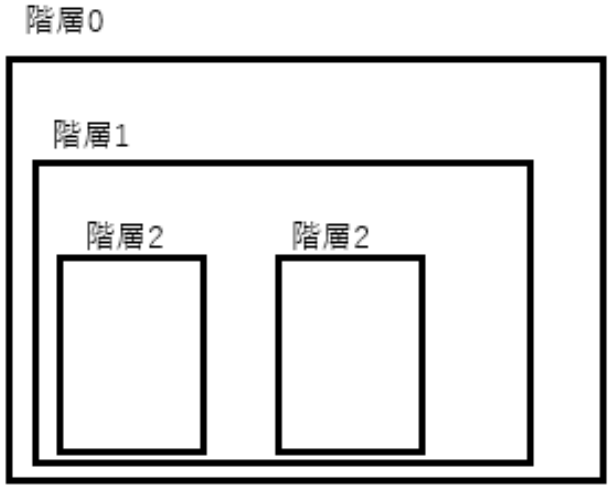
適応的二値化処理という，
全体的な閾値を固定せずに，
注目画素とその周囲にある
画素の平均値を閾値
とする方法を採用.

GUI要素には様々なパターンの
配色が設定される可能性が
あるため



処理の詳細

Step4　GUI要素のツリー構造構築　（OpenCVを活用）



GUI要素の階層表現

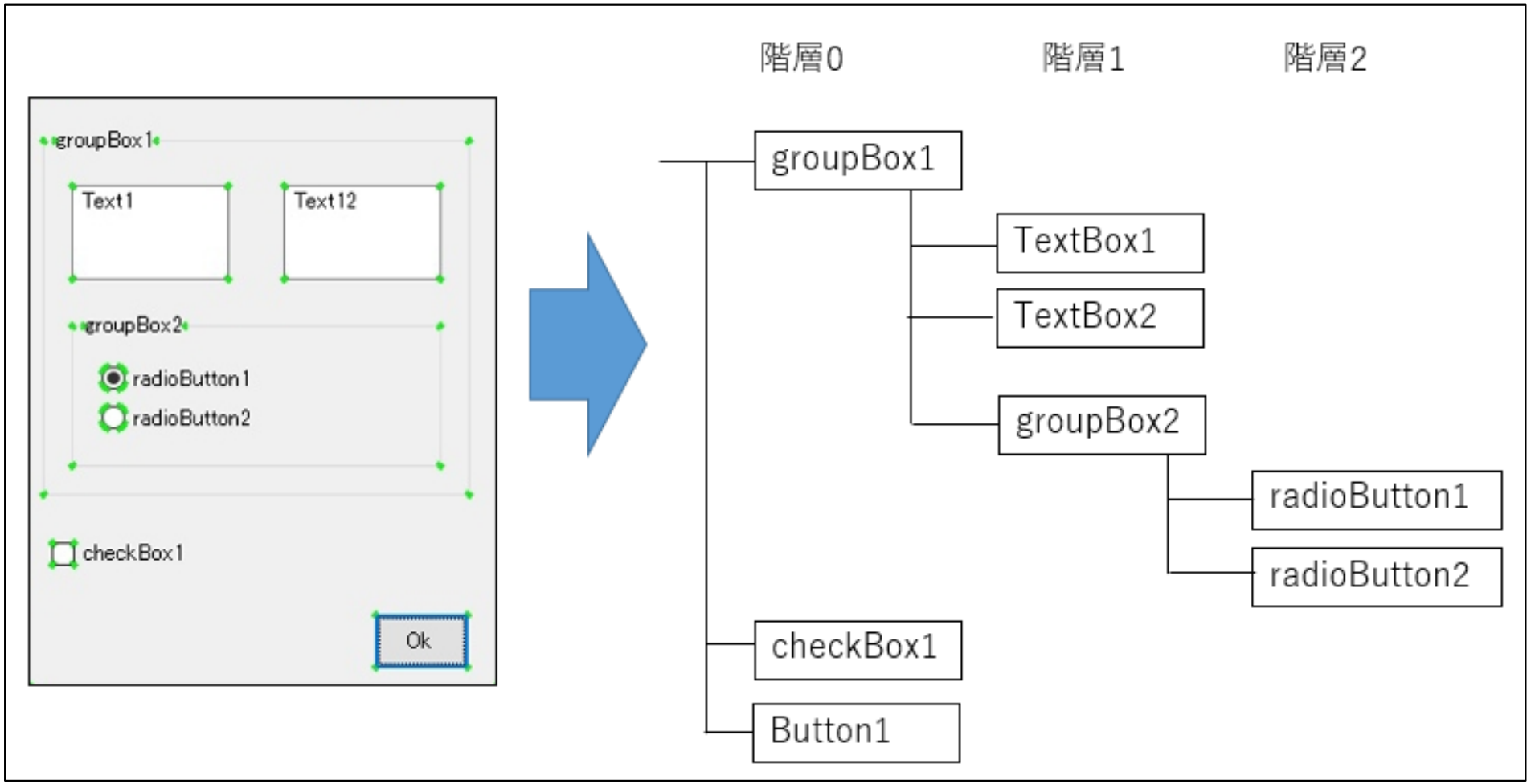
一番外側の輪郭が階層0,
その一つ内側に含まれる輪郭が階層1,
さらにその一つ内側に含まれる輪郭が
階層2のようになる.

階層0と階層1は親子関係となっており,
階層0に位置する要素が親要素,
階層1に位置する要素が子要素となる.
その配下の階層も同様に親子関係を持つ.

フラットにGUI要素の比較を行うよりも,
階層構造を持たせることで，差分個所を特定しやすくなる

処理の詳細

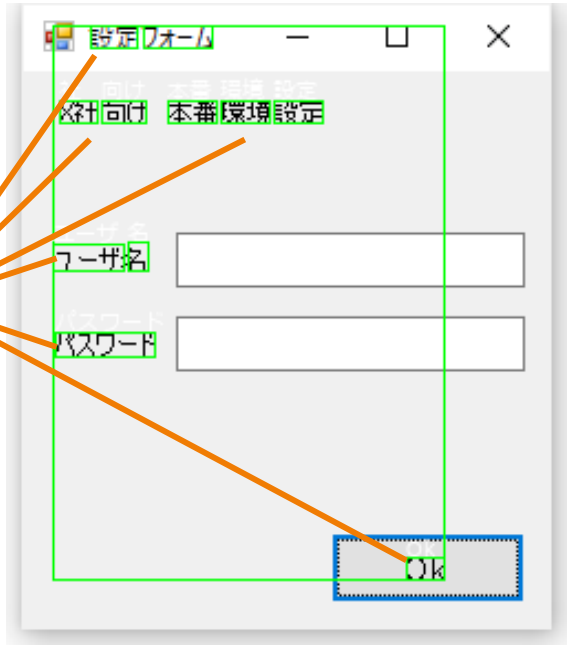
Step4　GUI要素のツリー構造構築　（OpenCVを活用）



処理の詳細

Step5　ラベル等の輪郭を持たないGUI要素をVisionAPIにより取得

テキストを含む
領域を検出



VisionAPIの事前トレーニング済み機械学習モデルを利用して、
オブジェクト（GUI要素）を検出し、
・対象矩形領域（座標，面積）
・文字列情報
を取得することが出来る

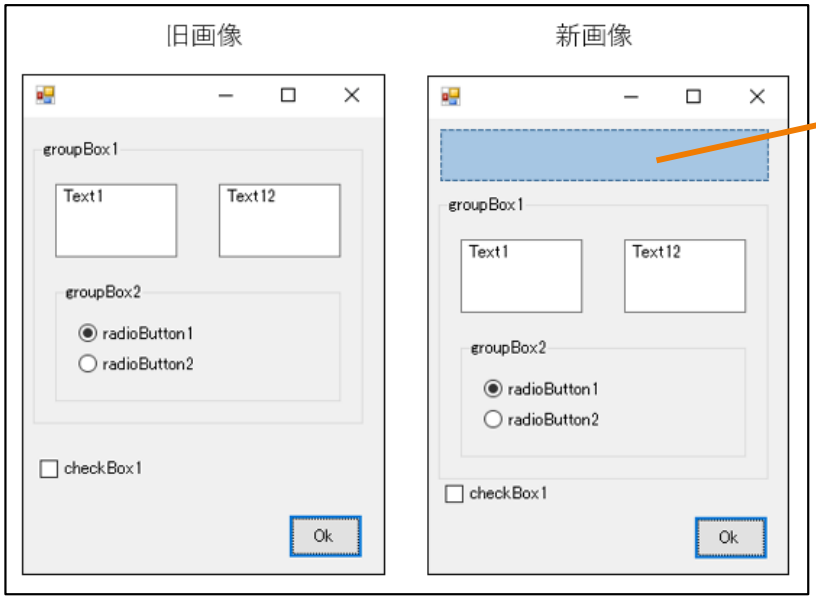
◆検証方法

◆検証結果

◆考察とまとめ

◆今後の課題

検証方法



ブランク領域
（微小なズレ）

テストケースとして，代表的なGUI要素を使用して，
GUI要素間の位置関係に微小なズレのみが
存在する場合を想定したテスト対象画面を
用意して評価を実施

評価ポイント
・評価項目A：画面内の全GUI要素を検出できているか
・評価項目B：取得したGUI要素によるツリーが適切な階層構造で構築できているか
・評価項目C：対応するGUI要素間で適切に差分比較が行われているか

検証結果

評価ポイント
・評価項目A：画面内の全GUI要素を検出できているか
・評価項目B：取得したGUI要素によるツリーが適切な階層構造で構築できているか
・評価項目C：対応するGUI要素間で適切に差分比較が行われているか

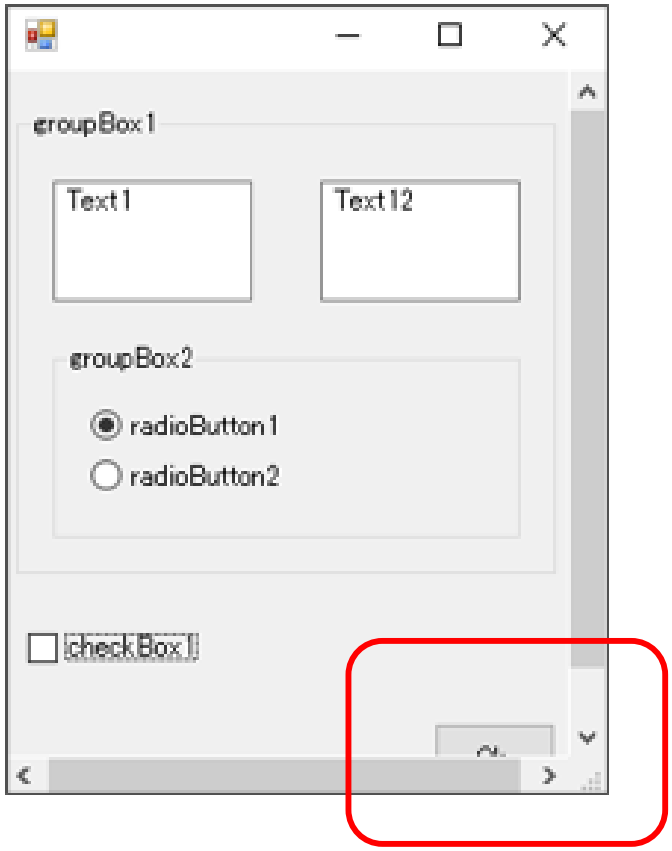
評価結果

No	テスト対象画面	評価項目A	評価項目B	評価項目C
1	1つ以上のButtonが存在	○	○	○
2	1つ以上のCheckBoxが存在	○	○	○
3	1つ以上のComboBoxが存在	○	○	○
4	1つ以上のListBoxが存在	○	○	○
5	1つ以上のScrollBarが存在	○	○	×
6	1つ以上のLabelが存在	○	○	○
7	1つ以上のGroupBoxが存在	○	○	○
8	1つ以上のTextBoxが存在	○	○	○
9	1つ以上のRadioButtonが存在	○	○	○

No5のScrollBar以外は成功

検証結果

成功しなかった、スクロールバーを含む例



GUI要素の一部が
欠落表示されてしまっている

考察とまとめ

1. ScrollBarのように、GUI要素欠落が発生するパターンについては，
   画面キャプチャ時に全要素が表示されるようにする工夫等を入れないと，
   それ以降の対応は難しいと思われる

2. GUI要素検出の際、画像内文字列の一部や模様など，
   細かいオブジェクトとして検出してしまうことが見受けられた．

→ ある閾値よりも面積が小さいオブジェクトは検出対象外となるようにしたところ，
問題なくGUI要素の輪郭を検出し，差分比較が行えるようになった．

文字「B」の囲み部分が
輪郭検出された例

考察とまとめ

3. デフォルト設定よりも境界を太くした状態で階層構造を構築しようとした際，
境界線の外側と内側に複数の輪郭を検出し，冗長な階層作成が見受けられた．

→ この状態でも問題なく差分比較を実施できるものの，冗長な出力結果となる．
冗長構成を解消するには，ほぼ同一位置に存在し，
面積もほぼ同じ要素は同一視するような処理等が必要になると思われる．

4. 「微小なズレ」が対象プログラムの仕様によっては
レイアウトの不具合となっている可能性もある．

→ 比較を行う際の外部パラメータとして，どの程度の距離までを微小と見なすか
といった閾値設定を設ける等の工夫が必要と思われる

今後の課題

・本研究での差分比較方法を実際の運用フェーズに組み込み，
実用に耐えられるものかをさらに検証する必要がある

・解決策として使用したデータ数はまだ十分とは言えない為，
より多くのテストパターンで検証を積み重ねていく必要がある

・テスト自動化では大量のテストが一気に実行される．
そのため，大量の差分比較結果をテスターが確認する際，
効率よく把握できるように，レポート出力も工夫が必要に
なると考えられる

ご清聴ありがとうございました