Vol.2021-MUS-130 No.14
Vol.2021-EC-59 No.14
2021/3/16

SAGUS: 顕著性マップを用いた間違い探し自動生成の提案と実装

陶山 晴南[1,a)]　青山 泰志[1]　土屋 慶吾[1]　髙木 亜蘭[2]　濱川 礼[1]

概要：本論文では，入力画像から顕著性マップを用いて間違い探しを自動で生成する手法と，その手法を基に実装したシステム「SAGUS」について述べる．間違い探しは，ルールが単純であるため子供から大人まで楽しめるゲームである．しかし一度問題を解いてしまうと，書き込みをしてしまったり間違いのある箇所をある程度覚えてしまい，同じ問題を繰り返し解くことができない．また既存の間違い探しの問題は 1 題材につき間違い画像が 1 問しかなく，同じ題材で異なる問題を楽しむことができない．「SAGUS」では入力した 1 枚の画像から複数の物体を抽出し，抽出した物体の中から顕著性マップを用いて間違いの決定と問題の生成を行うことで，1 組の間違い探しを自動生成する．顕著性マップとは，画像の中で人間が注視しやすい場所を 1 枚の画像として視覚化する画像表現である．1 つの物体に対し間違い生成前後の物体の顕著性マップの値を求め，その値の差を基に画像中に生成する間違いの位置を決定する．また間違いの個数はユーザが選択することができ，間違いの種類は「削除」,「追加」,「色変化」の 3 つからランダムで決定するため，同じ画像・同じ間違いの個数を選択しても毎回異なる問題を生成することができる．それに加え難易度を「易しい」,「普通」,「難しい」の 3 つから選択することができ，1 枚の画像で繰り返し異なる問題を解くことのできる，エンタテインメント性に富んだシステム「SAGUS」を開発した．

1. 背景と目的

簡単に遊べるゲームに間違い探しがある．間違い探しは 1 組の画像から間違い箇所を探し出すパズルゲームであり，ルールが単純なことから子供から大人まで楽しめるものになっている．実際全国チェーン店のファミリーレストラン，サイゼリヤでは間違い探しの問題が置かれている [1]．その問題数は間違い探しが導入された 2006 年から数えて 20 題以上あり利用者に長く楽しまれていることがわかる．また間違い探しには人間の視知覚を刺激して脳を働かせる効果があり，物忘れなどの記憶力の低下や認知機能の低下を防ぐ効果がある．それに加え，2 つの画像を交互に見比べることで注意力や集中力を養うこともできる [2][3]．

しかし間違い探しの問題を一度解いてしまうと，問題に書き込みをしてしまったり間違いのある箇所をある程度覚えてしまう．そのため同じ問題を繰り返し解くことが難しい．また既存の間違い探しの多くは 1 題につき間違い画像は 1 問しかなく，同じ題材で異なる問題を解くことができない．

そこで本論文では，既存の間違い探しに使用されている画像やユーザが撮った写真等任意の画像 1 枚に対し難易度別に問題が作成でき，同じ入力画像・同じ難易度を選択しても異なる問題を作成し繰り返し遊べることを目的としたシステム「SAGUS (*Spot-the-difference Automatic Generator Using Saliency*)」を開発した．

2. 関連研究

2.1 間違い生成

間違いの生成に関する研究として Jin らの研究 [4] がある．Jin らは，ユーザが選択した入力画像内の領域に対し自動で間違いを生成し，生成した間違いの難易度を評価することの出来るシステムを提案した．ユーザはあらかじめ抽出された領域に対して，間違いを生成する領域，間違いを生成しない領域に分割する操作を行う．間違いを生成する領域では間違いが自動で生成される．

この研究ではユーザが間違い生成に関する操作を行う必要があり，生成される間違い箇所が分かってしまう．そのため，ユーザは生成された間違い探しを楽しむことが出来ない．「SAGUS」ではユーザが間違い生成に関する操作を行わないため間違い探しを楽しむことが出来る点で異なる．

1 中京大学工学部情報工学科
2 中京大学大学院工学研究科情報工学専攻
a) horn31music@gmail.com

Vol.2021-MUS-130 No.14
Vol.2021-EC-59 No.14
2021/3/16

表 1　各種類の間違いの個数

種類	削除	追加	色変化	大きさ変化	状態変化	変形	置換
個数	772	596	253	207	171	151	50

2.2　間違い探しゲーム

間違い探しゲームの生成に関する研究として Park らの研究 [5]，Liu らの研究 [6] がある．Park らは，ユーザが選択した入力画像内の領域に対し自動で間違いを生成し，間違い探しゲームコンテンツを生成するシステムを提案した．間違いは選択された領域ごとに複数パターン生成され，ユーザが各領域ごとに最適な間違いを選択することで一つの間違いが生成される．

この研究ではユーザは間違いを生成する領域や各領域ごとに生成する間違いを選択する必要がある．また，間違い探しの難易度は時間で設定され，生成した間違いの面積に応じて制限時間が変化する．「SAGUS」ではユーザが間違いを生成する領域を選択する必要がない点や，間違い探しの難易度を単純に面積と時間ではなく後述する顕著性マップを用いて設定している点で異なる．

3.　提案手法

本章では，「SAGUS」において扱う間違いの種類を決定するための調査と，1 枚の画像から間違い探しを生成するための手法について述べる．「SAGUS」はユーザが用意した 1 枚の画像を題材とし，ユーザが指定した間違いの個数，難易度で間違い探しの問題を生成する．

3.1　既存の問題における間違いの種類の傾向調査

既存の間違い探しの問題において，間違いの種類の傾向を見つけるため調査を行った．使用した間違い探しの問題は，写真の間違い探しを多く取り扱っている *Independently published* 社出版の本 3 冊 [7][8][9] であり，全 150 題で調査した．調査の際，物体が削除される「削除」，物体が追加される「追加」，物体の色が変化する「色変化」，物体の大きさが変化する「大きさ変化」，物体の状態が変化する「状態変化」，物体の形が変化する「変形」，物体が別の物体に置き換えられる「置換」の 7 種類の傾向があったため，全ての間違いをこの 7 種類に分類した．左側を正解画像（以下元画像），右側を間違い箇所がある画像（以下間違い画像）とし，各種類の間違いの例を図 1〜 図 7 に示す．

各種類につき何個問題があったか調査した結果，間違いの個数全 2200 個に対し各種類の間違いの個数は表 1 のようになった．

表 1 から問題の傾向として「削除」,「追加」,「色変化」の 3 つが特に多いことがわかり，「大きさ変化」,「状態変化」,「変形」,「置換」の 4 つは前述の 3 つに比べ問題数が少ないことがわかった．その他にも画像処理による生成の行いやすさを考慮し，「SAGUS」では「削除」,「追加」,「色変化」の 3 種類の間違いを生成することにした．

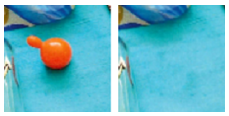
図 1　削除


図 2　追加


図 3　色変化


図 4　大きさ変化

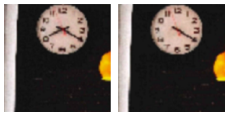
図 5　状態変化

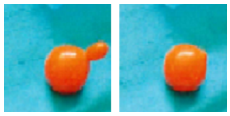
図 6　変形

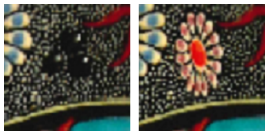
図 7　置換

3.2　顕著性マップ

顕著性マップは Itti らが提案した画像表現である [10]．顕著性マップは画像内の人の注視しやすい箇所を画像特徴量から推定する計算モデルにより算出される．顕著性マップの値はピクセルごとに算出し，視線の集中する確率を 0.0 ～ 1.0 で表す．この顕著性マップの値が大きいほど人の注視しやすい箇所であることを示す．顕著性マップは 1 枚のグレースケール画像やヒートマップとして表されることが多く，色が白い箇所ほど値が大きい．実際の使用例としては人物追跡 [11] や視線教示 [12] などが挙げられる．

3.2.1　顕著性マップの利用

本研究では，間違い画像生成時に顕著性マップの「人が注視しやすい箇所を可視化する」という特徴を利用できないかと考えた．間違い探しは，元画像と間違い画像を交互に見比べ間違いを探すのが主な解法である．その際，間違い画像のみを見て間違いに気付く時もある．そのため，顕著性が低い「目立たない」間違いを生成するというだけでは，2 つの画像間での違いが分からず，見つけにくい間違いを生成できているかの判断が難しい．それを解決するために，生成する間違いの顕著性のみを判断材料にするのではなく，元画像と間違い画像，両者の顕著性を考慮し間違いを生成すべきだと考えた．また，間違い探しにおける「見つける」という動作と，画像における「目立ちやすさ（顕著性）」に関連性があると考えた．そこで本研究では，間違いがある箇所において元画像と間違い画像の顕著性の差が小さいほど，その間違いは見つけにくいと仮定し間違い画

像生成の実装を行った．

3.2.2 顕著性の差の算出

元画像と間違い画像における顕著性の差を算出する手法を図 8 に示す．まず元画像と間違い画像の顕著性マップを生成する．次に生成した 2 枚の顕著性マップについて，同じ位置のピクセルの差分の絶対値をそれぞれ取った「顕著性差マップ」を生成する．この「顕著性差マップ」における間違い箇所のピクセルの値を全て合計したものを，その間違いにおける顕著性の差とする．なお顕著性マップの生成には，*opencv contrib* に実装されている中で，計算量の小さい Hou らの *Spectral Residual* 法 [13] を用いたものを使用する．この手法で生成した顕著性マップでは，1 *pixel* ごとに 0.0 ～ 1.0 の値が顕著性として算出され，この値が高いほど顕著性が高いことを示す．

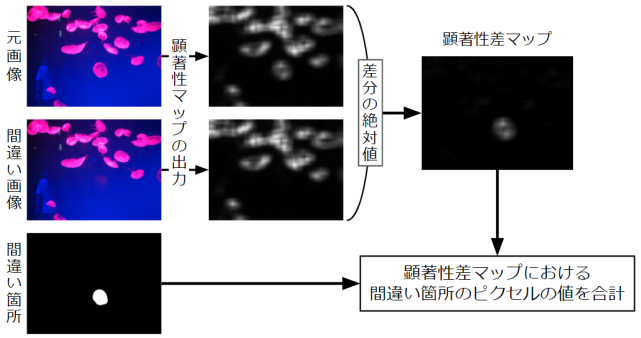

図 8 顕著性の差の取得方法

3.2.3 難易度と顕著性の差の関連性調査

3.2.1 節で述べたように，本研究では元画像と間違い画像の顕著性の差が小さいほど，見つける難易度が高いと仮定した．その仮定を検証するため，既存の間違い探しとそれを実際に解いた時のデータを利用した調査を行った．それに加え「SAGUS」に実装する難易度設定の参考にするべく，各難易度において間違いの顕著性の差がどの程度であるかの調査も兼ねた．なお本研究では難易度の種類を「易しい」，「普通」，「難しい」の 3 つに設定した．

調査方法

調査は 3.1 節で利用した市販の間違い探しの問題 150 題に含まれる間違いの内，「SAGUS」で扱う「削除」，「追加」，「色変化」の間違いを対象に行った．また難易度設定の基準としては実際に問題を解き，各間違いを見つけた順番を使用した．各問題に対し間違いを見つけた順番を 3 分割し，見つけた順番が早いものから間違いの難易度を「易しい (仮)」，「普通 (仮)」，「難しい (仮)」に分類した．具体的には計 15 個の間違いのある問題では，1 ～ 5 番目に見つけた間違いを「易しい (仮)」，6 ～ 10 番目に見つけた間違いを「普通 (仮)」，11 ～ 15 番目に見つけた間違いを「難しい (仮)」に分類した．これらの 3 つの難易度と 3 種類の間違いの組み合わせである計 9 通りについて，それぞれに分類された間違いの元画像との顕著性の差の平均を調査した．なお調査に使用した画像のサイズは，元画像・間違い画像ともに 500 × 310 *pixel* である．

調査結果

調査結果を表 2 に示す．「削除」「追加」「色変化」のいずれも「易しい (仮)」から「難しい (仮)」に向かって顕著性の差が小さくなっており 3.2.1 節での仮定には矛盾していないことが実証された．

表 2 調査結果

種類＼難易度	易しい (仮)	普通 (仮)	難しい (仮)
削除	38.21	37.24	22.53
追加	42.84	40.65	34.70
色変化	125.90	110.92	79.35

3.2.4 難易度設定

「SAGUS」では，3.2.3 節で述べたように「易しい」，「普通」，「難しい」の 3 段階の難易度を扱う．難易度は，間違いを生成する際に元画像との顕著性の差が各難易度の目標値に近くなるように設定する．表 2 から，顕著性の差の総和が小さくなるほど難易度の高い問題となることが分かった．そのため，「SAGUS」では各難易度・間違いの種類における顕著性の差の目標値を最終的に以下のように設定した．このような値に設定した理由としては「SAGUS」において生成する問題の難易度の差を強調し，難易度設定による効果を検証するためである．なお表 3 における値は調査した際に使用した画像サイズ 500 × 310 *pixel* の目標値であり，画像サイズに比例して変化する．

表 3 顕著性の差の目標値

種類＼難易度	易しい	普通	難しい
削除	38.2	19.1	0.0
追加	42.8	21.4	0.0
色変化	126.0	63.0	0.0

3.3 提案手法

「SAGUS」では大別して「物体抽出部」と「間違い生成・決定部」の 2 つで構成する．物体抽出部で題材画像内の物体を複数抽出し，抽出した物体を間違いの候補として間違い生成・決定部で最終的に出力する間違いを決定する．

3.3.1 物体抽出部

物体抽出部では，入力された画像から複数の物体を抽出し各物体ごとにマスク画像を作成する．マスク画像作成するためにユーザが入力した画像をグレースケール画像に変換し二値化処理を行う．二値化処理したとき白色になった部分を物体として抽出しマスク画像を作成し，そのマスク画像を使用し間違いの生成・決定を行う．

3.3.2 間違い生成・決定部

間違い生成・決定部では，まず生成する間違いの種類を 1 種類選び，選んだ種類の間違いを間違い候補として複数パターン生成，そしてその間違い候補の中から 1 つを決定する．決定の際には 3.2.4 節で述べたように，各難易度と間違いの種類ごとに目標値を設定し，間違い生成前後での

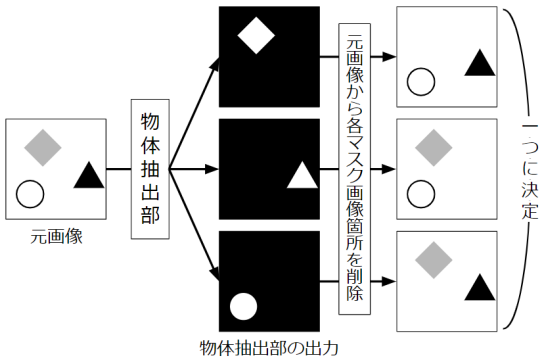

図 9 「削除」の間違いの決定手法

顕著性の差の値がその目標値に最も近いものを選ぶ．以上のことを，ユーザが指定した間違いの個数分繰り返すことで，決定した間違い全てが含まれた 1 枚の間違い画像を生成する．なお「SAGUS」における間違いの生成は，3.1 節で述べたように「削除」，「追加」，「色変化」の 3 種類を対象とする．

間違いの種類の決定

「SAGUS」では，エンタテインメント性を鑑みて様々な種類の間違いが一様に生成されるように，「削除」，「追加」，「色変化」の 3 つの間違いの種類からランダムに決定する．

「削除/色変化」の間違い生成・決定

「削除」の間違いを決定する手法を図 9 に示す．まず物体抽出部から受け取ったマスク画像群が示す箇所をそれぞれ間違いの場所の候補とし，それらのマスク画像の箇所を画像処理により削除した際の元画像との顕著性の差を記録する．全てのマスク画像について検証した後，顕著性の差が難易度設定により指定された目標値に最も近かったものを決定する．「色変化」の間違いを決定する手法についても削除と同様である．

「追加」の間違い生成・決定

「追加」の間違いを決定する手法を図 10 に示す（物体抽出部以前は図 9 と同様であるため省略）．まず物体抽出部から受け取ったマスク画像群の中から 1 枚をランダムに選択し，そのマスク画像により元画像から物体を切り抜く．切り抜いた物体を元画像に貼り付け，元画像との顕著性の値の差を記録する．この時に物体が元画像からはみ出したり，元々その物体があった場所，他の間違いと重なるような位置には物体を貼り付けない．以上のことを，物体を貼り付ける場所を変えながら繰り返し行う．物体を貼り付ける場所については，今述べた条件を満たす場所を 1 *pixel* 単位で全て検証することが理想である．こうして検証した中で，顕著性の差が難易度設定により指定された目標値に最も近かったものを決定する．

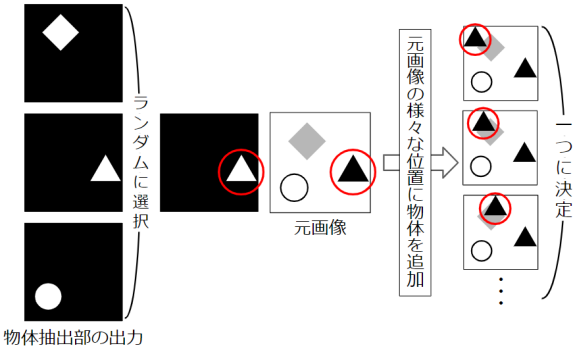

図 10 「追加」の間違いの決定手法

3.3.3 Web アプリケーション

概要でも述べた通り，「SAGUS」はエンタテインメント性に富んだものとなるよう開発を行った．また，「SAGUS」で間違い探しを生成した際，実際に間違い探しを解くには 2 枚の出力画像を横並びに表示する等の操作が必要となる．そのため，本研究ではユーザに対して楽しさと間違い探しの手軽さを提供するために「SAGUS」を搭載する為のアプリケーションを開発した．

4. システム構成

「SAGUS」のシステム構成図を図 11 に示す．「SAGUS」では入力された画像から物体を検出・抽出する物体抽出部，検出した物体に対し間違いを生成する場所と生成する間違いの種類を決定する間違い生成・決定部の 2 つで構成されている．ユーザは画像に加えて生成する間違いの個数と難易度を入力する．なお「SAGUS」は，*AMD Ryzen*™5 3600（6 コア 12 スレッド）を搭載し *Windows* 10 で動作する PC で開発を行っており，主要な開発言語には *Python* を使用し，画像処理では特筆しない限り *OpenCV* を用いている．

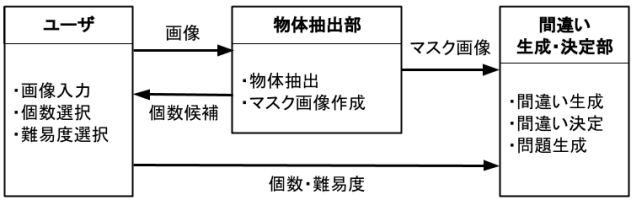

図 11 システム構成図

4.1 ユーザの入力

ユーザは以下の 3 つを入力する．

- 間違い探しの題材にする画像（元画像）
- 間違いの個数
- 難易度（易しい，普通，難しい）

画像は，写真やイラスト等ユーザが間違い探しの題材にしたい画像を入力する．また間違いの個数は 1 以上を選択しその上限は後述する物体抽出部で抽出した物体数となる．

4.2 物体抽出部

物体抽出部の全体の流れを図 12 に示す．物体抽出部では，ユーザが入力した画像をグレースケール画像に変換し，平滑化，二値化処理，物体除去を行ってマスク画像を作成する．作成するマスク画像の数は入力された画像によって異なり，抽出した物体の数だけマスク画像を作成する．

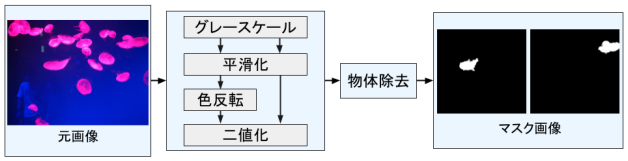

図 12 流れ図

4.2.1 二値化処理

二値化処理では，変換したグレースケール画像と閾値を基に画像を二値化する．物体を抽出する際の閾値は 0 ～ 255 まで 1 ずつ値を増やしていき，検出した物体の数を数え一番物体の数が多かった閾値で二値化しマスク画像を作成する．物体を抽出する前に，全体を任意の閾値で二値化し，物体が画像の半分以上を占めた場合グレースケール画像の色を反転させてから物体を抽出する．この時，閾値を自動で決定する大津の判別分析法を使用する [14]．

4.2.2 物体除去

物体を抽出する際，以下の 2 つの条件に当てはまる物体を除去する．

- 物体の輪郭面積が 200 *pixel* 以下の物体
- 一番輪郭面積が大きい物体

これは変化に気づきにくい小さい物体と抽出した物体が画像の大部分占めるのを防ぐためである．これらを除いた各物体のマスク画像を作成する．図 13 に入力画像と，今述べた手法で生成したマスク画像の一部を示す．

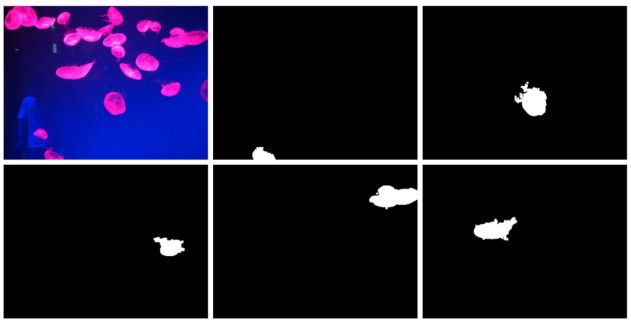

図 13 入力画像（左上）と生成したマスク画像の一部

4.3 間違い生成・決定部

間違い生成・決定部では物体抽出部から受け取ったマスク画像群を基に，ユーザが指定した数の間違いを含む間違い画像を生成する．ここでは提案手法で述べた事柄の内，間違いの生成についてより具体的に述べる．

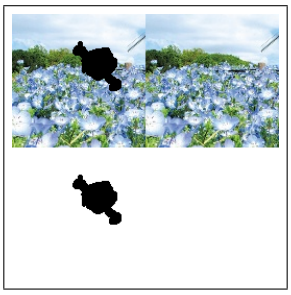

図 14 データセットの一例

4.3.1 「削除」の間違い生成

「削除」の間違いを生成する際には，Telea の手法 [16] を用いた物体削除と，後述する Isola らの *pix2pix* の技術 [15] を基に，学習したモデルを用いた物体削除を併用する．これは Telea の手法による物体削除では背景が単純な画像で生成結果が良く，今回学習したモデルによる物体削除では背景が複雑な画像で生成結果が良いという傾向が見られたためである．この双方の利点を生かす為に，2 つの物体削除の手法を併用している．

Telea の手法を用いた物体削除

OpenCV に実装されている，Telea の手法を基にしたものを用いて物体削除を行う．元画像と，削除する箇所を示すマスク画像を入力として，マスク画像の箇所が削除された画像を得ることができる．

***pix2pix* による学習モデルを用いた物体削除**

pix2pix の技術を基に学習を行い，学習したモデルを用いて物体削除を行う．*pix2pix* は条件付き *GAN* を利用した生成モデルの一種であり，入力画像と目標画像のペアで学習を行うことで，画像を入力としてそれに対応した画像を出力するモデルを生成する．今回は物体削除を学習するため，物体が削除される前後の画像のペアで学習を行った．学習の主要な条件は以下の通りである．

- データセット数: 20000 枚
- 入力・出力画像サイズ: 128 × 256 *pixel*
- エポック数: 30
- バッチサイズ: 4

なおデータセットは 2000 枚の背景画像と 100 枚の物体画像を用意し，それらを組み合わせることで作成した．実際のデータセットの一例を図 14 に示す．左半分が入力画像，右半分が目標画像であり，削除対象の画像とマスク画像を上下に並べたものを入力とすることで削除する場所を指定する．

2 種の削除結果の合成

これらの 2 種の削除結果を合成するにあたって，まず画像の複雑度を値として得る必要がある．これには，Ryan らの *Pixel Approximate Entropy* (*PAE*) を用いる [17]．これは折れ線グラフの視覚的な複雑度の尺度を表す値であり，画像を入力とすることはできない．そのため，*PAE* を計算するためのデータを以下の手順で取得する（図 15）．

(1) 削除対象の物体を包む凸包を求める．
(2) 求めた凸包を右回りになぞるように，凸包上のピクセルの元画像における *RGB* 値を順に配列に記録する．
(3) *RGB* ごとにデータを分割し，それら 3 つの 1 次元配列を折れ線グラフのデータとする．

これによって，凸包上における背景の *RGB* 値の変化が折れ線グラフとして取得でき，その *RGB* それぞれの折れ線グラフの *PAE* の平均を，背景画像の複雑度とする．この複雑度は 0.0 ～ 1.0 の値を取り，値が大きいほど背景画像が複雑であることを表すため，複雑度がそのまま学習モデルを用いた削除結果の合成比率となる．合成比率を基にして，Telea の手法による削除結果と学習モデルによる削除結果を合成したものが，最終的な削除結果となる．物体削除を行った結果を図 16 に，背景が単純な例（図上部）と複雑な例（図下部）を用いて示す．これらの例では学習モデルによる削除結果の合成比率が，背景が単純な例では 4%，複雑な例では 46%となった．

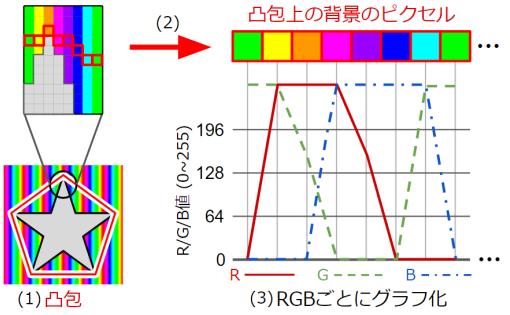

図 15 PAE の計算に使うグラフの取得

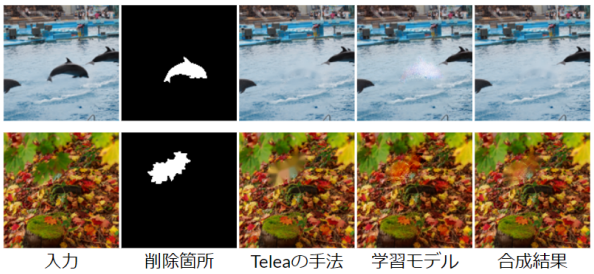

図 16 背景が単純な例（図上部）と複雑な例（図下部）の削除結果

4.3.2 「色変化」の間違い生成

「色変化」の間違いの生成では，画像の明度を変化させる．これによって，彩度が 0 であり色相情報を持たない色に対しても，適切な変化を与えられる．実際に生成された「色変化」の間違いの例を図 17 に示す．「色変化」の間違い生成は以下の手順で行う．

図 17 「色変化」の間違いの一例（左: 元画像，右: 間違い画像）

(1) マスク画像が示す箇所を色変化の対象ピクセルとする．
(2) 対象ピクセルにおける明度（0 ～ 255）の平均を計算し，明度の平均が 128 未満の場合は明度増加，128 以上の場合は明度減少を選ぶ．
(3) 対象ピクセルに対し，明度増加の場合は 64 を加算し，明度減少の場合は 64 を減算する．この際に明度変化によって値が 0 ～ 255 の範囲を逸脱するようなピクセルに対しては，明度の値を明度増加の場合は 255，明度減少の場合は 0 にすることで値の逸脱を防ぐ．

4.3.3 「追加」の間違い生成

「追加」の間違いの生成は，物体抽出部から受け取ったマスク画像により切り抜いた物体を，画像内に貼り付ける形で行う．マスク画像の選択はエンタテインメント性を鑑みて，様々なパターンの問題が生成されるようにランダムに行う．元画像から切り抜いた物体を，元画像の様々な位置に貼り付けたもの 1 枚 1 枚が間違い候補となる．物体を貼り付ける位置は，貼り付けた際に物体が元画像に収まるような範囲に，縦と横にそれぞれ指定数の物体を均等に並べた位置全てを検証する．ただし既に出力が決定している他の間違いと重なるような位置は除外する．また，縦と横に物体を並べる数についてはマルチスレッド処理の効率を考慮し，開発環境の *CPU* のスレッド数である 12 の倍数の 72 とした．なおこの値は 12 の倍数の中で，開発環境下における追加の間違いを 1 つ決定する際の処理時間が，100 例実行した中で 2 秒を超えなかった最大値である．本来であれば物体を 1 *pixel* 単位で全ての位置に貼り付けて検証することが理想であるが，今回はユーザが快適に使用できることを考慮し，暫定的にこの値とした．実際に生成された「追加」の間違いの例を図 18 と図 19 に示す．

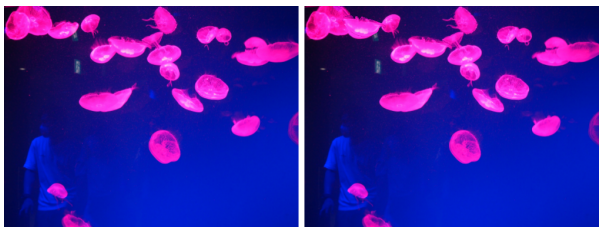

図 18 「追加」の間違いの一例（左: 元画像，右: 間違い画像）

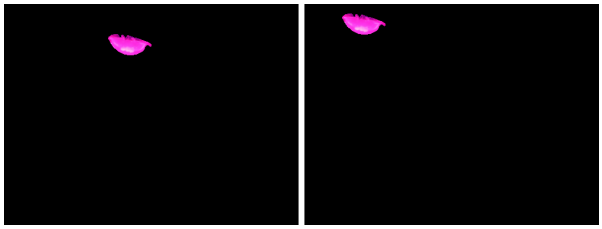

図 19 図 18 で追加物体として選ばれた箇所（左）と追加箇所（右）

Vol.2021-MUS-130 No.14
Vol.2021-EC-59 No.14
2021/3/16

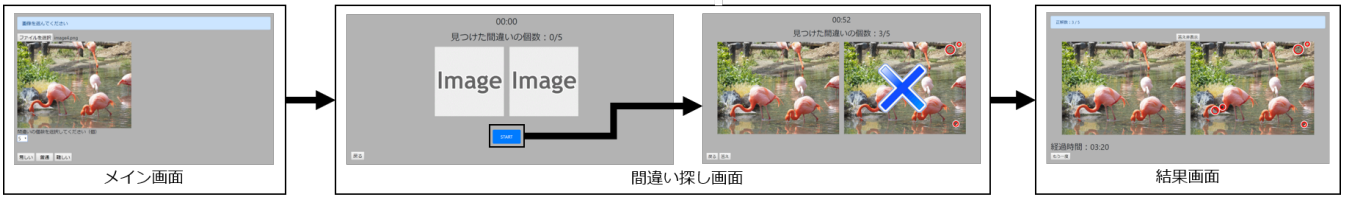

図 24　画面遷移図

4.4　結果の出力

「SAGUS」によって生成した間違い探しの例を難易度別に図 20～ 図 23 に示す．解答は 7 章に示す．

図 20　入力画像

図 21　「易しい」の間違い画像

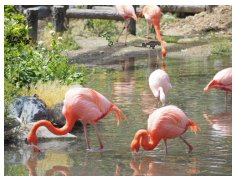

図 22　「普通」の間違い画像

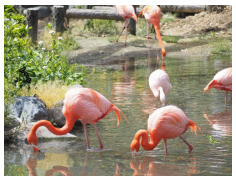

図 23　「難しい」の間違い画像

4.5　Web アプリケーション

アプリケーションは *Python* の *Eel* を用いて実装した．アプリケーションは，メイン画面，間違い探し画面，結果画面からなる．画面遷移図を図 24 に示す．

5.　評価・考察

本章では，大学生 11 人に「SAGUS」を実際に使用してもらい間違い探しの問題の完成度及びアプリケーションの使いやすさについての評価について述べる．

5.1　性能評価

5.1.1　評価方法

性能評価では，ユーザに間違いにしたい画像と間違いの個数を選んでもらい，「SAGUS」で各難易度の間違い画像をそれぞれ生成し解いてもらった．

5.1.2　評価結果

本評価実験では以下の 2 つの項目をそれぞれ 5 段階評価でアンケートをとった（表 4）．

(1) 問題を解いて楽しかったか
(2) 難易度が上がるにつれて問題は難しくなっていたか

表 4　性能評価のアンケート結果

項目		1	2	3	4	5	平均	
1	楽しくなかった	0	0	2	3	6	4.4	楽しかった
2	難しくなっていなかった	2	1	2	4	2	3.3	難しくなっていた

被験者からは「問題を解いて楽しかった」という問いに対し，「自分で用意した画像で間違い探しを行えたのが楽しかった」「市販の間違い探しでもありそうな問題で楽しかった」という意見を得られた．また「難易度が上がるにつれて問題は難しくなっていたか」という問いに対し，「難易度が上がるにつれて、間違いが小さいものが増えていった」「易しいから難しかった」との意見を得られた．生成結果の一例を難易度別に，図 25～ 図 28 に被験者が選んだ入力画像と共に示す．解答は 7 章に示す．

図 25　入力画像

図 26　「易しい」の間違い画像

図 27　「普通」の間違い画像

図 28　「難しい」の間違い画像

5.2　比較評価

5.2.1　評価方法

比較評価では，既存の間違い探しの問題と同じ題材を使用し「SAGUS」で生成した間違い探しの画像を解いてもらい比較した．使用した題材は全部で 3 題で間違いの個数は全題材 5 個の問題で行った．また「SAGUS」で生成した間違いの個数も既存のものに合わせ 5 個で行った．難易度は既存の間違い探しの問題には明記が無かったため「SAGUS」では中間の難易度である「普通」で間違い画像を生成した．

5.2.2　評価結果

本評価実験では「既存の問題と比べ遜色がなかったか」について 5 段階評価でアンケートをとった（表 5）．

表 5　比較評価のアンケート結果

項目		1	2	3	4	5	平均	
1	遜色があった	1	3	4	3	0	2.8	遜色がなかった

被験者からは「既存の問題と比べ遜色がなかったか」という問いに対し「システムで生成した間違い画像の方がノ

イズが強い」「物体の色の差でわかってしまう」「モザイクのような見てすぐ分かる間違いが複数あった」との意見を得られた．

5.3 アプリ評価

5.3.1 評価方法

アプリ評価では，実際に被験者にアプリを使い間違い探しを解いてもらった．この時使用した間違い探しは 5.2 節で述べた既存の間違い探しで，どの難易度を選択しても同じ問題が出力されるように設定した．

5.3.2 評価結果

本評価実験では以下の 2 つの項目をそれぞれ 5 段階評価でアンケートをとった（表 6）．

(1) アプリは使いやすかったか
(2) アプリをもう一度使いたいか

表 6 アプリ評価のアンケート結果

項目		1	2	3	4	5	平均	
1	使いにくかった	1	0	2	5	3	3.8	使いやすかった
2	使いたくない	1	0	1	4	5	4.5	使いたい

被験者からは「アプリは使いやすかったか」という問いに対し，「操作が分かりやすい」「直感的に操作できるようになっていた」という意見を得られた．また「アプリをもう一度使いたいか」という問いに対し，「ランダムに生成されて面白い」「色んな画像を試したい」との意見を得られた．

6. まとめ

「SAGUS」では，ユーザが入力した画像から複数パターンの間違い探しの問題を作成することができる．これにより，今まで 1 題につき 1 度しか問題を解くことができないという問題点を改善することができた．

一方で，生成できる間違いの種類の少なさや難易度設定の正確性が欠けている．3.1 節の通り間違いの種類を 3 つに絞り実装したが，調査では全部で 7 種類の間違いが存在した．問題をより楽しめるようにするためには「大きさ変化」や「状態変化」等今回実装していない 4 つの間違いの種類を実装し問題を作成する必要がある．また，3.2.4 節の通り，各難易度の顕著性の差の目標値を設定することで難易度別の問題の作成を可能にしてきた．しかし「SAGUS」では，難易度をわかりやすくするために「普通」と「難しい」の難易度を調査結果の通りの目標値に設定しなかったが，目標値を調査結果の通りの値に設定し直すことでより複雑な難易度設定が可能になると考えられる．これに加え，難易度設定の指標を見つけた順番だけでなく見つけるまでに経過した時間も考慮することでより絶対的な難易度設定が可能であると考えられる．これらを改善することで，より楽しめるシステムになると考えられる．

7. 付録

4.4 節と 5.1 節の間違い画像に対応する各難易度の正解を示すマスク画像を図 29 と図 30 に示す．

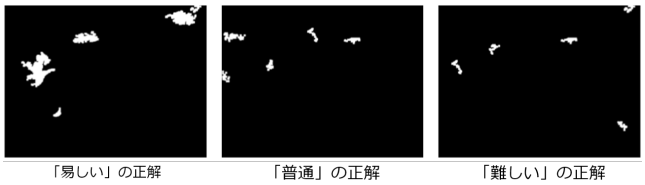

図 29 4.4 節の間違い画像に対応する正解

図 30 5.1 節の間違い画像に対応する正解

参考文献

[1] サイゼリヤ キッズメニュー 間違い探し，入手先〈https://www.saizeriya.co.jp/entertainment/〉（参照 2021-02-18）
[2] 太城敬良：「懐かしい！」が脳を若返らせる昭和レトロ間違い探し，宝島社 (2019).
[3] Fukuba, et al. “Brain Activation during the Spot the Difference Game”, MRMS, 8(1), (2009).
[4] Jin, JH., et al. “SPOID: a system to produce spot-the-difference puzzle images with difficulty”, The Visual Computer, 29, (2013).
[5] Park, SH., et al. “Automatic Generation of Spot-the-difference Game Contents using Image Inpainting”, Korea Game Society, 15(6), (2015).
[6] Liu, S., et al.“Snap & play: Auto-generate personalized find-the-difference mobile game”, MM’11, (2011).
[7] Supreme Spot the Difference Book for Adults: Various Food Picture Puzzles，Independently published(2019).
[8] Supreme Spot the Difference Book for Adults: Animal Picture Puzzles，Independently published(2019).
[9] Supreme Spot the Difference Book for Adults，Independently published(2019).
[10] Itti, L., et al.“A Model of Saliency-Based Visual Attention for Rapid Scene Analysis”, TPAMI, 20(11), (1998).
[11] 王茂ほか：視線追跡と顕著性マップに基づく全方向車椅子のためインタラクティブ教示システムの構築，FSS2014, 30, (2014).
[12] 前田陽一郎：顕著性マップに基づく全方向車椅子の視線教示システムにおける人間の目標指示についての行動意図推定，知能と情報, 30(5), (2018).
[13] Hou, X., et al.“Saliency Detection: A Spectral Residual Approach”, CVPR 2007, (2007).
[14] 田中成彦ほか：濃度共起ヒストグラムを用いた大津の判別分析法，MIRU2011, (2011).
[15] Isola, P., et al. “Image-to-Image Translation with Conditional Adversarial Networks”, CVPR 2017, (2017).
[16] Telea, A. “An Image Inpainting Technique Based on the Fast Marching Method”, Journal of Graphics Tools, 9(1), (2004).
[17] Ryan, G., et al. “At a Glance: Pixel Approximate Entropy as a Measure of Line Chart Complexity”, TVCG 2018, 25(1), (2018).