Vol.2022-HCI-197 No.35
2022/3/15

木製部材の画像認識結果への修正作業に対する格子線の影響

森茂智彦[1]　山口穂高[1]　藤巻吾朗[1]　生駒晃大[2]

概要：木製家具製造業における棚卸時の部材の在庫数をカウントするための支援 Web アプリケーションを開発してきた．このアプリケーションでは，画像認識により検出した部材に自動で印を付け，未検出や誤検出のものは後から修正する．この修正は人が行うため，修正ミスが発生する可能性がある．そこで本研究では，画像を格子線により小分けにすることで，より効率的に修正作業が行えるか検討を行った．格子の有無による未修正の数と作業時間の違いを比較した結果，格子線により小分けにすることで，未修正の数を減らせることがわかった．

キーワード：棚卸，視覚探索，格子状分割，Web アプリケーション

Effect of Grid Lines on the Correction Process for Image Recognition Results of Wooden Components

TOMOHIKO MORIMO[†1]　HODAKA YAMAGUCHI[†1]
GOROH FUJIMAKI[†1]　AKIHIRO IKOMA[†2]

Abstract: We have developed a web application to support inventory process by counting stocked parts for wooden furniture manufacturing. In this application, parts detected by image recognition are automatically marked, and undetected or incorrect detected parts are corrected later. Since this correction is done manually, there is a possibility that a human error may occur. In this study, we investigated whether the correction work can be done more efficiently by dividing the image into small parts using grid lines. As a result of comparing the difference in the number of uncorrected images and the work time between images with and without grid lines, it was found that the number of uncorrected images could be reduced by dividing the images into small parts with grid lines.

Keywords: Inventory, Visual Search, Grid Division, Web Application

1. はじめに

棚卸は，企業の正確な利益の把握や在庫管理に必要な作業である[1]．これまで筆者らは，木製家具製造業に焦点を当て棚卸の効率化に取り組んできた[2]．特に，椅子に使われる丸棒部材は，図 1 のように棚の中に規則的に並んでいるとは限らないため，在庫数を数えている最中に数え終えたものとまだ数えていないものの区別が分からなくなったり，丸棒が転がり配置が変わることでやり直しとなったりすることがあるため，効率的に在庫数をカウントできることが求められている．この課題に対し，図 2 に示すカウント支援 Web アプリケーションを開発した[2]．このアプリケーションは，部材の画像を撮影すると画像認識により，丸棒に自動で印が付される．その後，作業者が目視により未検出の部材に印を付したり，誤検出した印を消したりといった作業を行う．しかし，人が修正するため修正箇所を見逃してしまい，修正できていないところが生じる場合がある．

図 1　不規則に並ぶ丸棒の例

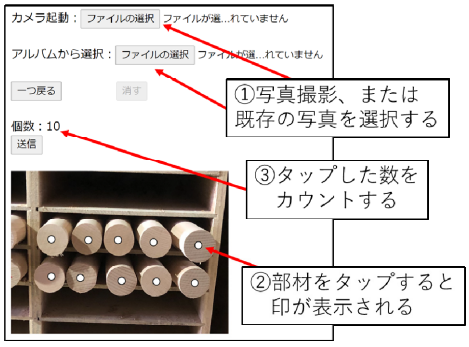


図 2　アプリケーション画面

1 岐阜県生活技術研究所
Gifu Pref. Research Institute for Human Life Technology
2 岐阜県産業技術総合センター
Gifu Pref. Industrial Technology Center

対策として，画像認識の精度向上と修正時の見逃しの効率化が挙げられる．前者は機械学習を用いる方法が挙げられるが，認識精度が常に 100%にはならないため，後者に対する対応を考えることも重要である．

図 2 のアプリケーションにおける修正作業は，目視により印の付されていない部材を見つける，もしくは部材に付されていない印を見つけることが要となるため，視覚探索といえる．視覚探索に関する研究として，Nakashima ら[3]によると，格子状に画像を分割することで仕切りが各領域を順に注目していくように誘導するため，探索の助けになると述べている．また，Li ら[4]によると，画面に表示された円形の印の数を数える際，領域を格子状に分割することで，時間が短縮され，数えた個数の正答率も高くなると述べている．さらに，画像を小分けにすることで数量判断が効率的になり，数量判断の際に起こりがちな見逃しの間違いを防ぐことも期待されている．

関連研究を踏まえ，本研究では，格子を用いて画像を小分けにすることでより効率的に修正作業が行えるかを検討する．具体的には，修正作業を行ったときの格子の有無による未修正の数と時間の違いを比較し検証する．

2. 関連研究

2.1 格子状分割に関する研究

Nakashima ら[3]は，領域を格子状に分割することによる視覚探索への影響を検討している．領域内に散りばめられた複数の C の文字の中にある 1 つの O の文字，または複数の O の文字の中にある 1 つの C の文字を目視で探す際の時間を評価した結果，分割枠は，最初は視覚探索を妨害するが，その後，各領域を順に注目していくように誘導するため，視覚探索の助けになると述べている．

Li ら[4]は，領域を格子状に小分けにすることで数量判断が効率的になると述べている．画面に表示された円形の印の数を数える際，領域を格子状に分割することで，時間が短縮され，数えた個数の正答率も高くなると述べている．

2.2 目視検査に関する研究

製品の目視検査と，画像上の部材に印が付されているかの確認は関連があると考えたため，下記研究を紹介する．

石井[5]は，製造業における検品作業での目視検査における周辺視目視検査法を紹介している．周辺視目視検査法とは，視野を拡大した周辺視の状態で視点を飛ばしながら見ていき，良品とは異なると違和感を持った付近を中心視で見て，良・不良を判断する方法である．習熟すれば誰もが不良を見逃すことなく高速かつ低疲労で検査できる方法であると述べている．

2.3 本研究の位置づけ

本研究では，Nakashima ら[3]と Li ら[4]の研究に着目する．先行研究では，実験に用いる画像の背景や文字・印の色はそれぞれ単一であるが，本研究における丸棒の部材棚は，図 1 のように棚の枠や丸棒部材の側面などが映っており，また，木口面が木目により均一ではない．さらに，格子線が部材の上を通過する時もあり，完全に小分けにできるとは限らない．また，Nakashima ら[3]の研究では，探索する文字は 1 か所であるが，本研究における修正箇所は画像認識結果により多数生じる場合もあり，修正箇所の数が不明な中で，それらすべてを見つける必要がある．そのため，本研究での修正作業に対する格子状分割の影響については明らかではない．

3. 実験方法

3.1 概要

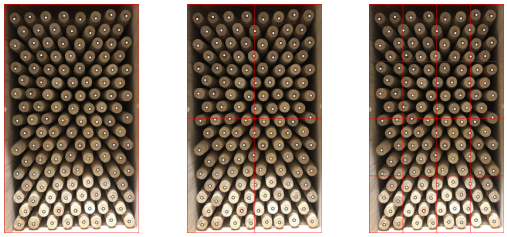
予め画像認識により印が付された複数の画像において，未検出の部材に印を付け，誤検出された印を消す作業を実験参加者に行わせた．なお，誤検出された印を未検出の部材上に移動させても良いとした．この修正作業を格子の分割条件ごとに繰り返し，条件間で未修正の数と時間を比較した．実験は，タブレット端末上で実施した．

3.2 実験システムの構築

実験システムは，Web アプリケーションを用いて作成した．画像の読み込みと予め付す印の表示，印の追記と削除には，Fabric.js[6]を用いた．Fabric.js は，HTML5 の Canvas 要素上で図形の描画・サイズ変更・削除などの操作を行うことができる JavaScript ライブラリである．撮影した部材棚の画像は，Canvas 要素の背景画像に設定した．図 3 に実験システムの画面を示す．表示領域の幅 400px，高さ 601px とし，実験参加者による表示の拡大縮小は不可に設定した．

＜操作方法＞
・印の追記：画像上をタップする．
・印の削除：印を選択した後，消すボタンを押す．
・印の選択（1 個）：印をタップする．
・印の選択（範囲指定）：タップした位置を開始点として，指をスライドさせ矩形の範囲を作る．その範囲内にある印が全て選択される．
・印の移動：印を選択した後，印をドラッグする．
・一つ戻る：一つ戻るボタンを押す．誤って追記した直前の印を消すことができる．
・修正完了：画像の左下にある完了ボタンを押す．

図 3　実験システムの画面



格子なし　　格子 2×2　　格子 4×4

図 4　格子条件別のサンプル画像の例

3.3 実験条件

(1) 格子条件

格子の分割条件は，格子なし，格子 2×2，格子 4×4 の 3 条件とし，色は赤色(#F00)，太さは 1px とした．各格子の条件で，サンプル画像や修正箇所は同じとした．

(2) サンプル画像

サンプル画像は，実際の木製家具製造現場での部材棚に収納された丸棒の画像 60 枚とした．部材棚を Apple iPhone8 で撮影し，24bit の画像を得た後，画像内の部材部分のみが映るようにトリミングした．これは，格子線で区切った際に，部材部分ができるだけ均等に分割されるようにするためであった．トリミング後の画像を，縦横比固定のまま幅 300px となるようにリサイズした．印は直径 6px，枠の太さ 1px で黒色(#000)，内部は白色(#FFF)とした．図 4 に格子条件別のサンプル画像の例を示す．

サンプル画像 60 枚のうち，解析に用いる画像は 50 枚とし，残り 10 枚は作業に慣れるための練習用とした．最初に練習用をランダムに並び替えた 10 枚を実施し，その後解析用をランダムに並び替えた 50 枚を実施した．実験参加者には画像が練習用と解析用に分かれていることは知らせず，実験参加者は 60 枚を通して行った．

修正前に画像に付される印は，機械学習によって認識された結果を用いた．学習モデルには YOLOv5[7]を用い，TensorFlow.js[8]上で画像認識させて得られた印の座標値を座標データとして保存した．本実験に使用した学習モデルの作成方法は，付録 A.1 に示す．

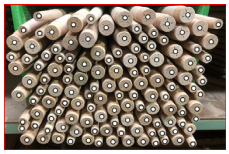
表 1 に，サンプル画像ごとに認識結果をまとめた結果を示す．認識結果から，練習用 10 枚を除くと，修正なし（全ての部材が認識されており，誤検出がないもの）17 枚，修正あり 33 枚となった．さらに，修正ありの内訳は，未検出のみ（認識されていない部材があるもの）6 枚，誤検出のみ（丸棒がないところに付された印があるもの）19 枚，未検出と誤検出両方があるもの 6 枚，低検出（認識結果が極端に悪かったもの）2 枚となった．サンプル画像に認識した印を付した画像を表 2 に示す．

表 1　サンプル画像の詳細

	No	本数	検出印数	未検出数	誤検出数	修正箇所計	画像幅	画像高さ
練習	1	16	15	4	3	7	300	141
	2	22	24	0	2	2	300	110
	3	28	30	0	2	2	300	195
	4	28	29	0	1	1	300	206
	5	32	32	0	0	0	300	303
	6	46	48	1	3	4	300	371
	7	53	51	2	0	2	300	400
	8	78	85	10	17	27	300	458
	9	78	78	0	0	0	300	510
	10	102	102	0	0	0	300	510
修正なし	11	14	14	0	0	0	300	121
	12	31	31	0	0	0	300	166
	13	32	32	0	0	0	300	186
	14	32	32	0	0	0	300	310
	15	35	35	0	0	0	300	295
	16	42	42	0	0	0	300	257
	17	42	42	0	0	0	300	312
	18	47	47	0	0	0	300	207
	19	60	60	0	0	0	300	357
	20	78	78	0	0	0	300	514
	21	86	86	0	0	0	300	239
	22	96	96	0	0	0	300	248
	23	114	114	0	0	0	300	265
	24	128	128	0	0	0	300	501
	25	130	130	0	0	0	300	198
	26	136	136	0	0	0	300	353
	27	213	213	0	0	0	300	187
未検出のみ	28	18	17	1	0	1	300	266
	29	21	20	1	0	1	300	99
	30	24	23	1	0	1	300	141
	31	35	33	2	0	2	300	151
	32	63	61	2	0	2	300	418
	33	91	90	1	0	1	300	225
誤検出のみ	34	11	12	0	1	1	300	100
	35	22	26	0	4	4	300	176
	36	24	28	0	4	4	300	122
	37	24	32	0	8	8	300	205
	38	25	27	0	2	2	300	160
	39	26	29	0	3	3	300	193
	40	28	31	0	3	3	300	208
	41	29	30	0	1	1	300	160
	42	29	32	0	3	3	300	227
	43	29	31	0	2	2	300	260
	44	31	40	0	9	9	300	458
	45	35	37	0	2	2	300	238
	46	40	42	0	2	2	300	207
	47	45	46	0	1	1	300	155
	48	52	54	0	2	2	300	331
	49	68	73	0	5	5	300	433
	50	72	73	0	1	1	300	261
	51	78	83	0	5	5	300	510
	52	215	216	0	1	1	300	396
未検出+誤検出	53	14	17	1	4	5	300	131
	54	22	23	2	3	5	300	258
	55	70	72	1	3	4	300	213
	56	164	161	4	1	5	300	368
	57	228	228	1	1	2	300	398
	58	262	262	1	1	2	300	238
低検出	59	29	7	23	1	24	300	73
	60	97	71	33	7	40	300	152

Vol.2022-HCI-197 No.35
2022/3/15

表 2　サンプル画像（解析用のみ・格子なし時）

No	+0	+1	+2	+3	+4
11					
16					
21					
26					
31					
36					
41					
46					
51					
56					

(3) 実験手順

実験の流れを図 5 に示す．画像と予め付す印の座標データを読み込み中には，関連研究[3][4]を参考に，灰色の中心に十字の画像（サイズ：300×400px）を表示させた．その後，印を付した画像を表示させ修正作業を実施させた．修正作業完了後，完了ボタンを押すと，次の画像と印の座標データを読み込み，灰色に十字の画像の表示から繰り返した．

実験参加者には，操作説明を行った後，「できるだけ早く行ってください」と説明した[4]．各回終了後，感想を聞いた．加えて 3 回目終了後は，以下のアンケートに答えさせた．

＜アンケート内容＞
Q1. これまで含め赤色の格子の有無に気づきましたか？
Q2. これまで含め赤色の格子が 2×2 と 4×4 の 2 パターンあったことに気づきましたか？
Q3. 格子の有無やパターンによってやりやすさに違いはありましたか？
Q4. 一番やりやすかったのはどれですか？その理由は？
Q5. 一番やりにくかったのはどれですか？その理由は？

実験参加者は，1 日 1 時間以上はスマホ，タブレットを操作している 30 代～50 代の男女 12 名（男性：6 名，女性：6 名）・(30 代 3 名，40 代 4 名，50 代 5 名)とした．

実験に使用するタブレットは，NEC 製 LAVIE TabE-TE710/KAW，ブラウザは Google Chrome とした．

実験は，1 回目 2021 年 12 月 14 日～2021 年 12 月 15 日，2 回目 2021 年 12 月 22 日～2021 年 12 月 27 日，3 回目 2022 年 1 月 6 日～2022 年 1 月 12 日に行った．記憶の影響を減らすため 1 週間以上空けてから次の条件を行った．各実験参加者の格子条件の実施順序を表 3 に示す．

(4) 記録内容と解析

実験時，サンプル画像ごとに修正後のサンプル画像，印の総数，作業時間を記録した．時間は，サンプル画像が表示されてから完了ボタンを押すまでの時間とした．

記録内容から，実験参加者と格子条件ごとに 50 枚のサンプル画像における未修正枚数，未修正印数，作業時間の合計をそれぞれ求めた．未修正枚数は，未修正部分が見られた画像の枚数，未修正印数は，50 枚の画像内で未修正であった印の個数，作業時間は，サンプル画像が表示されてから完了ボタンを押すまでの時間とした．この結果を用いて，下記の順で解析を行った．

まず順序効果を確認するために，各実験参加者の未修正数（未修正枚数，未修正印数）と作業時間を実施回数ごとに比較した．その後，格子条件間で比較するため，各実験参加者の未修正数と作業時間を格子条件ごとに比較した．

また，全体およびサンプル画像の修正内容ごとに，各実験参加者の未修正数と時間の合計の平均値と標準偏差を求めた．

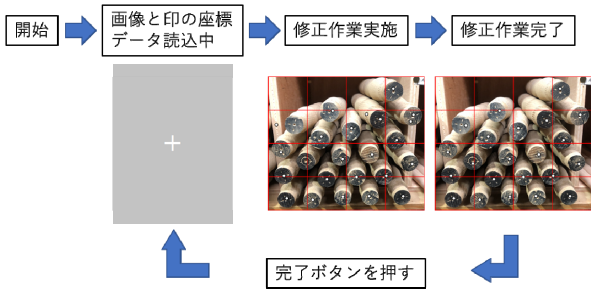


図 5　実験の流れ

表 3　実験順序

被験者	男性(年齢昇順)	S1	S2	S3	S4	S5	S6
	女性(年齢降順)	S7	S8	S9	S10	S11	S12
1回目		なし	なし	2×2	2×2	4×4	4×4
2回目		2×2	4×4	なし	4×4	なし	2×2
3回目		4×4	2×2	4×4	なし	2×2	なし

4. 結果と考察

4.1 順序効果

各実験参加者の未修正数と時間の合計を回数ごとに求めた結果を表 6 に示す．未修正枚数，未修正個数の結果から，実験を重ねるごとに未修正数が減るという明確な傾向は見られなかった．作業時間は実験を重ねるごとに短くなる傾向が見られた．実験後，実験参加者から意見を聞いたところ，「操作になれたため，手際がよくなった感じがする」などの意見があり，操作に慣れたことによるものであると考えられる．時間については一定の順序効果が出てしまったが，被験者間の代表値を算出することで，結果には影響しないレベルだと判断し，平均値と標準偏差を用いて結果を確認することとした．

4.2 格子条件間の比較

各実験参加者の未修正数と時間の合計を格子条件ごとに求めた結果を表 7 に示す．また，全体およびサンプル画像の修正内容ごとに，各実験参加者の未修正数と時間の合計の平均値と標準偏差を求めた結果を表 8 に示す．未修正枚数，未修正印数ともに，格子 2×2 が最も小さく，修正が正しく行えている結果となった．時間は全体で大きな違いは見られなかった．修正の有無や修正内容によって，傾向に大きな違いは見られなかった．

4.3 感想とアンケート結果

(1) 感想（抜粋）

・赤い線に沿って見やすい時と，線が邪魔なときがあった

・線があるほうがやりやすかった．線がある分，最初は戸惑ったけど
・線がないと一度見たところを何回も見ている気がした
・探している際の基準にはなるため，探しやすかった
・線と点が重なると見えにくいこともある
・径の大きい部材は見てすぐにわかるので，線はいらない
・格子が邪魔で見づらい，ブロック単位で見ていないため
・小さい画像に 4×4 だと見にくいし，大きい画像は 2×2 だと枠が広すぎると思う

(2) アンケート結果

表 4　Q1～Q3 の結果

	Q1	Q2	Q3
はい	10 人	6 人	10 人
いいえ	1 人	4 人	1 人
覚えていない	1 人	2 人	1 人

表 5　Q4，Q5 の結果

	Q4	Q5
格子なし	3 人 (S3,S4,S5)	6 人 (S6,S7,S8,S9, S10,S11)
格子あり(2×2)	2 人 (S8,S11)	0 人
格子あり(4×4)	4 人 (S6,S7,S9,S10)	3 人 (S1,S3,S5)
覚えていない	3 人 (S1,S2,S12)	3 人 (S2,S4,S12)

＜Q4 の理由＞
○格子なしが一番やりやすかった人の意見
・格子によって部材や白点がみにくくなる
・(格子があると)見づらい

○格子あり(4×4)が一番やりやすかった人の意見
・確認しやすい
・格子内をパッと見て判断しやすい
・確認をする時，格子ごとに見ていくことができたから

※格子あり(2×2)が一番やりやすかった人の意見は得られなかった

＜Q5 の理由＞
○格子なしが一番やりにくかった人の意見
・順番に確認しづらい
・本数が多く，点が細かいものは，途中で見失うから
・何本も木材があるとどこをみているか，途中で分からなくなる

○格子あり(4×4)が一番やりにくかった人の意見
・棒に被って見にくい，小さいもので特に
・格子によって部材や白点がみにくくなる

表 6　回数毎の結果

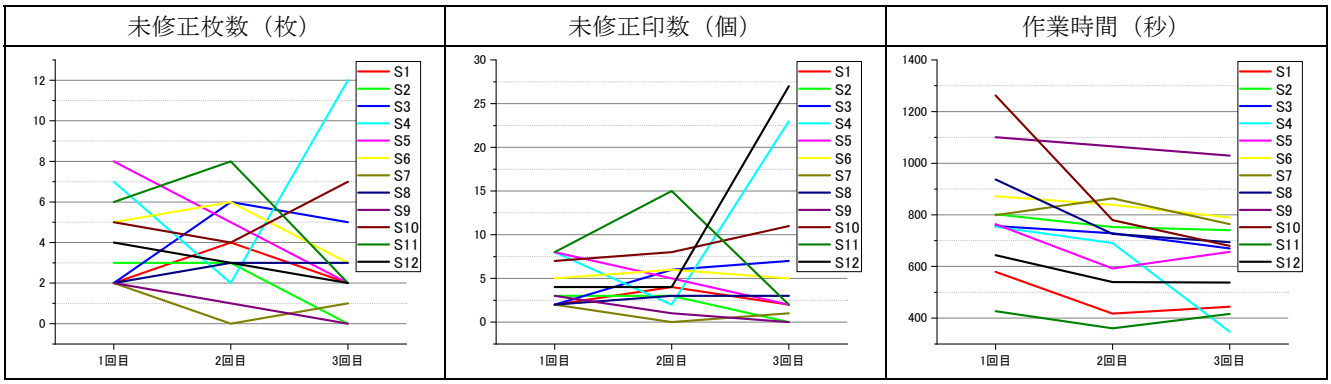


表 7　格子条件毎の結果

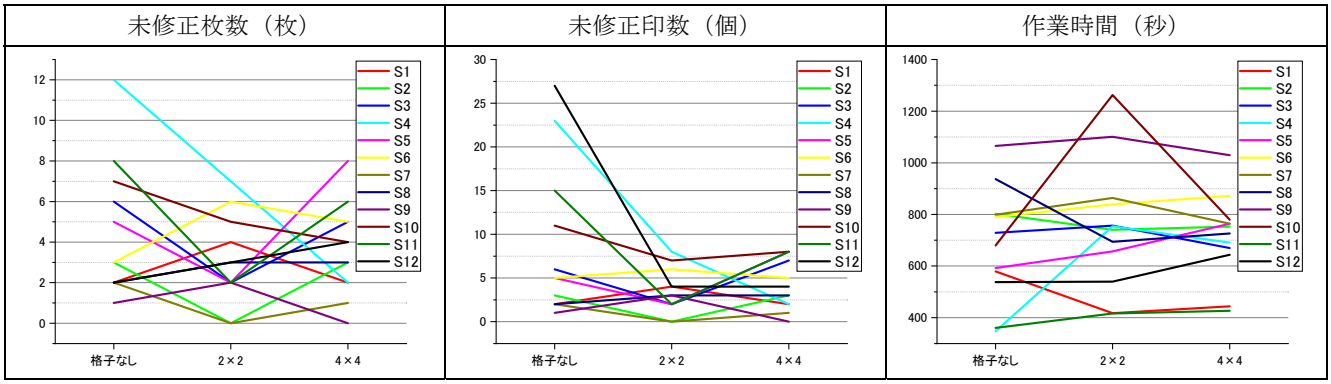
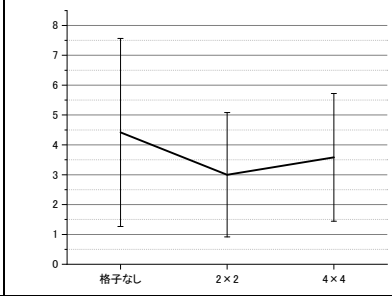
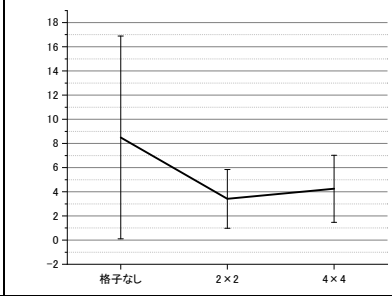
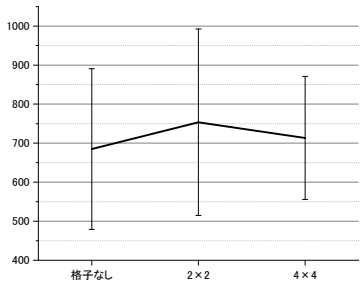
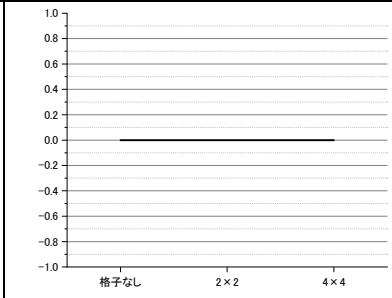
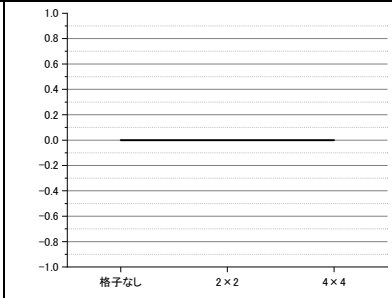
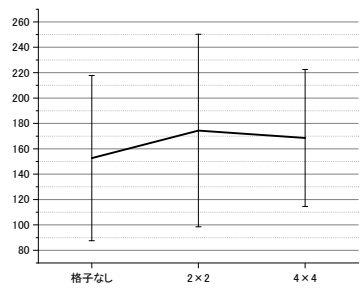
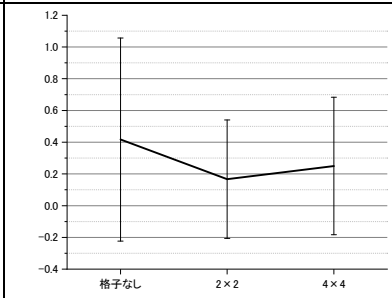
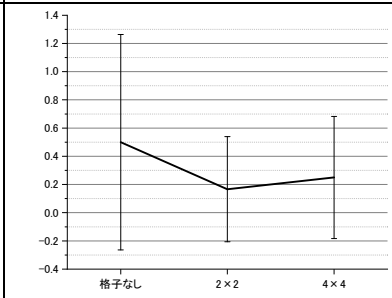
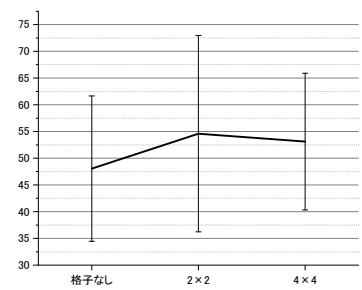
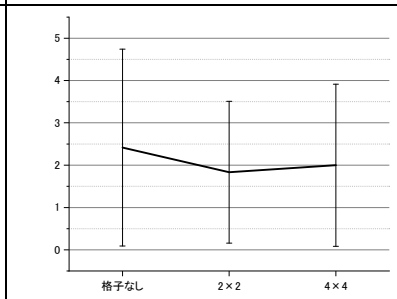
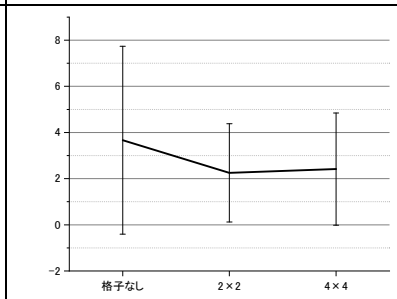
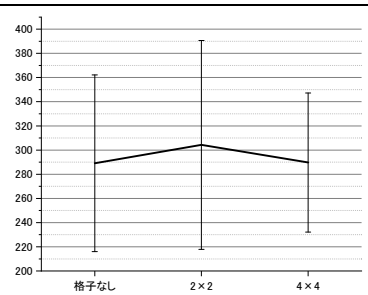
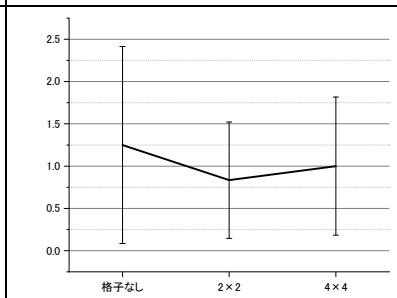
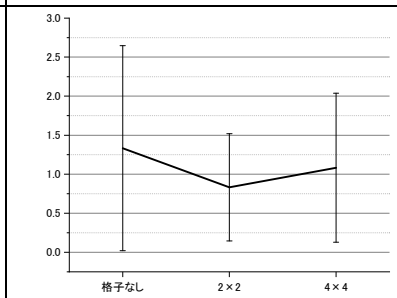
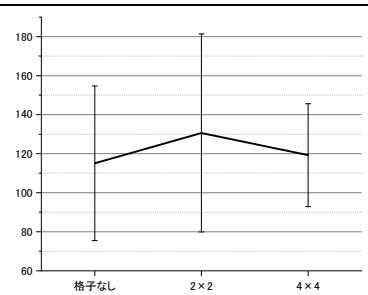
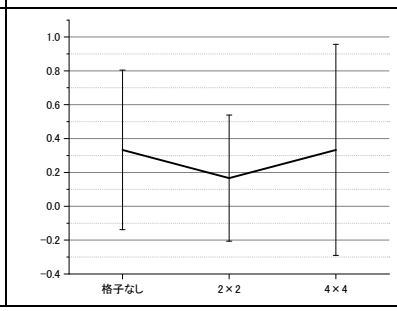
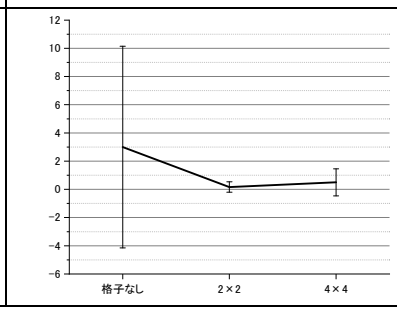
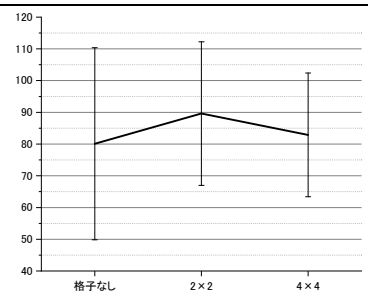


表 8　全体およびサンプル画像の修正内容ごとの結果

	未修正枚数（枚）	未修正印数（個）	作業時間（秒）
全体平均			
修正なし			
未検出のみ			
誤検出のみ			
未検出+誤検出			
低検出			

4.4 考察

表 8 から，未修正枚数，未修正印数ともに，格子 2×2 が最も小さく，特に未修正印数については，実験参加者間のばらつきも小さい結果となった．また，未修正印数の格子なしでは，ばらつきの範囲が大きく，問題なく修正できる可能性は低いと考えられる．

表 5 にて格子なしが最もやりやすかったと答えた S3, S4, S5 に着目する．表 7 の未修正枚数と未修正印数ともに，3 者とも格子なしより，格子 2×2 の条件のほうが少ない結果となっている．本人の意識とは異なり，格子 2×2 のほうが安定して良い結果が出せると考えられる．この点については，格子なしを好む人が，格子ありの条件で修正作業を実施した際に，実は格子を順に注目しているため未修正数が減ったのではないかと考えており，格子条件ごとに修正作業中の視線を計測するなどして検証していきたい．

作業時間については，格子条件間で大きな違いは見られなかった．これは，印を認識する時間に対して，印を追記や削除する時間の方が長く，差が現れなかったのではないかと考えており，作業時間の評価は今後の課題としたい．

その他の課題として，感想での意見から画像サイズや部材の径の大きさ，本数によって，より最適となる格子条件の導出が挙げられる．また，最適な格子線の太さも検討していきたい．

5. まとめ

本研究では，木製の丸棒部材の画像認識結果を修正し，全ての部材に印を付けるため，未検出の部材に印を付け，誤検出の印を消す作業において，画像を格子線により小分けにした際の影響について検討を行った．

画像を格子なし，格子あり（2×2），格子あり（4×4）に小分けした場合について修正作業を実施し，各実験参加者の未修正数の合計の平均値と標準偏差を格子条件ごとに比較した結果，格子あり（2×2）が最も安定して未修正数を減らせる結果となった．図 2 に示す Web アプリケーションに格子を実装し，棚卸に活用していきたい．

今後の課題として，部材の径の大きさ，本数によって，より最適となる格子条件の導出が挙げられる．導出することで，例えば画像認識の結果に応じて，小分けにする数が変更されて格子線が表示されるなどの活用が見込まれる．

謝辞 研究のきっかけを作っていただき，木製家具製造現場の写真撮影にご協力いただきました関係者の方々，実験に参加していただいた方々にこの場を借りて厚く御礼申し上げます．

参考文献

[1] “棚卸が重要なワケとは？基礎知識から効率化へのポイントをまとめてみました“. https://www.mylogi.jp/logistics/tanaoroshi-toha/, (参照 2021-03-17).
[2] 森茂智彦, 山口穂高, 藤巻吾朗. 木製家具製造業における棚卸の効率化の基礎検討と棚卸支援 Web アプリケーションの開発. 情報処理学会　研究報告ヒューマンコンピュータインタラクション（HCI）, 2021, vol. 193, no. 1, p. 1-7.
[3] R. Nakashima and K. Yokosawa. Visual search in divided areas: Dividers initially interfere with and later facilitate visual search. Attention, Perception, & Psychophysics, 2013, 75, p. 299–307.
[4] Q. Li, R. Nakashima and K. Yokosawa. Task-irrelevant spatial dividers facilitate counting and numerosity estimation. Scientific Reports, 2018, 8, 15620.
[5] 石井　明. いまだからこそ目視検査を見直す. 精密工学会誌, 2018, vol. 84, no. 12, p. 963-966.
[6] “Fabric.js Javascript Canvas Library“. http://fabricjs.com/, (参照 2021-04-19).
[7] “ultralytics/yolov5“. https://github.com/ultralytics/yolov5, (参照 2022-01-27).
[8] “TensorFlow.js | JavaScript デベロッパー向けの機械学習“. https://www.tensorflow.org/js, (参照 2022-01-27).

付録

付録 A.1 学習データの作成方法

学習モデルには，YOLOv5[7]を用いた．学習には，丸棒の画像 2,611 枚を用意した．モデルのパラメータファイルは yolov5s.yaml を使用，認識解像度 640×640，バッチサイズ 8，Epoch 数 300 で学習させた．

YOLOv5 は PyTorch で記述されているため，学習後，対応した形式にモデルを変換した．これは，本研究とは直接関連しないが，図 2 の Web アプリケーションでの機械学習による画像認識機能の実装に TensorFlow.js[8]を用いるため，それに合わせた．変換順序は，PyTorch→onnx→TensorFlow→TensorFlow.js であった．

学習モデルを用いて，TensorFlow.js 上で用意した画像を認識させた．認識後のバウンディングボックスは，しきい値（検出スコア*検出の確かさ）が 0.5 以上，ボックス同士の重なり比率（IOU）が 0.3 以下のものを最終的な認識結果とした．