# OCR

December 7, 2025

[1]:
```python
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.ndimage import interpolation as inter
import pytesseract
```

[62]:
```python
def load_and_binarize_image(image_path):
    img = cv.imread(image_path, 0)  # Load in grayscale
    _, binarized = cv.threshold(img, 200, 255, cv.THRESH_BINARY)
    return binarized
```

[63]:
```python
def get_negative_image(binarized):
    return 255 - binarized
```

[64]:
```python
def extract_connected_components(negative_img):
    num_labels, labels, stats, centroids = cv.
 ↪connectedComponentsWithStats(negative_img, connectivity=8)
    return num_labels, labels, stats, centroids
```

[120]:
```python
def select_candidate_points(labels, stats, centroids, strategy='centers'):
    candidate_points = []

    if strategy == 'centers':
        for c in centroids:
            candidate_points.append((int(c[0]), int(c[1])))

    elif strategy == 'max_y':
        for i in range(1, len(stats)):  # Skip the background
            component_mask = (labels == i)  # Get mask of connected component
            y_indices, x_indices = np.where(component_mask)  # Get indices of␣
 ↪all pixels in the component

            if len(y_indices) > 0:
                max_y_idx = np.argmax(y_indices)  # Find the index of the␣
 ↪maximum y-coordinate
                max_x = x_indices[max_y_idx]      # Corresponding x-coordinate
```

```python
                max_y = y_indices[max_y_idx]       # Maximum y-coordinate

                candidate_points.append((max_x, max_y))

    elif strategy == 'all':
        candidate_points = np.argwhere(labels > 0)
        candidate_points = [(int(pt[1]), int(pt[0])) for pt in candidate_points]

    return candidate_points
```

[7]:
```python
def remove_non_candidate_points(negative_img, candidate_points):
    result = np.zeros_like(negative_img)
    for point in candidate_points:
        result[point[1], point[0]]  = 255
    return result
```

[8]:
```python
def hough_transform(negative_img, threshold = 15):
    lines = cv.HoughLines(negative_img, 1, np.pi / 180, threshold)
    angles = [line[0][1] for line in lines]
    median_angle = np.median(angles)
    #document_angle = (median_angle - np.pi / 2) * 180 / np.pi
    document_angle = ((median_angle - (np.pi / 2)) * 180) / math.pi
    return document_angle
```

[9]:
```python
def deskew_image(image_path, angle):
    img = cv.imread(image_path)
    (h, w) = img.shape[:2]
    center = (w // 2, h // 2)
    M = cv.getRotationMatrix2D(center, angle, 1.0)
    rotated = cv.warpAffine(img, M, (w, h), flags=cv.INTER_CUBIC, borderMode=cv.
 ↪BORDER_REPLICATE)
    return rotated
```

[10]:
```python
def perform_ocr(image):
    return pytesseract.image_to_string(image)
```

[11]:
```python
def convert_to_pdf(image, output_path):
    pdf = pytesseract.image_to_pdf_or_hocr(image, extension='pdf')
    with open(output_path, 'w+b') as f:
        f.write(pdf)
```

[161]:
```python
# Function to display images using matplotlib
def display_images(images, titles, cmap='gray'):
    plt.figure(figsize=(15, 8))
    for i, image in enumerate(images):
        plt.subplot(1, len(images), i + 1)
```

```
        plt.imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB))  # Convert BGR to RGB␣
↪for proper display
        plt.title(titles[i])
        plt.axis('off')  # Turn off the axis ticks
    plt.show()
```

[121]:
```
# Load and process the image
image_path = 'doc.jpg'
binarized = load_and_binarize_image(image_path)
negative_img = get_negative_image(binarized)
num_labels, labels, stats, centroids =␣
↪extract_connected_components(negative_img)
```

[122]:
```
# Perform OCR on actual image
ocr_result_skewed = perform_ocr(cv.imread(image_path))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed)
```

```
OCR Result for Skewed Image:
4 deman on 3D data analysis Howeve, ten hap.
num @ PoSterion; Pens that the 3p data Cannot be Obtaj, lalit
e co "8 SPondin, MRE "Wationay Shown in Fig, ),
L PProximation lechnigy e is adopre Or the MAp Speci
€Stimation, The Proposed Method Valuated On both this
Wificial data nd ye ata obtained OM re, "SIrUCtion
Practical Scenes. xpering Fesults have OWN rh,
Obustness and e CNCY of 1 Propose, Nethg, ? Fepaip.
ing NOISY ang ‘Nncon le BD s,
1. Introduction
IPtion
Suppose we €N a sep of PR Mages of re Onstructeg
AN Objecy Capt t multip} View, ints, ct in the
Feal wory, i Pace) ig nl ren NStructe, ing some Relateg Work
3D re, struct, SOrithm, Td ally, objec; Can be op,
"Ved in RGR, -D Mages, it can M Reeonstructors Exist Pe Comp] On @PProache, rE
Use of, Seo.
Wever, in © have foung that the "Construction Metric in, Presenteg at either |,
"vel or high
Often £ "ven if the R BARG, "D data ig Complete. This evel, Scribes fy wuctures,
@
5 becaus, the thing OF the Ri a in Mecture- from. local ines, Hsed to fin Small
holes a
Motion baseq TECORStrCtign Methods ( 114) cong no: broke, aces, Por exam ©
Curless an {31 pro.
One accurat, ‘Pecial} OF objects of Unifi Colours, Pesed 0 ex °t Strfa Xamining
the boun Y OF un
°F recons ucti ethod. sing eg. y D, the S€en an, pty voxels. US met] Fequires
ad-
```

```
Missing "Pth coulg alsy use © incomple, ess. We Mtion; '8e in BES t ve away
Tedundant aces. In
iNustrate Tal cases OF thi. Situarj, in Fj, 1, }, Da 'S Ct al, filled 8aps ang S
on br en Surfaces
Recent adva NCES of 3 ‘Qisitio, Vices and 3p Scene y Performing a Olu on the
Signed distance Values,
Peconsine ion Research > 39, 40, gy have enableq TI 'S Process Was Tepeated UNEl
a ne IMDLicit surface Could
large-scale ACqUisition of 3D Scene data and this hag Taiseg be defineg & the
gang. In 116 " 4 broke; as repre.
Sented in 4 Octre, OD Which Inne] Outer grig Points
This work wa intel When Dye Thanh Neuyon 28 Working ap Were determin broken
Object was tp OUstructeg
the Singapore University oF Technology a Design, ry Contouring th Points, Inf M,
token
```

[123]:
```python
#Measuring speed and accuracy of 'Centers'
import time
start_time_center = time.time()

candidate_points_centers = select_candidate_points(labels, stats, centroids)
cleaned_negative_img_centers = remove_non_candidate_points(negative_img,␣
    ↪candidate_points_centers)
document_angle_centers = hough_transform(cleaned_negative_img_centers)
deskewed_image_centers = deskew_image(image_path, document_angle_centers)
ocr_result_deskewed_centers = perform_ocr(deskewed_image_centers)

end_time_centers = time.time()
elapsed_time_centers = end_time_centers - start_time_center
print("Time taken by strategy = Centers: ", elapsed_time_centers)
```

```
Time taken by strategy = Centers:  5.371365070343018
```

[124]:
```python
cv.imwrite('negative_image_centers.png', cleaned_negative_img_centers)
cv.imwrite('deskewed_image_centers.png', deskewed_image_centers)
```

[124]: True

[125]:
```python
#pytesseract.pytesseract.tesseract_cmd = r'C:\Program␣
    ↪Files\Tesseract-OCR\tesseract.exe'
```

[126]:
```python
# Convert to PDF
convert_to_pdf(cv.imread(image_path), 'skewed_document.pdf')
convert_to_pdf(deskewed_image_centers, 'deskewed_document_centers.pdf')
```

[127]:
```python
print("\nOCR Result for Deskewed Image, strategy = centers:")
print(ocr_result_deskewed_centers)
```

OCR Result for Deskewed Image, strategy = centers:
This CVPE

# A Field Model for Repairing 3D Shapes*

Duc Thanh Nguyen', Binh-Son Hua, Minh-Khoi Tran", Quang-Hieu Pham', and Sai-Kit Yeung?
1§chool of Information Technology, Deakin University, Australia
?Singapore University of Technology and Design, Singapore

## Abstract

This paper proposes a field model for repairing 3D shapes constructed from multi-view RGB data. Specifically, we represent a 3D shape in a Markov random field (MRF) in which the geometric information is encoded by random binary variables and the appearance information is retrieved from a set of RGB images captured at multiple viewpoints. The local priors in the MRF model capture the local structures of object shapes and are learnt from 3D shape templates using a convolutional deep belief network. Repairing a 3D shape is formulated as the maximum a posteriori (MAP) estimation in the corresponding MRF. Variational mean field approximation technique is adopted for the MAP estimation. The proposed method was evaluated on both artificial data and real data obtained from reconstruction of practical scenes. Experimental results have shown the robustness and efficiency of the proposed method in repairing noisy and incomplete 3D shapes.

## 1. Introduction

Suppose we are given a set of RGB/RGB-D images of an object captured at multiple viewpoints. The object in the real world (i.e. 3D space) is then re-constructed using some 3D reconstruction algorithm. Ideally, if an object can be observed in RGB/RGB-D images, it can be well reconstructed. However, in reality we have found that the reconstruction often fails even if the RGB/RGB-D data is complete. This is because the matching of the RGB data in structure-from-motion based reconstruction methods (e.g. [14]) could not be done accurately, specially for objects of uniform colours. For reconstruction methods using depth (e.g. [39, 4]), the missing of depth could also cause the incompleteness. We illustrate several cases of this situation in Fig. 1.

Recent advances of 3D acquisition devices and 3D scene

reconstruction research [28, 38, 39, 40, 4] have enabled
large-scale acquisition of 3D scene data and this has raised

s work was conducted when Duc Thanh Nguyen was working at
the Singapore University of Technology and Design.

ing [4].

a demand on 3D data analysis. However, it often hap-
pens that the 3D data cannot be obtained at high quality (as
shown in Fig. 1), even by recent reconstruction methods,
e.g. [4]. Specifically, the 3D surfaces are missing and/or
broken and this phenomenon causes difficulties for many
sequential tasks such as 3D object detection and recognition
[30, 36], shape analysis [20, 19], and scene understanding
{32, 12]. Repairing missing and broken surfaces thus plays
a critical role and deserves in-depth study. In this paper,
we focus on repairing incomplete 3D shapes. This problem
can be also referred to as shape completion. We assume
objects are not occluded, i.e. they can be fully observed in
RGB/RGB-D images. However, this assumption does not
mean that objects can be completely reconstructed.

1.1. Related Work

Existing shape completion approaches make use of geo-
metric information represented at either low-level or high-
level. Low-level geometry describes local structures, e.g.
local smoothness, and can be used to fill small holes on
broken surfaces. For example, Curless and Levoy [5] pro-
posed to extract surfaces by examining the boundary of un-
seen and empty voxels. However, this method requires ad-
ditional range images to carve away redundant surfaces. In
{7], Davis et al. filled gaps and holes on broken surfaces
by performing a convolution on the signed distance values.
This process was repeated until a new implicit surface could
be defined at the gaps. In [16], a broken object was repre-
sented in an octree grid on which inner and outer grid points
were determined. The broken object was then constructed
by contouring the grid points. In [29], holes on a broken

[128]:
```
#Tmeasuring speed and accuracy of 'max_y'
start_time_max_y = time.time()

candidate_points_max_y = select_candidate_points(labels, stats, centroids,␣
 ↪strategy='max_y')
```

```
cleaned_negative_img_max_y = remove_non_candidate_points(negative_img,␣
 ↪candidate_points_max_y)
document_angle_max_y = hough_transform(cleaned_negative_img_max_y)
deskewed_image_max_y = deskew_image(image_path, document_angle_max_y)
ocr_result_deskewed_max_y = perform_ocr(deskewed_image_max_y)

end_time_max_y = time.time()
elapsed_time_max_y = end_time_max_y - start_time_max_y
print("Time taken by strategy = Max y: ", elapsed_time_max_y)
```

```
Time taken by strategy = Max y:  70.80184078216553
```

[129]:
```
cv.imwrite('negative_image_max_y.png', cleaned_negative_img_max_y)
cv.imwrite('deskewed_image_max_y.png', deskewed_image_max_y)
```

[129]:
```
True
```

[130]:
```
# Convert to PDF
convert_to_pdf(deskewed_image_max_y, 'deskewed_document_max_y.pdf')
```

[131]:
```
print("\nOCR Result for Deskewed Image, strategy = max_y:")
print(ocr_result_deskewed_max_y)
```

```

OCR Result for Deskewed Image, strategy = max_y:
This CVPE

A Field Model for Repairing 3D Shapes*

Duc Thanh Nguyen’, Binh-Son Hua, Minh-Khoi Tran", Quang-Hieu Pham’, and Sai-Kit
Yeung?
1§chool of Information Technology, Deakin University, Australia
?Singapore University of Technology and Design, Singapore

Abstract

This paper proposes a field model for repairing 3D
shapes constructed from multi-view RGB data. Specifically,
we represent a 3D shape in a Markov random field (MRF) in
which the geometric information is encoded by random bi-
nary variables and the appearance information is retrieved
from a set of RGB images captured at multiple viewpoints.
The local priors in the MRF model capture the local struc-
tures of object shapes and are learnt from 3D shape tem-
plates using a convolutional deep belief network. Repair-
ing a 3D shape is formulated as the maximum a posteriori
(MAP) estimation in the corresponding MRF. Variational
mean field approximation technique is adopted for the MAP
```

estimation. The proposed method was evaluated on both artificial data and real data obtained from reconstruction of practical scenes. Experimental results have shown the robustness and efficiency of the proposed method in repairing noisy and incomplete 3D shapes.

# 1. Introduction

Suppose we are given a set of RGB/RGB-D images of an object captured at multiple viewpoints. The object in the real world (i.e. 3D space) is then re-constructed using some 3D reconstruction algorithm. Ideally, if an object can be observed in RGB/RGB-D images, it can be well reconstructed. However, in reality we have found that the reconstruction often fails even if the RGB/RGB-D data is complete. This is because the matching of the RGB data in structure-from-motion based reconstruction methods (e.g. [14]) could not be done accurately, specially for objects of uniform colours. For reconstruction methods using depth (e.g. [39, 4]), the missing of depth could also cause the incompleteness. We illustrate several cases of this situation in Fig. 1.

Recent advances of 3D acquisition devices and 3D scene reconstruction research [28, 38, 39, 40, 4] have enabled large-scale acquisition of 3D scene data and this has raised

s work was conducted when Duc Thanh Nguyen was working at the Singapore University of Technology and Design.

ing [4].

a demand on 3D data analysis. However, it often happens that the 3D data cannot be obtained at high quality (as shown in Fig. 1), even by recent reconstruction methods, e.g. [4]. Specifically, the 3D surfaces are missing and/or broken and this phenomenon causes difficulties for many sequential tasks such as 3D object detection and recognition [30, 36], shape analysis [20, 19], and scene understanding {32, 12]. Repairing missing and broken surfaces thus plays a critical role and deserves in-depth study. In this paper, we focus on repairing incomplete 3D shapes. This problem can be also referred to as shape completion. We assume objects are not occluded, i.e. they can be fully observed in RGB/RGB-D images. However, this assumption does not mean that objects can be completely reconstructed.

## 1.1. Related Work

```
Existing shape completion approaches make use of geo-
metric information represented at either low-level or high-
level. Low-level geometry describes local structures, e.g.
local smoothness, and can be used to fill small holes on
broken surfaces. For example, Curless and Levoy [5] pro-
posed to extract surfaces by examining the boundary of un-
seen and empty voxels. However, this method requires ad-
ditional range images to carve away redundant surfaces. In
{7], Davis et al. filled gaps and holes on broken surfaces
by performing a convolution on the signed distance values.
This process was repeated until a new implicit surface could
be defined at the gaps. In [16], a broken object was repre-
sented in an octree grid on which inner and outer grid points
were determined. The broken object was then constructed
by contouring the grid points. In [29], holes on a broken
```

[132]:
```python
#measuring speed and accuracy of "all"
start_time_all = time.time()

candidate_points_all = select_candidate_points(labels, stats, centroids,␣
 ↪strategy='all')
cleaned_negative_img_all = remove_non_candidate_points(negative_img,␣
 ↪candidate_points_all)
document_angle_all = hough_transform(cleaned_negative_img_all)
deskewed_image_all = deskew_image(image_path, document_angle_all)
ocr_result_deskewed_all = perform_ocr(deskewed_image_all)

end_time_all = time.time()
elapsed_time_all = end_time_all - start_time_all
print("Time taken by strategy = all: ", elapsed_time_all)
```

```
Time taken by strategy = all:  2.9022371768951416
```

[133]:
```python
cv.imwrite('negative_image_all.png', cleaned_negative_img_all)
cv.imwrite('deskewed_image_all.png', deskewed_image_all)
```

[133]: True

[134]:
```python
# Convert to PDF
convert_to_pdf(deskewed_image_all, 'deskewed_document_all.pdf')
```

[135]:
```python
print("\nOCR Result for Deskewed Image, strategy = all:")
print(ocr_result_deskewed_all)
```

```
OCR Result for Deskewed Image, strategy = all:
```

- Time taken by strategy = **Centers:** 5.371365070343018
- Time taken by strategy = **Max y:** 70.80184078216553
- Time taken by strategy = **all:** 2.9022371768951416

[136]:
```
# Varying Threshold
# Define the function to run the experiment with varying threshold values
def run_experiment(threshold_values, strategy_name):
    for threshold in threshold_values:
        print(f"\nRunning experiment with threshold = {threshold} and strategy␣
↪= {strategy_name}:")

        # Start timing
        start_time = time.time()

        # Select candidate points based on the strategy
        candidate_points = select_candidate_points(labels, stats, centroids,␣
↪strategy=strategy_name)
        cleaned_negative_img = remove_non_candidate_points(negative_img,␣
↪candidate_points)

        # Apply Hough transform with the given threshold
        document_angle = hough_transform(cleaned_negative_img,␣
↪threshold=threshold)

        # Deskew the image using the calculated angle
        deskewed_image = deskew_image(image_path, document_angle)

        # Perform OCR on the deskewed image
        ocr_result = perform_ocr(deskewed_image)

        # End timing
        end_time = time.time()
        elapsed_time = end_time - start_time

        # Output the results
        print(f"Time taken: {elapsed_time:.4f} seconds")
        print(f"Document Angle (in degrees): {document_angle:.2f}")
        print("\nOCR Result:")
        print(ocr_result)
```

[137]:
```
# Experiment with three threshold values
threshold_values = [5, 10, 15, 20]
```

[138]:
```
# Experiment with strategy = centers
print("Experiment for Strategy = Centers")
run_experiment(threshold_values, strategy_name='centers')
```

```
Experiment for Strategy = Centers
```

```
Running experiment with threshold = 5 and strategy = centers:
Time taken: 1.7561 seconds
Document Angle (in degrees): -1.00

OCR Result:


Running experiment with threshold = 10 and strategy = centers:
Time taken: 2.1491 seconds
Document Angle (in degrees): -15.00

OCR Result:
4 hp
wigs, fas
elbogg
Msi 1g Ng,
Use6 "ion, es or
ye dena, (0,
2o, 19, ang
renee Ke,
Ole ‘d Lesa, °S
Won "Paine it
Can be Us fy
Yer, a
RG,
CB. Mage, or
Wy ing de Bjeo, Dep
y ne si,
Rony ayy, es
d. CR (Ge toan be
Wey, ° tn, ty Beane
" bay. Len, ir
Pea Le
Yon ba,



Running experiment with threshold = 15 and strategy = centers:
Time taken: 5.1444 seconds
Document Angle (in degrees): 10.00

OCR Result:
This CVPE

A Field Model for Repairing 3D Shapes*

Duc Thanh Nguyen’, Binh-Son Hua, Minh-Khoi Tran", Quang-Hieu Pham’, and Sai-Kit
```

Yeung?
1§chool of Information Technology, Deakin University, Australia
?Singapore University of Technology and Design, Singapore

Abstract

This paper proposes a field model for repairing 3D shapes constructed from multi-view RGB data. Specifically, we represent a 3D shape in a Markov random field (MRF) in which the geometric information is encoded by random binary variables and the appearance information is retrieved from a set of RGB images captured at multiple viewpoints. The local priors in the MRF model capture the local structures of object shapes and are learnt from 3D shape templates using a convolutional deep belief network. Repairing a 3D shape is formulated as the maximum a posteriori (MAP) estimation in the corresponding MRF. Variational mean field approximation technique is adopted for the MAP estimation. The proposed method was evaluated on both artificial data and real data obtained from reconstruction of practical scenes. Experimental results have shown the robustness and efficiency of the proposed method in repairing noisy and incomplete 3D shapes.

1. Introduction

Suppose we are given a set of RGB/RGB-D images of an object captured at multiple viewpoints. The object in the real world (i.e. 3D space) is then re-constructed using some 3D reconstruction algorithm. Ideally, if an object can be observed in RGB/RGB-D images, it can be well reconstructed. However, in reality we have found that the reconstruction often fails even if the RGB/RGB-D data is complete. This is because the matching of the RGB data in structure-from-motion based reconstruction methods (e.g. [14]) could not be done accurately, specially for objects of uniform colours. For reconstruction methods using depth (e.g. [39, 4]), the missing of depth could also cause the incompleteness. We illustrate several cases of this situation in Fig. 1.

Recent advances of 3D acquisition devices and 3D scene reconstruction research [28, 38, 39, 40, 4] have enabled large-scale acquisition of 3D scene data and this has raised

s work was conducted when Duc Thanh Nguyen was working at the Singapore University of Technology and Design.

ing [4].

a demand on 3D data analysis. However, it often happens that the 3D data cannot be obtained at high quality (as shown in Fig. 1), even by recent reconstruction methods, e.g. [4]. Specifically, the 3D surfaces are missing and/or broken and this phenomenon causes difficulties for many sequential tasks such as 3D object detection and recognition [30, 36], shape analysis [20, 19], and scene understanding {32, 12]. Repairing missing and broken surfaces thus plays a critical role and deserves in-depth study. In this paper, we focus on repairing incomplete 3D shapes. This problem can be also referred to as shape completion. We assume objects are not occluded, i.e. they can be fully observed in RGB/RGB-D images. However, this assumption does not mean that objects can be completely reconstructed.

1.1. Related Work

Existing shape completion approaches make use of geometric information represented at either low-level or high-level. Low-level geometry describes local structures, e.g. local smoothness, and can be used to fill small holes on broken surfaces. For example, Curless and Levoy [5] proposed to extract surfaces by examining the boundary of unseen and empty voxels. However, this method requires additional range images to carve away redundant surfaces. In {7], Davis et al. filled gaps and holes on broken surfaces by performing a convolution on the signed distance values. This process was repeated until a new implicit surface could be defined at the gaps. In [16], a broken object was represented in an octree grid on which inner and outer grid points were determined. The broken object was then constructed by contouring the grid points. In [29], holes on a broken

Running experiment with threshold = 20 and strategy = centers:
Time taken: 7.3379 seconds
Document Angle (in degrees): 10.00

OCR Result:
This CVPE

A Field Model for Repairing 3D Shapes*

Duc Thanh Nguyen’, Binh-Son Hua, Minh-Khoi Tran", Quang-Hieu Pham’, and Sai-Kit Yeung?
1§chool of Information Technology, Deakin University, Australia
?Singapore University of Technology and Design, Singapore

Abstract

This paper proposes a field model for repairing 3D shapes constructed from multi-view RGB data. Specifically, we represent a 3D shape in a Markov random field (MRF) in which the geometric information is encoded by random binary variables and the appearance information is retrieved from a set of RGB images captured at multiple viewpoints. The local priors in the MRF model capture the local structures of object shapes and are learnt from 3D shape templates using a convolutional deep belief network. Repairing a 3D shape is formulated as the maximum a posteriori (MAP) estimation in the corresponding MRF. Variational mean field approximation technique is adopted for the MAP estimation. The proposed method was evaluated on both artificial data and real data obtained from reconstruction of practical scenes. Experimental results have shown the robustness and efficiency of the proposed method in repairing noisy and incomplete 3D shapes.

## 1. Introduction

Suppose we are given a set of RGB/RGB-D images of an object captured at multiple viewpoints. The object in the real world (i.e. 3D space) is then re-constructed using some 3D reconstruction algorithm. Ideally, if an object can be observed in RGB/RGB-D images, it can be well reconstructed. However, in reality we have found that the reconstruction often fails even if the RGB/RGB-D data is complete. This is because the matching of the RGB data in structure-from-motion based reconstruction methods (e.g. [14]) could not be done accurately, specially for objects of uniform colours. For reconstruction methods using depth (e.g. [39, 4]), the missing of depth could also cause the incompleteness. We illustrate several cases of this situation in Fig. 1.

Recent advances of 3D acquisition devices and 3D scene reconstruction research [28, 38, 39, 40, 4] have enabled large-scale acquisition of 3D scene data and this has raised

s work was conducted when Duc Thanh Nguyen was working at the Singapore University of Technology and Design.

ing [4].

a demand on 3D data analysis. However, it often happens that the 3D data cannot be obtained at high quality (as

```
shown in Fig. 1), even by recent reconstruction methods,
e.g. [4]. Specifically, the 3D surfaces are missing and/or
broken and this phenomenon causes difficulties for many
sequential tasks such as 3D object detection and recognition
[30, 36], shape analysis [20, 19], and scene understanding
{32, 12]. Repairing missing and broken surfaces thus plays
a critical role and deserves in-depth study. In this paper,
we focus on repairing incomplete 3D shapes. This problem
can be also referred to as shape completion. We assume
objects are not occluded, i.e. they can be fully observed in
RGB/RGB-D images. However, this assumption does not
mean that objects can be completely reconstructed.

1.1. Related Work

Existing shape completion approaches make use of geo-
metric information represented at either low-level or high-
level. Low-level geometry describes local structures, e.g.
local smoothness, and can be used to fill small holes on
broken surfaces. For example, Curless and Levoy [5] pro-
posed to extract surfaces by examining the boundary of un-
seen and empty voxels. However, this method requires ad-
ditional range images to carve away redundant surfaces. In
{7], Davis et al. filled gaps and holes on broken surfaces
by performing a convolution on the signed distance values.
This process was repeated until a new implicit surface could
be defined at the gaps. In [16], a broken object was repre-
sented in an octree grid on which inner and outer grid points
were determined. The broken object was then constructed
by contouring the grid points. In [29], holes on a broken
```

Running experiment with threshold = 15 and strategy = centers:
Time taken: 5.1444 seconds

Running experiment with threshold = 20 and strategy = centers: Time taken: 7.3379 seconds

[31]:
```python
#Experiment with stretegy = Max y
print("\nExperiment for Strategy = Max y")
run_experiment(threshold_values, strategy_name='max_y')
```

```
Experiment for Strategy = Max y

Running experiment with threshold = 10 and strategy = max_y:
Time taken: 1.4301 seconds
Document Angle (in degrees): -25.00

OCR Result:
```

Running experiment with threshold = 15 and strategy = max_y:
Time taken: 3.6080 seconds
Document Angle (in degrees): 10.00

OCR Result:
This CVPE

A Field Model for Repairing 3D Shapes*

Duc Thanh Nguyen’, Binh-Son Hua, Minh-Khoi Tran", Quang-Hieu Pham’, and Sai-Kit Yeung?
1§chool of Information Technology, Deakin University, Australia
?Singapore University of Technology and Design, Singapore

Abstract

This paper proposes a field model for repairing 3D shapes constructed from multi-view RGB data. Specifically, we represent a 3D shape in a Markov random field (MRF) in which the geometric information is encoded by random binary variables and the appearance information is retrieved from a set of RGB images captured at multiple viewpoints. The local priors in the MRF model capture the local structures of object shapes and are learnt from 3D shape templates using a convolutional deep belief network. Repairing a 3D shape is formulated as the maximum a posteriori (MAP) estimation in the corresponding MRF. Variational mean field approximation technique is adopted for the MAP estimation. The proposed method was evaluated on both artificial data and real data obtained from reconstruction of practical scenes. Experimental results have shown the robustness and efficiency of the proposed method in repairing noisy and incomplete 3D shapes.

1. Introduction

Suppose we are given a set of RGB/RGB-D images of an object captured at multiple viewpoints. The object in the real world (i.e. 3D space) is then re-constructed using some 3D reconstruction algorithm. Ideally, if an object can be observed in RGB/RGB-D images, it can be well reconstructed. However, in reality we have found that the reconstruction often fails even if the RGB/RGB-D data is complete. This is because the matching of the RGB data in structure-from-motion based reconstruction methods (e.g. [14]) could not be done accurately, specially for objects of uniform colours.

For reconstruction methods using depth (e.g. [39, 4]), the missing of depth could also cause the incompleteness. We illustrate several cases of this situation in Fig. 1.

Recent advances of 3D acquisition devices and 3D scene reconstruction research [28, 38, 39, 40, 4] have enabled large-scale acquisition of 3D scene data and this has raised

s work was conducted when Duc Thanh Nguyen was working at the Singapore University of Technology and Design.

ing [4].

a demand on 3D data analysis. However, it often happens that the 3D data cannot be obtained at high quality (as shown in Fig. 1), even by recent reconstruction methods, e.g. [4]. Specifically, the 3D surfaces are missing and/or broken and this phenomenon causes difficulties for many sequential tasks such as 3D object detection and recognition [30, 36], shape analysis [20, 19], and scene understanding {32, 12]. Repairing missing and broken surfaces thus plays a critical role and deserves in-depth study. In this paper, we focus on repairing incomplete 3D shapes. This problem can be also referred to as shape completion. We assume objects are not occluded, i.e. they can be fully observed in RGB/RGB-D images. However, this assumption does not mean that objects can be completely reconstructed.

## 1.1. Related Work

Existing shape completion approaches make use of geometric information represented at either low-level or high-level. Low-level geometry describes local structures, e.g. local smoothness, and can be used to fill small holes on broken surfaces. For example, Curless and Levoy [5] proposed to extract surfaces by examining the boundary of unseen and empty voxels. However, this method requires additional range images to carve away redundant surfaces. In {7], Davis et al. filled gaps and holes on broken surfaces by performing a convolution on the signed distance values. This process was repeated until a new implicit surface could be defined at the gaps. In [16], a broken object was represented in an octree grid on which inner and outer grid points were determined. The broken object was then constructed by contouring the grid points. In [29], holes on a broken

Running experiment with threshold = 20 and strategy = max_y:

Time taken: 3.7948 seconds
Document Angle (in degrees): 10.00

OCR Result:
This CVPE

A Field Model for Repairing 3D Shapes*

Duc Thanh Nguyen', Binh-Son Hua, Minh-Khoi Tran", Quang-Hieu Pham', and Sai-Kit Yeung?
1§chool of Information Technology, Deakin University, Australia
?Singapore University of Technology and Design, Singapore

Abstract

This paper proposes a field model for repairing 3D shapes constructed from multi-view RGB data. Specifically, we represent a 3D shape in a Markov random field (MRF) in which the geometric information is encoded by random binary variables and the appearance information is retrieved from a set of RGB images captured at multiple viewpoints. The local priors in the MRF model capture the local structures of object shapes and are learnt from 3D shape templates using a convolutional deep belief network. Repairing a 3D shape is formulated as the maximum a posteriori (MAP) estimation in the corresponding MRF. Variational mean field approximation technique is adopted for the MAP estimation. The proposed method was evaluated on both artificial data and real data obtained from reconstruction of practical scenes. Experimental results have shown the robustness and efficiency of the proposed method in repairing noisy and incomplete 3D shapes.

1. Introduction

Suppose we are given a set of RGB/RGB-D images of an object captured at multiple viewpoints. The object in the real world (i.e. 3D space) is then re-constructed using some 3D reconstruction algorithm. Ideally, if an object can be observed in RGB/RGB-D images, it can be well reconstructed. However, in reality we have found that the reconstruction often fails even if the RGB/RGB-D data is complete. This is because the matching of the RGB data in structure-from-motion based reconstruction methods (e.g. [14]) could not be done accurately, specially for objects of uniform colours. For reconstruction methods using depth (e.g. [39, 4]), the missing of depth could also cause the incompleteness. We illustrate several cases of this situation in Fig. 1.

Recent advances of 3D acquisition devices and 3D scene reconstruction research [28, 38, 39, 40, 4] have enabled large-scale acquisition of 3D scene data and this has raised

s work was conducted when Duc Thanh Nguyen was working at the Singapore University of Technology and Design.

ing [4].

a demand on 3D data analysis. However, it often happens that the 3D data cannot be obtained at high quality (as shown in Fig. 1), even by recent reconstruction methods, e.g. [4]. Specifically, the 3D surfaces are missing and/or broken and this phenomenon causes difficulties for many sequential tasks such as 3D object detection and recognition [30, 36], shape analysis [20, 19], and scene understanding {32, 12]. Repairing missing and broken surfaces thus plays a critical role and deserves in-depth study. In this paper, we focus on repairing incomplete 3D shapes. This problem can be also referred to as shape completion. We assume objects are not occluded, i.e. they can be fully observed in RGB/RGB-D images. However, this assumption does not mean that objects can be completely reconstructed.

## 1.1. Related Work

Existing shape completion approaches make use of geometric information represented at either low-level or high-level. Low-level geometry describes local structures, e.g. local smoothness, and can be used to fill small holes on broken surfaces. For example, Curless and Levoy [5] proposed to extract surfaces by examining the boundary of unseen and empty voxels. However, this method requires additional range images to carve away redundant surfaces. In {7], Davis et al. filled gaps and holes on broken surfaces by performing a convolution on the signed distance values. This process was repeated until a new implicit surface could be defined at the gaps. In [16], a broken object was represented in an octree grid on which inner and outer grid points were determined. The broken object was then constructed by contouring the grid points. In [29], holes on a broken

Running experiment with threshold = 15 and strategy = max_y:
Time taken: 3.6080 seconds

Running experiment with threshold = 20 and strategy = max_y:
#Time taken: 3.7948 seconds

[139]:
```python
#Experiment with stretegy = All
print("\nExperiment for Strategy = All")
run_experiment(threshold_values, strategy_name='all')
```

```
Experiment for Strategy = All

Running experiment with threshold = 5 and strategy = all:
Time taken: 3.0370 seconds
Document Angle (in degrees): -5.00

OCR Result:


Running experiment with threshold = 10 and strategy = all:
Time taken: 2.9414 seconds
Document Angle (in degrees): -6.00

OCR Result:


Running experiment with threshold = 15 and strategy = all:
Time taken: 2.6198 seconds
Document Angle (in degrees): -6.00

OCR Result:


Running experiment with threshold = 20 and strategy = all:
Time taken: 2.3931 seconds
Document Angle (in degrees): -7.00

OCR Result:
2
```

The best results are shown by the following set of parameters.

Threshold = 15, Strategy = max_y

These parameters give the best result in the shortest amount of time

Below are the test reults of 5 document images with the best parameters and strategies

Threshold = 15, Strategy = max_y

### 0.0.1 Testing on images with different orientation

[164]:
```
#TESTING ON LOCAL IMAGE test.jpg
# Load and process the image
image_path_test = 'test.jpg'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =␣
 ↪extract_connected_components(negative_img_test)
```

[165]:
```
#Strategy = centers
candidate_points_maxY_test = select_candidate_points(labels, stats, centroids,␣
 ↪strategy='max_y')
cleaned_negative_img_maxY_test = remove_non_candidate_points(negative_img_test,␣
 ↪candidate_points_maxY_test)
cv.imwrite('negative_image_maxY_test.png', cleaned_negative_img_maxY_test)
```

[165]:
```
True
```

[166]:
```
# Detect skew and deskew the image
document_angle_maxY_test = hough_transform(cleaned_negative_img_maxY_test,␣
 ↪threshold=15)
deskewed_image_maxY_test = deskew_image(image_path_test,␣
 ↪document_angle_maxY_test)
cv.imwrite('deskewed_image_maxY_test.png', deskewed_image_maxY_test)
```

[166]:
```
True
```

[167]:
```
#Perform OCR on skewed version
ocr_result_skewed_test = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test)
```

```
OCR Result for Skewed Image:
```

[168]:
```
# Perform OCR
ocr_result_deskewed_maxY_test = perform_ocr(deskewed_image_maxY_test)
print("\nOCR Result for Deskewed Image, strategy = Max y:")
print(ocr_result_deskewed_maxY_test)
```

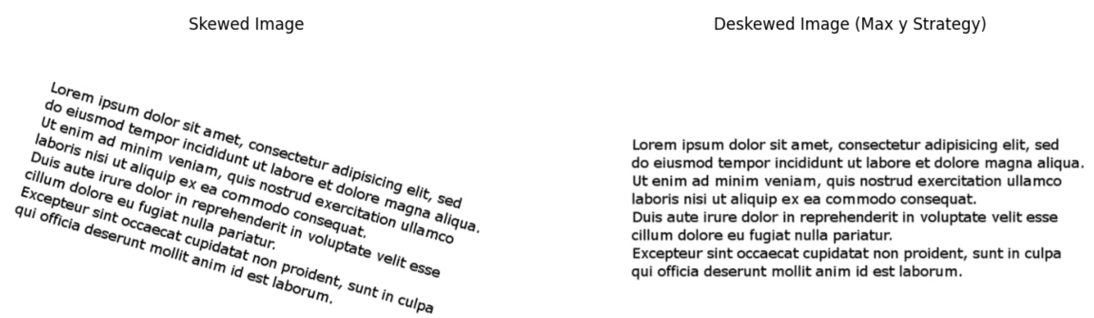
```

OCR Result for Deskewed Image, strategy = Max y:
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed

do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat.
```

Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur.

Excepteur sint occaecat cupidatat non proident, sunt in culpa
qui officia deserunt mollit anim id est laborum,

[169]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxY_test  # Deskewed image
```

[170]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```




[171]:
```python
#TESTING ON LOCAL IMAGE test2.jpg
# Load and process the image
image_path_test = 'test2.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids = extract_connected_components(negative_img_test)
```

[172]:
```python
#Strategy = centers
candidate_points_maxy_test2 = select_candidate_points(labels, stats, centroids, strategy='max_y')
cleaned_negative_img_maxy_test2 = remove_non_candidate_points(negative_img_test, candidate_points_maxy_test2)
cv.imwrite('negative_image_maxy_test2.png', cleaned_negative_img_maxy_test2)
```

[172]: True

[173]:
```
# Detect skew and deskew the image
document_angle_maxy_test2 = hough_transform(cleaned_negative_img_maxy_test2,␣
 ↪threshold=15)
deskewed_image_maxy_test2 = deskew_image(image_path_test,␣
 ↪document_angle_maxy_test2)
cv.imwrite('deskewed_image_maxy_test2.png', deskewed_image_maxy_test2)
```

[173]:
```
True
```

[174]:
```
#Perform OCR on skewed version
ocr_result_skewed_test2 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test2)
```

```
OCR Result for Skewed Image:
Photo (for example the text on sig
*t superimposed on an image (for es

" from subtitle te;

! whether from a
ns and

xample: from a

whether passport documents,
```

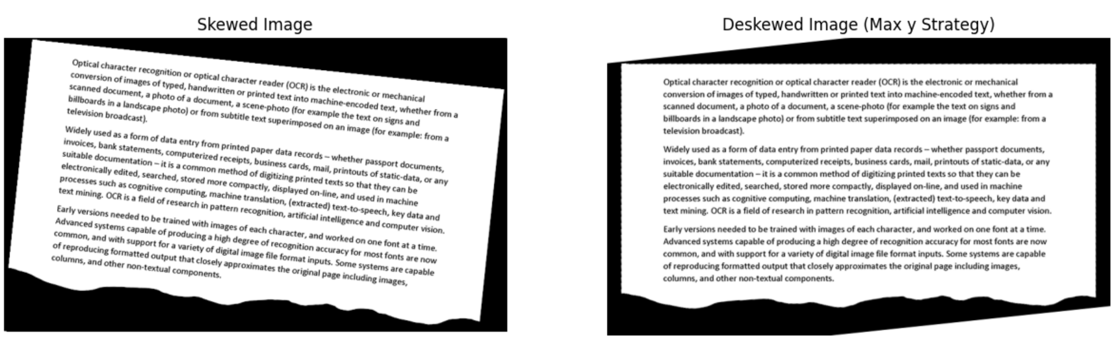
[175]:
```
# Perform OCR
ocr_result_deskewed_maxy_test2 = perform_ocr(deskewed_image_maxy_test2)
print("\nOCR Result for Deskewed Image, strategy = Max y:")
print(ocr_result_deskewed_maxy_test2)
```

```

OCR Result for Deskewed Image, strategy = Max y:
Optical character recognition or optical character reader (OCR) is the
electronic or mechanical
conversion of images of typed, handwritten or printed text into machine-encoded
text, whether froma
scanned document, a photo of a document, a scene-photo (for example the text on
signs and
billboards in a landscape photo) or from subtitle text superimposed on an image
(for example: from a
television broadcast).

Widely used as a form of data entry from printed paper data records - whether
passport documents,
```

```
invoices, bank statements, computerized receipts, business cards, mail,
printouts of static-data, or any
suitable documentation -it is a common method of digitizing printed texts so
that they can be

electronically edited, searched, stored more compactly, displayed on-line, and
used in machine
processes such as cognitive computing, machine translation, (extracted) text-to-
speech, key data and
text mining. OCR is a field of research in pattern recognition, artificial
intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on
one font at a time.
Advanced systems capable of producing a high degree of recognition accuracy for
most fonts are now
common, and with support for a variety of digital image file format inputs. Some
systems are capable
of reproducing formatted output that closely approximates the original page
including images,
columns, and other non-textual components.
```

[176]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test2  # Deskewed image
```

[177]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```



[179]:
```
#TESTING ON LOCAL IMAGE test4.png
# Load and process the image
image_path_test = 'test4.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =␣
 ↪extract_connected_components(negative_img_test)
```

[180]:
```
#Strategy = centers
candidate_points_maxy_test4 = select_candidate_points(labels, stats, centroids,␣
 ↪strategy='max_y')
cleaned_negative_img_maxy_test4 =␣
 ↪remove_non_candidate_points(negative_img_test, candidate_points_maxy_test4)
cv.imwrite('negative_image_maxy_test4.png', cleaned_negative_img_maxy_test4)
```

[180]: True

[181]:
```
# Detect skew and deskew the image
document_angle_maxy_test4 = hough_transform(cleaned_negative_img_maxy_test4,␣
 ↪threshold=15)
deskewed_image_maxy_test4 = deskew_image(image_path_test,␣
 ↪document_angle_maxy_test4)
cv.imwrite('deskewed_image_maxy_test4.png', deskewed_image_maxy_test4)
```

[181]: True

[182]:
```
#Perform OCR on skewed version
ocr_result_skewed_test4 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test4)
```

```
OCR Result for Skewed Image:
```

[183]:
```
# Perform OCR
ocr_result_deskewed_maxy_test4 = perform_ocr(deskewed_image_maxy_test4)
print("\nOCR Result for Deskewed Image, strategy = Max y:")
print(ocr_result_deskewed_maxy_test4)
```

```

OCR Result for Deskewed Image, strategy = Max y:
Ashish Vaswani* Noam Shazeer* Niki Parmar* Jakob Uszkoreit*
Google Brain Google Brain Google Research Google Research
avaswani@google.com noam@google.com nikip@google.com usz@google.com

Llion Jones* Aidan N. Gomez* * Lukasz Kaiser*
Google Research University of Toronto Google Brain
llion@google.com aidan@cs.toronto.edu lukaszkaiser@google.com
```

```
Illia Polosukhin* ?
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or
convolutional neural networks that include an encoder and a decoder. The best
performing models also connect the encoder and decoder through an attention
mechanism. We propose a new simple network architecture, the Transformer,
based solely on attention mechanisms, dispensing with recurrence and
convolutions
entirely. Experiments on two machine translation tasks show these models to
be superior in quality while being more parallelizable and requiring
significantly
less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-
to-German translation task, improving over the existing best results, including
ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task,
our model establishes a new single-model state-of-the-art BLEU score of 41.8
after
training for 3.5 days on eight GPUs, a small fraction of the training costs of
the
best models from the literature. We show that the Transformer generalizes well
to
other tasks by applying it successfully to English constituency parsing both
wit!
large and limited training data.
```

[184]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test4  # Deskewed image
```

[185]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```


[186]:
```
#TESTING ON LOCAL IMAGE test7.png, Different orientation
# Load and process the image
image_path_test = 'test7.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids = extract_connected_components(negative_img_test)
```

[188]:
```
#Strategy = max_y
candidate_points_maxy_test7 = select_candidate_points(labels, stats, centroids, strategy='max_y')
cleaned_negative_img_maxy_test7 = remove_non_candidate_points(negative_img_test, candidate_points_maxy_test7)
cv.imwrite('negative_image_maxy_test7.png', cleaned_negative_img_maxy_test7)
```

[188]: True

[189]:
```
# Detect skew and deskew the image
document_angle_maxy_test7 = hough_transform(cleaned_negative_img_maxy_test7, threshold=15)
deskewed_image_maxy_test7 = deskew_image(image_path_test, document_angle_maxy_test7)
cv.imwrite('deskewed_image_maxy_test7.png', deskewed_image_maxy_test7)
```

[189]: True

[190]:
```
#Perform OCR on skewed version
ocr_result_skewed_test7 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test7)
```

```
OCR Result for Skewed Image:
enoyncerna. vot #2 NO" _nuare@AUARY 20
```

srocn008 moo
matical Inference

Natural Language Gram
with Recurrent Neural Networks

1 IEEE, C. Lee Giles: Fellow.

eee TRANEACTIONS

IEEE, and Sandiway Fons

gteve Lawrence, Membe!
rads pe aT ea rnc car IE et seca, ve wsk consid
neat vaneg ante OORT wa anguae See etl nara emcee sae
stl ss pe lam Fag varnont of GOFOTENS ying eon. Now
oe rene we ee onroveared en org rece ST
cere anne oak pavly great ea oseaas tngushe
sy ane corn ns ai
cae ch 8 OS a feat Howe ering seve
ving comes Sayieant
ores ore sero oon oe
ae er eri gated.

oeriaues ere 8
tcl new "Seri rere
enon = Wejyre. Fry.

sjornce, goverment on

4 twrRODUCTION
vas paper corer: He of dassifying cater! ait ees a7 Ey the networks
engage semienecs 28 ramonatical OF siiamatical. and the Na whee an invests
cexteaction.
tame to ter neural Se ‘oithont the bifurea 1S paper nengacized 28 FN Bean 2
provides He
We ante, Harned ve MOE vormponents assumed PY peta oe the task ere ction 3
provides a htc
ate me, pigments 2 0 mattjacbon 1 formal STATE, eer grammatical HET
ats ae cata, ence ard dese nl ergata 4
ee jmvetigated and PONCE etails
‘presents the

produce tl

sharply ga

wa computationally,

ete networks. Setion 5 PT
rg bests nd

ating, Section S
ils. and

steal /angsa
ies are investi

gated for neural et

neural
ents of ives

Cm atonal e350
compute nore power a may oar networks
ner ee ET vad nee ena ofan Sin
bed ett) eu eg in ols an 3
Feapentis of aris PF recurrent ne presen as the operon A armors. The extraction"
papers att Ea and ee cl dT Sonata

meee, ae Walia 3 apse OWED) st gant seme

ay neo Fresno ee pene eens

ia fod at J

ret ett are ble SF spine Beata?

ore a prong <OnNET 2 MOTIVATION
* Representational Powet
onal Poway en tel OS

fang, wehnia
acs. ThE most

Miter implemen
ater mE genet cet ackpropa
gee gn ming ag NS ed echeperton Nal MESA |
terri 28 ES aly on of amt OTS, wadtontive pee
Ot Ue recurrent Hew Fras Feamed-apeci syrmgeful stochastic ANSE aye ave Deen
based O°
Carsten of es NE form of determing seen sept He aiden Markov
automat, ‘ Bris However, Anite verete cannot fepeesene
Net ge Se ‘evra tor
ie ve eer, toocn sed FOF

```
fost, Recurrent

vackine teasing, paredisims this probl
cpventigates pore fens OP (2h, 19. 2,
i eratler natura

other m
veer focuses on fecuent eu networks,
oe : Bepal networks have Bem
"em eat ae th NEC Bm tate, # tare Wa neural me problems, 68 PTT the Elman nets
Sans language sks INE TOL 2h 28), (58) Ch
fo rat network models have BESS setpwan to be able 1

Pac MoS
nates
ead 1 nes 1386

spec
et 19 Spl 1997

nee

a
1 ie
van een 0 oe Ma mea

i ie nae 3A ahem er

fe

roe 8
```

[191]:
```python
# Perform OCR
ocr_result_deskewed_maxy_test7 = perform_ocr(deskewed_image_maxy_test7)
print("\nOCR Result for Deskewed Image, strategy = Max Y:")
print(ocr_result_deskewed_maxy_test7)
```

```
OCR Result for Deskewed Image, strategy = Max Y:
28 EGE TRANSACTIONS ON KNOWLEDOE AND DATA ENGNGERING, VOL 12, NO. 1

Natural Language Grammatical Inference
with Recurrent Neural Networks

Steve Lawrence, Member, IEEE, C. Lee Giles, Fellow, IEEE, and Sandiway Fong

[Abstract-This pape examin te induc inference ot complex yamvmar wit court
```

netwons--epocticaly. the ask considered
‘Sita of raning« etwor to casily natural angatgesectences as Grarmatcl cr ungrammatal Crereby exhating the same kid
‘ot ascomnatoy power proved by the Prtieplee and Parameers lagu tamewank of Govereen-and Sieg theory. Nora!
‘etworks ar vaied, wou te vison mo tearied\e nate components assumed by Chomsky, nbn altempt to produce he ame
yhgmeonts as nave epaakors on thal gamumatcarungrarmascal data. How a recent neva netwck coud postass Ingus
Capabity andthe properties of variove common recurs mural tatwork arctan are discussed. Te problom exis Waning
tenavor whch 9 te oot erecent wah emaber grammare an! Waning wes may afeut. Hower, ater implementing sovral

feoheiques emad at mgroung te convergence Otte grant deccant backpropagation tice Varing ogo, giant
teareng was poses i was fod that ceran archaecures we baer abot lear an appropriate gramvnar. The operat of he
‘lors und ne tiring is anlyze Fay, he extractor of ras inthe or cl determine ee sate auomotn 8 vesigated,

Incex Terme- Recurrent naurl networks, natal lnguadge processing, grmmtic!arene, govementandindng Meo
(alent Joscet, smulated annealing, princpls and parameters lramework, automata exracten.

1. itropuction

ss paper considers the task of classifying. natural language sentences as grammatical or ungratnanatical We attempt to train neural networks, without the bifurea tion into Jearned vs. innate components assumed by Chomsky, to produce the same judgments as ati speakers on sharply geamvmatical/ungammatical data Only recurrent neutral networks are investigated for ‘computational reasons. Computationally, recurrent neural networks are more powerful than feudforward networks and some recurrent architectures have been shown to be at Feast Turing equivalent [53], [54]. We investigate the properties of various popubir recurrent neural network architectures, in particular Elman, Narendra and Parthasor athy (N&P), and Williams and: Zipser (W&Z) recurrent networks, and also Prasconi-GoriSoda (FGS) locally recur- rent networks, We find that both Elman and WEz recurrent networks ane able to lear ait appropriate grammar ater implementing. techniques for improving the conver agence of the gradient descent based. backpropagation- through-lime training algorithm, We analyze the operation fof the networks and investigate a rule approximation of What the recurrent network has Ieanved-specifially, the ‘extraction of rues in the form of deterministic finite state Sutomate

Previous work {38] has compated neural networks with

other machine learning paradigms on this problem-this work focuses on recurrent neueal networks, investigates

45 The ators are sth NEC Reels ite
Princ, POS
a: ae

+ tadgrdence Way

Marnie reid 1 8
hoe

1936 rei 19 Sip, 1997; ap 24 Feb

sie of vs tick, pee te a
apron ud ef? IELECS Un Num TORS,

are

additional networks, analyzes the operation of the networks land the training algorithm, and investigates rule extraction

This paper is organized as follows: Section 2 provides the motivation forthe task attempted. Section 3 provides a brief introduction to formal grammars and grammatical it cence and describes the data, Section 4 lists the recurrent neural network models investigated and provides details ob the data encoding for the networks, Section 5 presents the esults of investigation into various training heuristics and investigation of training with simulated annealing, Section 6 presents the main results and simulation details and fawestigates the operation of the networks. The extraction ‘of nles in the form of deterministic finite state automata is investigated in Scction 7 and Section 8 presents a discussion of the results and conclusions.

## 2 Motivation

### 2.1 Representational Power

Natural language has traditionally been handled using symbolic computation and recursive processes. The most

```
successful stochastic language models have been based on
finite-state descriptions such as n-grams ov hidden Markov
models. However, finite-state models cannot represent
hierarchical structures as found im natural language! 48}.
In the past few years, several recurrent neural network
architectures have emerged whieh have been used for
‘grammatical inference (9), {21}, 119} {20}, {68}, Recurrent
neural networks have been vised for several smaller natural
language problems, eg. papers using the Elman network
foe natural language tasks include: {0}. (12) 24), (58), (9)
Neural network models have been shown to be able to

1, Te imide we seat aleron 6 exten of Nom
"Motta intss beter areal tem Tae
‘pottery ny precede rete sou ron 8h
```

[193]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test7  # Deskewed image
```

[194]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```

[195]:
```python
#TESTING ON LOCAL IMAGE test10.jpg, Different orientation
# Load and process the image
image_path_test = 'test10.jpg'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =␣
 ↪extract_connected_components(negative_img_test)
```

[196]:
```python
#Strategy = max_y
candidate_points_maxy_test10 = select_candidate_points(labels, stats,␣
 ↪centroids, strategy='max_y')
cleaned_negative_img_maxy_test10 =␣
 ↪remove_non_candidate_points(negative_img_test, candidate_points_maxy_test10)
cv.imwrite('negative_image_maxy_test10.png', cleaned_negative_img_maxy_test10)
```

[196]: True

[197]:
```python
# Detect skew and deskew the image
document_angle_maxy_test10 = hough_transform(cleaned_negative_img_maxy_test10,␣
 ↪threshold=15)
deskewed_image_maxy_test10 = deskew_image(image_path_test,␣
 ↪document_angle_maxy_test10)
cv.imwrite('deskewed_image_maxy_test10.png', deskewed_image_maxy_test10)
```

[197]: True

[198]:
```python
#Perform OCR on skewed version
ocr_result_skewed_test10 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test10)
```

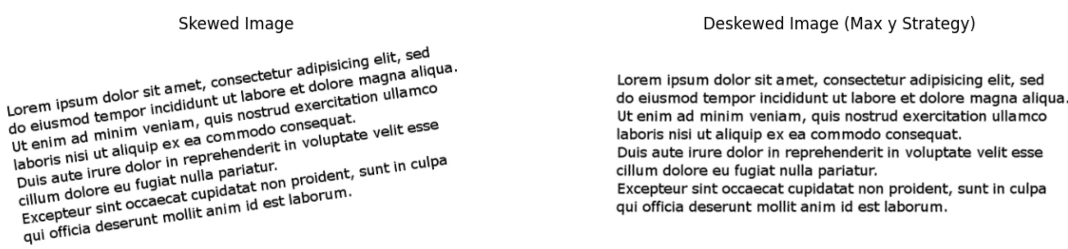
```
OCR Result for Skewed Image:
Lorem ipsum dolor sit amet, consectetur radii ng elit,
do eiusmod tempor incididunt ut labore et det magna qua.
utenim ad minim veniam, quis nostrud ex! piorsuon ullamco

\aboris nisi ut aliquip eX ea commodo cons' sequal
Duis aute irure dotor in reprenen nderit in woluptate te velit esse

excepteur sint oc aecat cupidatat non prol roident, sunt in culpa
qui officia deserunt mollit anim id est Jaborum.
```

[199]:
```python
# Perform OCR
ocr_result_deskewed_maxy_test10 = perform_ocr(deskewed_image_maxy_test10)
print("\nOCR Result for Deskewed Image, strategy = Max Y:")
print(ocr_result_deskewed_maxy_test10)
```

```
OCR Result for Deskewed Image, strategy = Max Y:
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed

do eiusmod tempor incididunt ut labore et dolore magna aliqua.
Ut enim ad minim veniam, quis nostrud exercitation ullamco
laboris nisi ut aliquip ex ea commodo consequat.

Duis aute irure dolor in reprehenderit in voluptate velit esse
cillum dolore eu fugiat nulla pariatur.

Excepteur sint occaecat cupidatat non proident, sunt in culpa
qui officia deserunt mollit anim id est laborum.
```

[200]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test10  # Deskewed image
```

[201]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```




[ ]:
```python
#Testing on different Language
```

[202]:
```python
#TESTING ON LOCAL IMAGE test6.png, Different language
# Load and process the image
image_path_test = 'test6.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =␣
↪extract_connected_components(negative_img_test)
```

[204]:
```python
#Strategy = centers
candidate_points_maxy_test6 = select_candidate_points(labels, stats, centroids,␣
↪strategy='max_y')
```

```
cleaned_negative_img_maxy_test6 =␣
  ↪remove_non_candidate_points(negative_img_test, candidate_points_maxy_test6)
cv.imwrite('negative_image_maxy_test6.png', cleaned_negative_img_maxy_test6)
```

[204]: True

[206]:
```
# Detect skew and deskew the image
document_angle_maxy_test6 = hough_transform(cleaned_negative_img_maxy_test6,␣
  ↪threshold=10)
deskewed_image_maxy_test6 = deskew_image(image_path_test,␣
  ↪document_angle_maxy_test6)
cv.imwrite('deskewed_image_maxy_test6.png', deskewed_image_maxy_test6)
```

[206]: True

[ ]:
```
'''
For this case I had to change the threshold value as the chosen value of 15 is␣
  ↪too high for the lines to be detected by cv.HoughLines().
'''
```

[207]:
```
#Perform OCR on skewed version
ocr_result_skewed_test6 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test6)
```

```
OCR Result for Skewed Image:
SC, dey
le 4
ow. Wigs we aS
50 Rasta )

4 Sy, ie 3
Pa "diag Se 2 ais, oes
Mg ha tLe See,
ea ie fe ag, he, WI ae 8s iy
tos iy ie 4 ls SS, om, Shy
```

[210]:
```
# Perform OCR
ocr_result_deskewed_maxy_test6 = perform_ocr(deskewed_image_maxy_test6)
print("\nOCR Result for Deskewed Image, strategy = Max Y:")
print(ocr_result_deskewed_maxy_test6)
```

```

OCR Result for Deskewed Image, strategy = Max Y:
gg phe a Taba gi da rine sot gale Jlge a g9ad * oe 59 sob Ian baat
gileins tty Ja Sarbeal a SS j 98 Nagar Surg al ped ppaloar de sachs
plan (254 eS 8a glide Ilion 2 Sy ae grt in tnged ulelen
```

```
PAI, a gh SI g4 9A S tpn Ba gsbl 9 SI gh lal ay had of Slew pr a5
Pen ge Saab | ln AE Ay gt Misleme My sty gel + geal Gat 9S
"apd gle

Be ie et He AS ald Galle sl les eS ile gn?
yd Shee Sed Seb ag LD SOS 9 si Ai cs gt 5S seed 39-099 es
Getyglys eal pened oF 2 MS sl alse So9? I eal sl got dail sl got
sesh bs ge rssh ITP oe ged FOIE irs ET grey oe 4g?
Oso p tetyhe al gtd eany hay 2 alle 0S 005 gh Gl ay MSD 395 sit 2959
DSS ey sal gt 6 pats ee bl porte shyt ghd cllgeg Kix ee
Pia al gh hee 2 ge Go ag ee gay AaB Dg tl oe Lb 35 9 gab pet aS
eh gt HAS Se co tbe) lew Sts - gabe lyst ah Sade tetas Kea, '
shew gle ag olajas NBS bas hoh Te Tipe Applian dees

Aadheg ell
```

[211]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test6  # Deskewed image
```

[212]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```


For different language, Houghman transformation was able to deskew the image, but OCR writes gibberish language.

### 0.0.2 Nosie Elements in Dosuments

[213]:
```
#TESTING ON LOCAL IMAGE test3.jpg
# Load and process the image
image_path_test = 'test3.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =␣
↪extract_connected_components(negative_img_test)
```

[214]:
```
#Strategy = centers
candidate_points_maxy_test3 = select_candidate_points(labels, stats, centroids,␣
↪strategy='max_y')
cleaned_negative_img_maxy_test3 =␣
↪remove_non_candidate_points(negative_img_test, candidate_points_maxy_test3)
cv.imwrite('negative_image_maxy_test3.png', cleaned_negative_img_maxy_test3)
```

[214]:
```
True
```

[215]:
```
# Detect skew and deskew the image
document_angle_maxy_test3 = hough_transform(cleaned_negative_img_maxy_test3,␣
↪threshold=15)
deskewed_image_maxy_test3 = deskew_image(image_path_test,␣
↪document_angle_maxy_test3)
cv.imwrite('deskewed_image_maxy_test3.png', deskewed_image_maxy_test3)
```

[215]:
```
True
```

[216]:
```
#Perform OCR on skewed version
ocr_result_skewed_test3 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test3)
```

```
OCR Result for Skewed Image:
I) Rc oti
Desr.

image. Can help 4 log ip
nt, nd do, OCR G OMR R, barcodes

detection or ost eaPrOve the

Feadabin;" ity Ned j ima
```

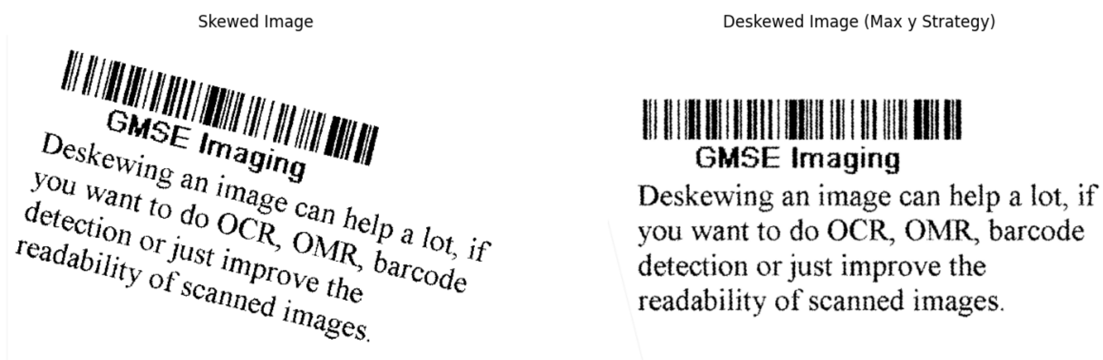
[219]:
```
# Perform OCR
ocr_result_deskewed_maxy_test3 = perform_ocr(deskewed_image_maxy_test3)
print("\nOCR Result for Deskewed Image, strategy = Max y:")
```

```
print(ocr_result_deskewed_maxy_test3)
```

```
OCR Result for Deskewed Image, strategy = Max y:
l . Wid MA AMIN

E Imaging
Deskening an image can help a lot, if
you want to do OCR, OMR, barcode
detection or just improve the
readability of scanned images.
```

[220]:
```
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test3  # Deskewed image
```

[221]:
```
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```




The transformation for noise value(barcode) was successfull.
The Document image has a barcode, It is detected as, l . Wid MA AMINE, The rest of the text is accurately detected

[222]:
```
#TESTING ON LOCAL IMAGE test5.png
# Load and process the image
image_path_test = 'test5.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =
 ↪extract_connected_components(negative_img_test)
```

[223]:
```python
#Strategy = centers
candidate_points_maxy_test5 = select_candidate_points(labels, stats, centroids,␣
    ↪strategy='max_y')
cleaned_negative_img_maxy_test5 =␣
    ↪remove_non_candidate_points(negative_img_test, candidate_points_maxy_test5)
cv.imwrite('negative_image_maxy_test5.png', cleaned_negative_img_maxy_test5)
```

[223]: True

[224]:
```python
# Detect skew and deskew the image
document_angle_maxy_test5 = hough_transform(cleaned_negative_img_maxy_test5,␣
    ↪threshold=15)
deskewed_image_maxy_test5 = deskew_image(image_path_test,␣
    ↪document_angle_maxy_test5)
cv.imwrite('deskewed_image_maxy_test5.png', deskewed_image_maxy_test5)
```

[224]: True

[225]:
```python
#Perform OCR on skewed version
ocr_result_skewed_test5 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test5)
```

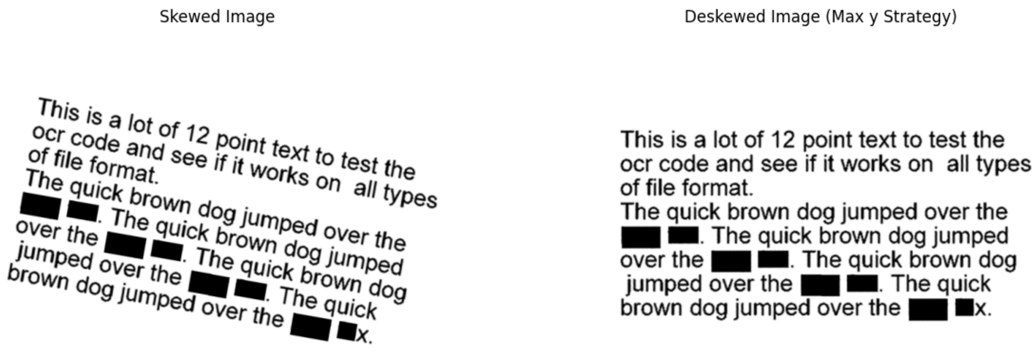
```
OCR Result for Skewed Image:
```

[226]:
```python
# Perform OCR
ocr_result_deskewed_maxy_test5 = perform_ocr(deskewed_image_maxy_test5)
print("\nOCR Result for Deskewed Image, strategy = Max Y:")
print(ocr_result_deskewed_maxy_test5)
```

```

OCR Result for Deskewed Image, strategy = Max Y:
This is a lot of 12 point text to test the
ocr code and see if it works on all types
of file format.

The quick brown dog jumped over the
Wl WB. The quick brown dog jumped
over the J MMM. The quick brown dog
jumped over the [ij MM. The quick
brown dog jumped over the Jj Mx.
```

[227]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test5  # Deskewed imagea
```

[228]:
```
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```




Transformation was successful as the noise elements(strikethrough words) were descewed using Houghman transformation In this case, there were some wordswhich were strikethrough with thick black layer, these words were detected as Wl, WB, J, MMM, ij, MM, Jj Mx.

[229]:
```
#TESTING ON LOCAL IMAGE test8.png, Different orientation
# Load and process the image
image_path_test = 'test8.png'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids =␣
 ↪extract_connected_components(negative_img_test)
```

[230]:
```
#Strategy = max_y
candidate_points_maxy_test8 = select_candidate_points(labels, stats, centroids,␣
 ↪strategy='max_y')
cleaned_negative_img_maxy_test8 =␣
 ↪remove_non_candidate_points(negative_img_test, candidate_points_maxy_test8)
cv.imwrite('negative_image_maxy_test8.png', cleaned_negative_img_maxy_test8)
```

[230]: True

[231]:
```
# Detect skew and deskew the image
document_angle_maxy_test8 = hough_transform(cleaned_negative_img_maxy_test8,␣
 ↪threshold=15)
deskewed_image_maxy_test8 = deskew_image(image_path_test,␣
 ↪document_angle_maxy_test8)
cv.imwrite('deskewed_image_maxy_test8.png', deskewed_image_maxy_test8)
```

[231]: True

[232]:
```python
#Perform OCR on skewed version
ocr_result_skewed_test8 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test8)
```

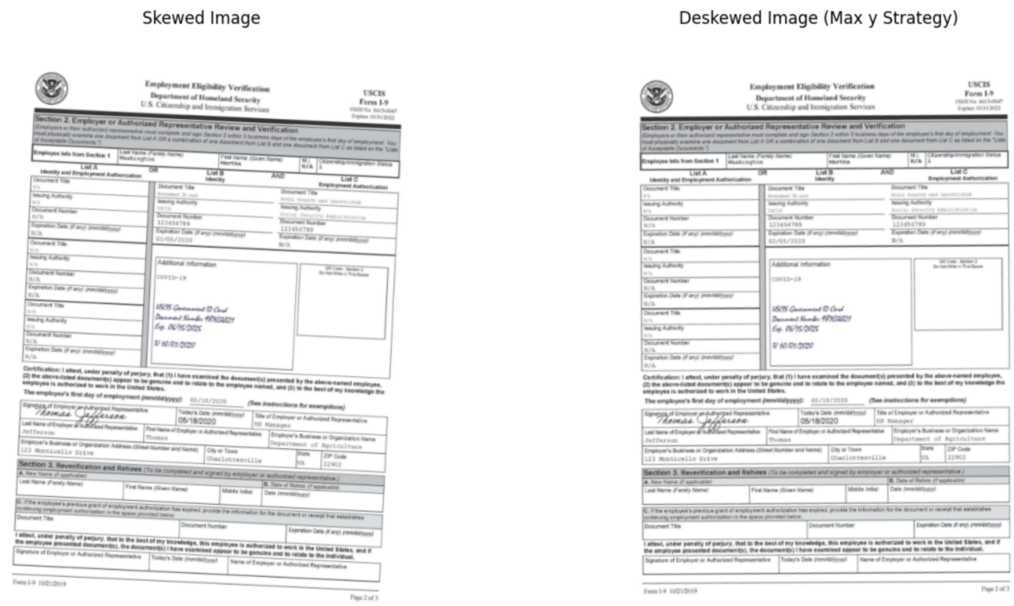
```
OCR Result for Skewed Image:
Department of Homeland Security
US. Citznship and Immigration Serces

uscis
Form 1-9

eae

blepe deed

SCS Gnmen 1 Cad

cciment) resented byte atovesaned employe,
move names snd) we best ay as te

[err eas w Spina Kane

Soa 3 Specs Aan Gone OR SST

harloceavinte

oepartnene of Agereulcure

ar come

Bors tee eres

Resse

Fem E91aaia019

Paezars
```

[233]:
```python
# Perform OCR
ocr_result_deskewed_maxy_test8 = perform_ocr(deskewed_image_maxy_test8)
print("\nOCR Result for Deskewed Image, strategy = Max Y:")
print(ocr_result_deskewed_maxy_test8)
```

```
OCR Result for Deskewed Image, strategy = Max Y:
[cae ta a IT Tei TT

ores 72038 wr

1595 cane ak
pumerery
es conse

vs0r9/2000

‘ericnton atest unde pony of paruy at) have examined fe desument( presente Se
abovenaned empoye,
Sine tenis sce pn gnu tm unm be cmp oso) e my

(See msrvceors tr evento)

an Resene --[acaya mevan) [Tin ol oy = ahr Reweais
Z osr182020 i Nanager

ate op pment en = nae apeatn [ervey Bares Opranin Nase

settereen on locparemene of hactoulvuce

Teneo can pn Kanes Stee! me) [aya Tome call dol

12) mtscelto oeive [Suarioccenvitte 22002

aap aan eae nae In nned HARTA Sora AT
[veer Te [Deon omer Son Som anh mT

‘anos ond pay of pay, ate a best wonidge lk ompoye s subordinate a
‘os roves estes aorumane te Socunest have snamned spew be Gene tnd ete Bw ea

Sian Ege ered Reena [Warm pore ond yee

Fen? ai Poet
```

[234]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test8  # Deskewed imagea
```

[235]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```




The houghman Transformation is successful in this case, the tabels were deskewed properly along with the texts of different sizes.
However, due to poor resolution, the OCR was unable to detect the text properly

[236]:
```python
#TESTING ON LOCAL IMAGE test9.jpg, Different orientation
# Load and process the image
image_path_test = 'test9.jpg'
binarized_test = load_and_binarize_image(image_path_test)
negative_img_test = get_negative_image(binarized_test)
num_labels, labels, stats, centroids = extract_connected_components(negative_img_test)
```

[237]:
```python
#Strategy = max_y
candidate_points_maxy_test9 = select_candidate_points(labels, stats, centroids, strategy='max_y')
cleaned_negative_img_maxy_test9 = remove_non_candidate_points(negative_img_test, candidate_points_maxy_test9)
cv.imwrite('negative_image_maxy_test9.png', cleaned_negative_img_maxy_test9)
```

[237]: True

[238]:
```python
# Detect skew and deskew the image
document_angle_maxy_test9 = hough_transform(cleaned_negative_img_maxy_test9, threshold=15)
deskewed_image_maxy_test9 = deskew_image(image_path_test, document_angle_maxy_test9)
```

```python
cv.imwrite('deskewed_image_maxy_test9.png', deskewed_image_maxy_test9)
```

[238]: True

[239]:
```python
#Perform OCR on skewed version
ocr_result_skewed_test9 = perform_ocr(cv.imread(image_path_test))
print("OCR Result for Skewed Image:")
print(ocr_result_skewed_test9)
```

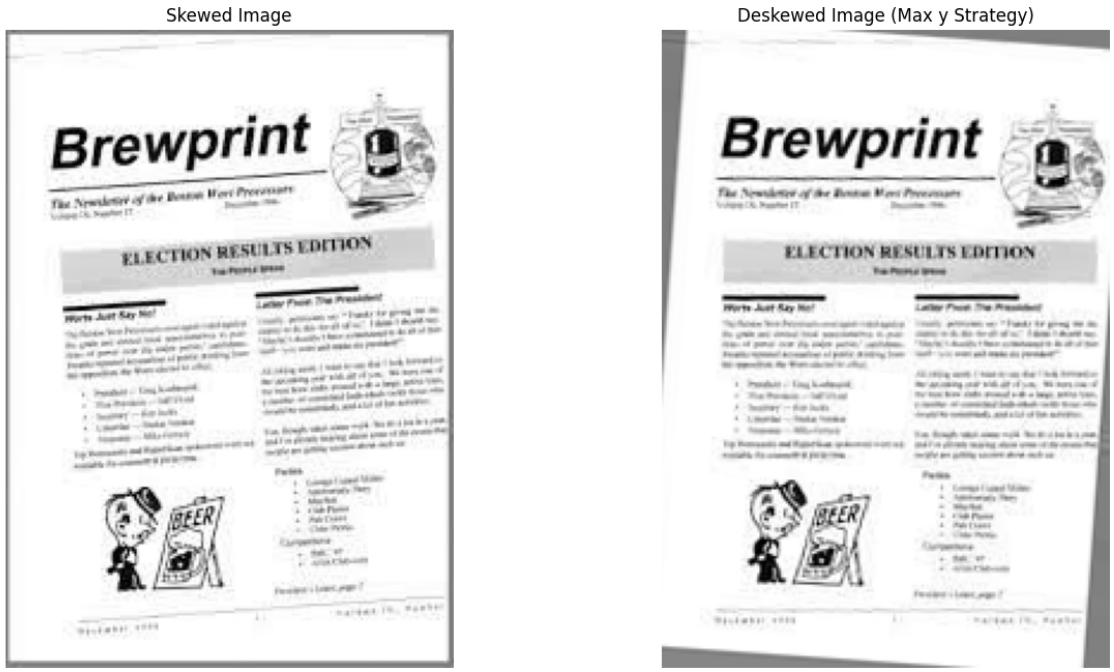
```
OCR Result for Skewed Image:
Brewprint : 38
```

[240]:
```python
# Perform OCR
ocr_result_deskewed_maxy_test9 = perform_ocr(deskewed_image_maxy_test9)
print("\nOCR Result for Deskewed Image, strategy = Max Y:")
print(ocr_result_deskewed_maxy_test9)
```

```
OCR Result for Deskewed Image, strategy = Max Y:
```

[243]:
```python
# Load the images to display
skewed_image = cv.imread(image_path_test)  # Skewed image (original)
deskewed_image = deskewed_image_maxy_test9  # Deskewed imagea
```

[244]:
```python
# Display the skewed and deskewed images using matplotlib
images = [skewed_image, deskewed_image]
titles = ['Skewed Image', 'Deskewed Image (Max y Strategy)']
display_images(images, titles)
```




The transformation is successful, all textual and non textual elements were skewed along with texts of different sizes.