画像処理適用のための
検証作業支援ツールの開発

電子・情報科　主任研究員　○鈴木健司
研究員　三瓶史花

質問はメールにて事務局までお気軽にお問い合わせください。
問い合せ先：福島県ハイテクプラザ　企画連携部産学連携科
e-mail：hightech-renkei@pref.fukushima.lg.jp

福島県ハイテクプラザ
FUKUSHIMA TECHNOLOGY CENTRE

背　景

○申請企業：クリナップ株式会社

住宅設備機器向けの素材に関する研究開発を行っており、画像処理を用いた評価手法の新規開発にも取り組もうとしている。

画像処理とは

画像の形状や色を加工・調整したり、情報を抽出したりすることを指す。一般的にはコンピュータを使用したデジタル画像処理を言う。研究開発分野では、実験結果の分析や定量評価への応用も考えられる技術である。

コンピュータでデジタル処理するため、プログラミングスキルがほぼ必須…

プログラミングを習得しないと自分で様々な画像処理の効果を確認することができない。

目　的

画像処理の一般的な課題

1.画像処理に関する知識が必要。

2.プログラミングスキルが必要。

3.画像処理のパラメータ調整など検証作業に手間がかかる。

4.処理結果をどう使うか。

（発報、ロボット制御など）

今回解決する課題の抽出

実現したい支援内容：

1.プログラミング初学者でも画像処理が実施できる。

2.簡単な操作で画像処理のパラメータ調整ができる。

3.画像処理に関するプログラミングスキルを習得してもらう。

目的

プログラミング初学者でも各種画像処理アルゴリズムの検証作業を可能とする、マウス操作のみで画像処理を実行できる支援ツールを開発する。

発表の流れ

1. 実装する画像処理アルゴリズム
→ ・平滑化（ぼかし）、鮮鋭化、グレースケール化と
二値化、エッジ検出

2.支援ツールの設計・製作
→ ・Python + TkinterによるGUIアプリケーション

3. 実際の使用感
→ ・操作動画
・コード画面

１．実装する画像処理アルゴリズム

画像処理の基本的なアルゴリズムを実装：

・**平滑化（ぼかし）**
→平滑化（ぼかし）を行う。ノイズ除去の効果がある。

・**鮮鋭化**
→ ぼやている輪郭（エッジ）を際立たせる効果。

・**グレースケール化**
→ カラー画像をモノクロ化する。二値化などの前処理に使われる。

・**二値化**
→各ピクセル値を閾値の大小で**0**（黒色）、**255**（白色）に分ける。

・**エッジ検出**
→画像のエッジ部分を検出し、輪郭線のみの二値化画像を得ることができる。

２．支援ツールの設計・製作

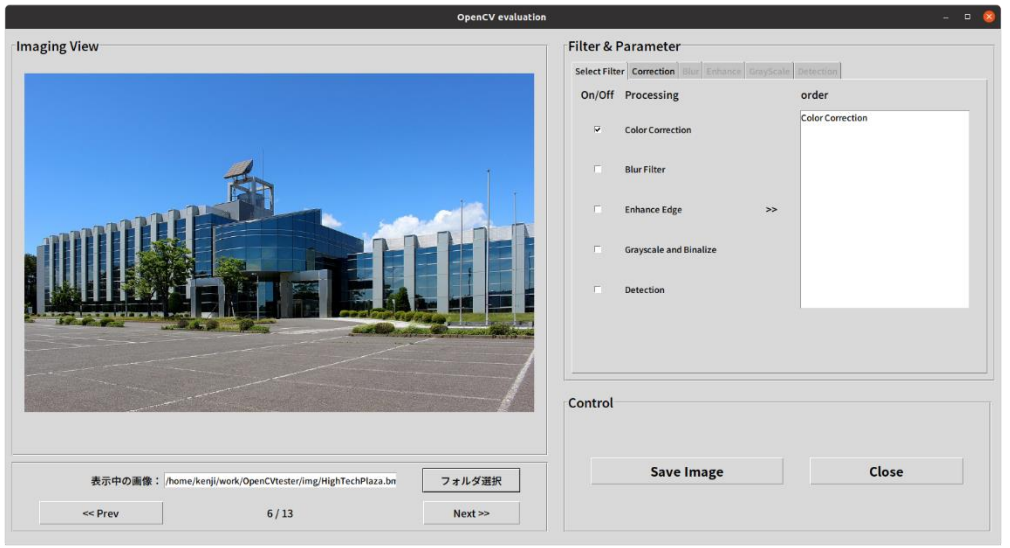
Python ＋ TkinterによるGUIアプリケーション：

・使用するプログラミング言語はPython。
・GUIアプリケーション(マウス操作のみで実行可)とし、Tkinterライブラリを使用。

python™

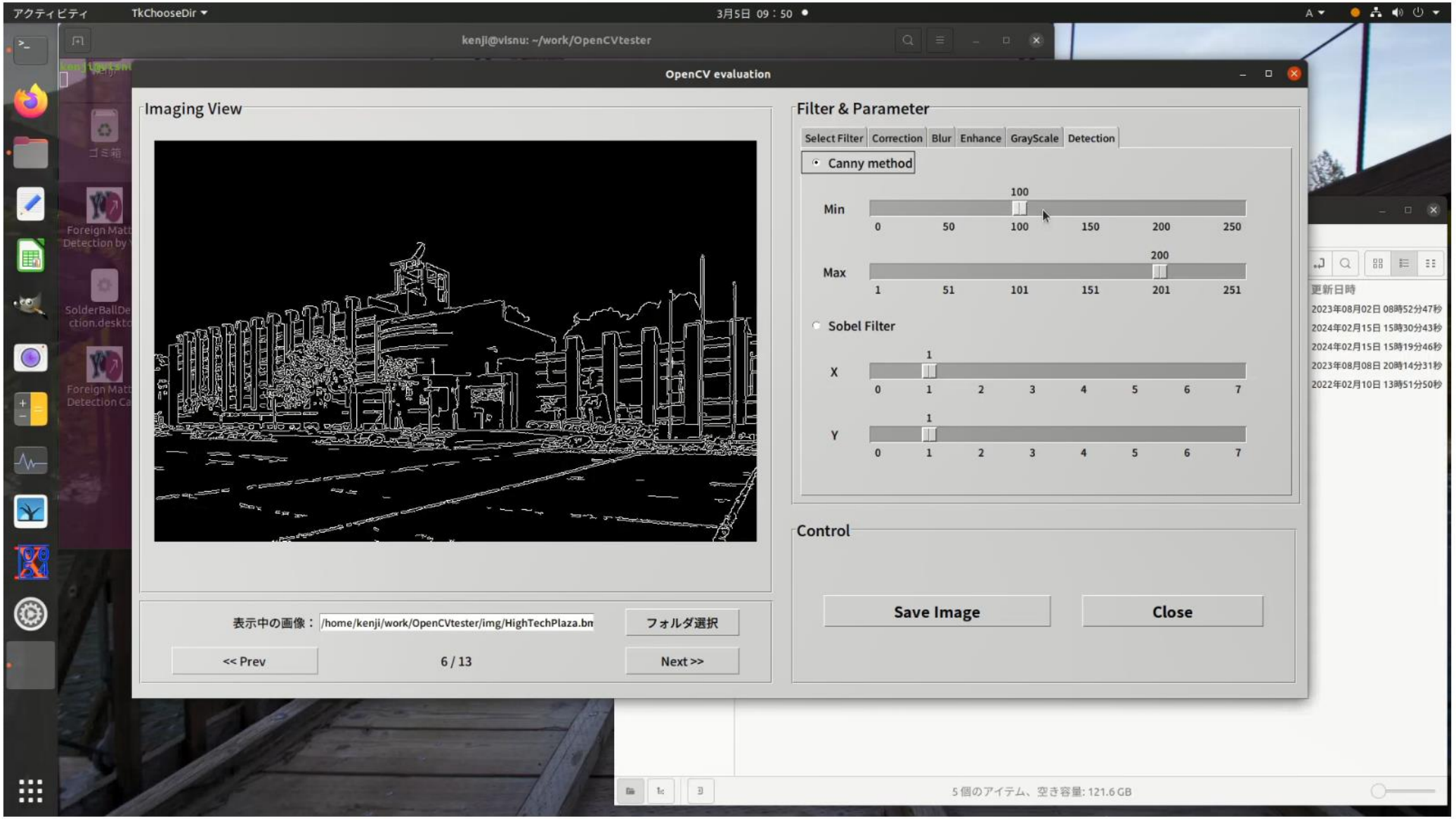
＋

【 tkinter 】
ライブラリ



GUIアプリケーションの
作製が可能！

３．実際の使用感 – 操作画面



３．実際の使用感 – コード画面

開く(O)　OpenCVtester.py　~/work/OpenCVtester　保存(S)

```
895     def activate_mode(self, inp_img):
896         if self.listbox.size() > 0:
897             active_filters = self.listbox.get(0, self.listbox.size()-1)
898             for i in active_filters:
899                 #filters = ['Color Correction','Blur Filter','Enhance Edge','Grayscale and Binalize','Detection']
900                 if i == 'Color Correction':
901                     inp_img = self.make_color_Correction(inp_img)
902                 if i == 'Blur Filter':
903                     knl_size_blur = self.kernel_size_blur.get()
904                     inp_img = cv2.blur(inp_img, (knl_size_blur, knl_size_blur))
905                 if i == 'Enhance Edge':
906                     knl_size_edge = self.kernel_size_edge.get()
907                     kernel = self.make_sharp_kernel(knl_size_edge)
908                     inp_img = cv2.filter2D(inp_img, -1, kernel).astype("uint8")
909                 if i == 'Grayscale and Binalize':
910                     inp_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)
911                     if self.flt3_bin.get() == True:
912                         thres = self.binalize_threshold.get()
913                         ret, inp_img = cv2.threshold(inp_img, thres, 255, cv2.THRESH_BINARY)
914                     inp_img = cv2.cvtColor(inp_img, cv2.COLOR_GRAY2BGR)
915                 if i == 'Detection':
916                     if self.flt4_a.get() == True:
917                         inp_img = cv2.Canny(inp_img, self.MinVal.get(), self.MaxVal.get())
918                     else:
919                         inp_img = cv2.cvtColor(inp_img, cv2.COLOR_BGR2GRAY)
920                         inp_img = cv2.Sobel(inp_img, cv2.CV_8U, self.x_dir.get(), self.y_dir.get(), 1)
921                     inp_img = cv2.cvtColor(inp_img, cv2.COLOR_GRAY2BGR)
922         return inp_img
923
```

Python　タブ幅: 8　(894 行、1 列)　[挿入]

まとめ

○画像処理の検証作業支援ツールを作製した。

- マウス操作のみの簡単な操作で各種画像処理を適用できる。
- 画像処理のパラメータはスライドバーで変更可能で、適用結果は即時画面に反映される。
- Pythonで実装され、コーディング部分の記述など確認が可能。

画像処理の初学者向けに効率的な検証作業とプログラミング習熟のアシスト。

今後の展望

作製した支援ツールを活用し、画像処理スキルを身に着けてもらい具体的な事例で定量評価できるよう支援する。