ULTRACOLD ATOM IMAGE ANALYSIS GUI

Carlo Sanchez and Maren E. Mossman
Department of Physics and Biophysics, University of San Diego

Abstract: Ultracold atoms, or atoms that have been cooled to temperatures very near absolute zero, are used to study a variety of fundamental physics concepts. These atoms behave quantum mechanically, which means that the atoms obey a different set of rules that are well defined by the theory of quantum mechanics. We manipulate these macroscopic systems by applying time-dependent magnetic and optical fields. After completing an experiment, we use absorption imaging techniques to image how the atoms have responded to these specific experimental parameters. These images are taken with a CCD camera and are passed back into the computer interface, where they need to be uploaded and quickly analyzed. In this project, we have developed a graphical user interface (GUI) using the Tkinter package in Python, which allows custom design and programming to suit the user's needs. For our applications, the GUI features Gaussian and Thomas-Fermi fitting functionality for atom number calibration, a zoom-in/zoom-out function to isolate the data, and automatic cross-sectional plots used to visualize the atomic density of the BEC in a two-dimensional space. In this poster, we will discuss the current status of the project as well as future directions for implementing this GUI in the lab.

Ultracold Atoms at USD

At the Quantum Hydrodynamics Lab at USD, we start with a hot vapor of rubidium atoms and cool them down using a sequence of applied lasers and magnetic fields. This macroscopic collection ultracold atom follow the laws governed by quantum mechanics, which describes physics at the most fundamental level. At the end of our experiments, we measure the system by taking an image of the atoms. A graphical user interface (GUI) is utilized to process and analyze data extracted from these experiments.

Physics of Absorption Imaging

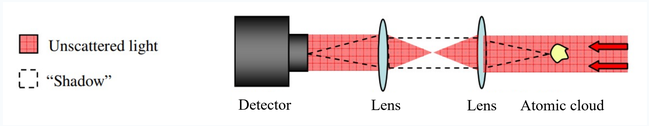

Figure 1 Sample schematic of absorption imaging. Picture depicts shadow left behind by the light beam. Image from Ref. [1].

In this setup, cold atoms are imaged via absorption imaging (Fig. 1), where on-resonant light is shined onto our system of atoms. Because the light is on-resonant, the atoms absorb the photons, appearing as a shadow onto the camera CCD. This shadow varies based on the density of atoms and is known as the optical density (OD). The intensity in the image is given by Beer's Law,

$$I = I_0 e^{-\mathrm{OD}},$$

where I is the intensity of the light on the CCD and I_0 is the background intensity from the probe beam. The measured optical density is defined by [2]

$$\mathrm{OD}_{\mathrm{meas}} = \ln\left(\frac{I_{\mathrm{light}} - I_{\mathrm{dark}}}{I_{\mathrm{shadow}} - I_{\mathrm{dark}}}\right),$$

where the "light", "dark" and "shadow" intensities are show in Fig. 2.

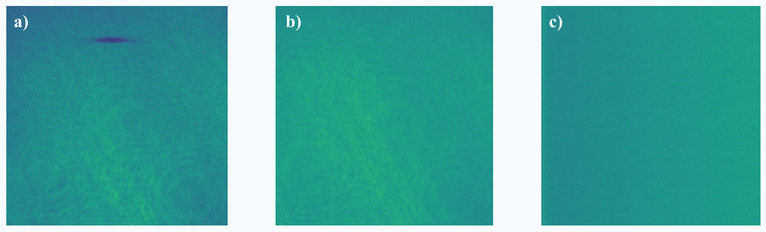

Figure 2 Typical images taken with a CCD camera in the lab, where the (a) shadow frame, (b) light frame, and (c) dark frame are shown. Images curtesy of the Engels lab at WSU.

Python-based Graphical User Interface (GUI)

The GUI was compiled and programmed with python, an open-source, free programming language. Example of the source code and GUI are shown in Fig. 3 and Fig. 4.

Constructing the GUI involved learning about and developing skills in:

- Object-oriented programming;
- Utilizing pre-made python packages (Tkinter);
- Importing different types of files into the system;
- Introducing files as features (plotting different types of imported data);
- Implementing complex fit functions for various types of experiments (Lmfit package);
- Incorporating applications of linear algebra to improve the integrated features in the GUI.

```
def zoom(self):
    """
    Takes coordinates obtained from click and drag rectangle
    and creates new image
    """
    zoom_rec = self.rectangle

    #Changes Aspect ratio of the image to a square
    dx = zoom_rec[:,1] - zoom_rec[:,0]
    dy = zoom_rec[:,3] - zoom_rec[:,2]

    #Check which side of the image selected is larger
    if dx > dy:
        #Splits add data to 'strech' image into a square
        delta_y1 =(dx - dy)//2
        delta_y2 = int(np.ceil((dx-dy)/2))
        zoom_rec[:,2] = zoom_rec[:,2] - delta_y1
        zoom_rec[:,3] = zoom_rec[:,3] + delta_y2


    #Coordinates cannot be less than 0 or else image won't load
        if zoom_rec[:,2] < 0:
            zoom_rec[:,2] += -1*zoom_rec[:,2]
            zoom_rec[:,0] +=
            zoom_rec[:,3] += delta_y2

    #For both below, if a coordinate is larger than 1024, the difference
    # of that larger coord and 1024 will be subracted by the smaller coord.
        elif zoom_rec[:,3] > 1024:
            zoom_rec[:,2] -= zoom_rec[:,3] - 1024
            zoom_rec[:,3] = 1024

        elif zoom_rec[:,2] > 1024:
            zoom_rec[:,3] -= zoom_rec[:,2] - 1024
            zoom_rec[:,2] = 1024

    elif dx != dy:
        delta_x1 =(dy - dx)//2
        delta_x2 = int(np.ceil(dy-dx/2))
        zoom_rec[:,0] =  zoom_rec[:,0] - delta_x1
        zoom_rec[:,1] =  zoom_rec[:,1] + delta_x2
        print('deltas x', delta_x1,delta_x2)
    else:
        None

    data_zoom = self.data[zoom_rec[:,2]:zoom_rec[:,3],
                          zoom_rec[:,0]:zoom_rec[:,1]]
```

Figure 3: Sample code of the built Zoom function that shrinks down the coordinates of the BEC data.

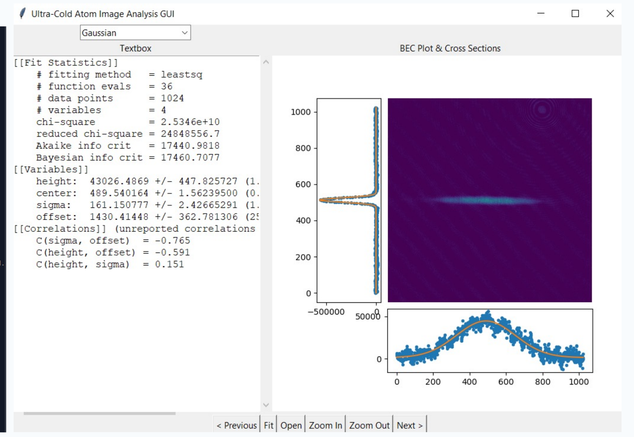

Figure 4: Image of Python based Graphical User Interface that includes data of a sample BEC (1024x1024), its cross sectional fits, and fitting parameters. Image of BEC curtesy of the Engels lab at WSU.

Features

The graphical user interface has a multitude of features available:

- **Selected file loading,** user can select file from local source to analyze.
- **Progressive file loading,** user can select first file in a series and automatically load the next file for analysis.
- **Zoom-in feature,** allowing user to manually select and isolate region of interest in data.
- **Automatic integrated cross-sections,** allowing user to assess and fit data as it is loaded and cropped.
- **Gaussian and Thomas-Fermi distribution fitting** with textbox output, allowing the user to measure characteristics of system by fitting integrated cross-sections of the data.

Next Steps: Production and Implementation

Currently, the GUI is able to upload images from data that was taken in past experiments, which can be used to analyze data well beyond the experimental date. Additional functionality includes:

- **Automatic data upload** of images as they are taken, which involves integrating the features of the GUI file upload with various camera, and thus, data sources.
- **Additional basic analysis features**, e.g. save function, pixel calibration setting, optical vs statistical Gaussian fitting
- **Additional complex analysis features**, e.g. image saturation correction, and Fourier analysis (eliminate unwanted background features from experimental vibrations).

Conclusion

Python-based object-oriented programming is an excellent and accessible tool for creating graphical-user interfaces for use in research settings. By using pre-made and freely available python packages, one can create and customize an interface to fit the needs of a research group for camera-software interfacing and data analysis.

Citations

1. Lewandowski, H, "Coherences and correlations in an ultracold Bose gas," **PhD Thesis**, University of Colorado, Boulder, 46-57 (2002).
2. Moravchik, D, "Imaging Methods of Cold Atoms," **Masters Thesis**, Ben-Gurion University of the Negev, 3-5 (2009).

Acknowledgements

This work is supported by the Henry Luce Foundation in connection to the Clare Boothe Luce Professorship Program and by the NSF (Grant No. PHY 2137848). We additionally acknowledge support from the Engels lab at Washington State University, who have provided example images for this project.