Summer Project Report

Azimuthal Analysis of Bose-Einstein Condensates Under Light-Induced Synthetic Gauge Fields

Submitted by:
Madhurima Sarkar
Roll No: P0221617
Department of Physical Sciences
UM-DAE Centre for Excellence in Basic Sciences

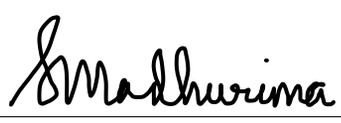
Supervisor:
Prof. Yu-Ju Lin
Institute of Atomic and Molecular Sciences
Academia Sinica

Declaration

I hereby declare that the work presented in this report is the result of my own original research, carried out under the guidance of my supervisor. To the best of my knowledge, it has not been submitted, either in whole or in part, for the award of any degree, diploma, or other qualification.

I further declare that all sources of information have been duly acknowledged and that this work has been conducted in compliance with ethical and academic standards.

The undersigned principal investigator (PI) hereby verifies that the work reported herein was carried out under their supervision and that, to the best of their knowledge, it is the student's own work.

Signature of Student	**Signature of Supervisor / PI**
	
Name: Madhurima Sarkar	Name: Prof. Yu-Ju Lin
Date: 2025/8/20	Date: 2025/8/20

Abstract

This project involved the azimuthal analysis of *in-situ* images of ^{87}Rb Bose–Einstein condensates (BECs) under light-induced synthetic gauge fields. A **Hall torus** is realized when a ring-shaped condensate's real spatial dimension is combined with a synthetic dimension created by coupling internal atomic states via Raman and microwave fields with periodic boundary conditions.

We used two co-propagating Raman beams-a Laguerre-Gaussian (LG) carrying orbital angular momentum and a Gaussian-to couple the atoms' spin to their center-of-mass orbital angular momentum (SOAMC). I developed Python-based tools for automated image analysis, including coordinate transformations, azimuthal averaging, and radial profiling, to study ring breaking, torus formation, and quench dynamics.

We observed characteristic two-minima density modulations unique to the Hall torus, controlled via the relative Raman phase, enabling a topological charge pump. Dynamically switching boundary conditions revealed nonequilibrium modulation emergence as the system acquired toroidal topology.

Acknowledgements

I would like to thank Prof.Yu-Ju Lin for the opportunity to work in her lab. I thank her for her patience and reassurance which were showered on me in the project's entirety.Her patience, valuable feedback, and constant availability throughout the project were invaluable to me. I aspire to have her great work ethic someday. I would also like to thank Prof. Ming-Shien Chang for teaching me the basics of laser locking and ECDL. The lab 107 group meetings were always thought-provoking and highly informative. I am grateful to Henry Su for providing the foundational code framework, which greatly facilitated my analysis work and for promptly helping me out when I was stuck with the simulation analysis . I would like to thank all members of Lab 107 as well- Justin, Derek, Cody, Frank, Ci-Ci for all the fun times and camaraderie. I had a lot of fun with them and their attempts to teach me both experimentation (and Mandarin) .Lastly, I would like to thank my mother for her constant support and belief in my abilities.

Contents

1 Introduction and Formalism

Bose–Einstein condensates (BECs) serve as an ideal platform to simulate complex quantum many-body phenomena owing to their high tunability, long coherence times, and ultracold temperatures. Formed when a dilute gas of bosonic atoms is cooled below a critical temperature T_c, a BEC consists of a macroscopic population of atoms occupying the lowest quantum state. The condensate is described by a mean-field macroscopic wavefunction $\psi(\mathbf{r}, t)$ whose evolution is governed by the Gross–Pitaevskii equation (GPE).

Traditionally, BECs are composed of neutral atoms, which do not interact with electromagnetic gauge fields in the same way that charged particles do. However, through carefully engineered atom–light interactions, it is possible to construct *synthetic gauge fields* that replicate the dynamics of charged particles under magnetic fields. These synthetic fields enable the exploration of phenomena such as the quantum Hall effect, topological phases, and spin–orbit coupling in a highly controlled and clean environment.

1.1 Gauge Fields

In quantum mechanics, the effect of an electromagnetic field on a charged particle is incorporated , where the canonical momentum $\hat{\mathbf{p}}$ is replaced by the kinetic momentum:

$$\hat{\mathbf{p}} \to \hat{\mathbf{p}} - q\mathbf{A}(\mathbf{r}), \tag{1}$$

where q is the charge and $\mathbf{A}(\mathbf{r})$ is the magnetic vector potential. The Hamiltonian of a particle in an external electromagnetic field becomes:

$$\hat{H} = \frac{1}{2m}\left(\hat{\mathbf{p}} - q\mathbf{A}(\mathbf{r})\right)^2 + V(\mathbf{r}), \tag{2}$$

and the corresponding Schrödinger equation encodes the magnetic effects entirely through the vector potential $\mathbf{A}(\mathbf{r})$, with the physical magnetic field given by:

$$\mathbf{B} = \nabla \times \mathbf{A}. \tag{3}$$

Neutral atoms do not experience Lorentz Force in an electromagnetic field. However, by engineering laser-induced couplings between internal spin states, one can generate an effective momentum shift analogous to $\mathbf{A}(\mathbf{r})$. Suitable laser combinations can make neutral atoms act like electrons in an electromagnetic field.

1.2 Light-Induced Synthetic Gauge Fields via Raman Coupling

To construct a synthetic gauge field for neutral atoms, we exploit Raman transitions induced by two laser beams that couple different internal (hyperfine) spin states. Consider

an atom with two relevant internal spin states, $|\uparrow\rangle$ and $|\downarrow\rangle$, coupled via a two-photon Raman process involving two laser beams: one Gaussian and the other a Laguerre–Gaussian (LG) beam carrying orbital angular momentum (OAM) $\ell\hbar$.

In a stimulated Raman transition, an atom absorbs a photon from one laser beam and emits a photon into a second laser beam. The total momentum imparted to the atom is the difference between the absorbed and emitted photon momenta.

Let the wavevectors of the two beams be $\mathbf{k}_1$ and $\mathbf{k}_2$, respectively. Then, the momentum imparted to the atom is:

$$\Delta\mathbf{p} = \hbar\mathbf{k}_1 - \hbar\mathbf{k}_2 = \hbar(\mathbf{k}_1 - \mathbf{k}_2). \tag{4}$$

In our experimental set-up, we use two co-propagating laser beams along e_z-one Gaussian and the other Laguerre Gaussian with an orbital angular momentum $l = \hbar$.The optical Raman coupling scheme enables coherent transitions between internal spin states while imparting photon momentum, resulting in spin–linear-momentum coupling—a form of spin–orbit coupling (SOC) between atomic spin and center-of-mass motion. A distinct variant, known as spin–orbital-angular-momentum coupling (SOAMC), couples atomic spin to orbital angular momentum using a Laguerre–Gaussian Raman beam carrying optical OAM.

In SOAMC systems, atoms interacting with Laguerre–Gaussian (LG) Raman beams acquire an azimuthal gauge potential of the form $\vec{A} = A(r)\,\hat{e}_\phi$. This results in an angular dispersion relation given by $(l - l_{\min})^2/2mr^2$, where l is the angular momentum quantum number, and $l_{\min} = rA(r)$ corresponds to the location of the shifted energy minimum. The gauge potential $A(r)$ is tunable through the Raman coupling strength $\Omega(r)$ and the Raman detuning δ, allowing experimental control over the dispersion landscape. This effective azimuthal gauge field is formally equivalent to introducing a synthetic rotation in the stationary Hamiltonian.

For atoms experiencing a sufficiently strong effective field $\vec{\Omega}_{\text{eff}} \cdot \vec{F}$, the kinetic energy term $-\frac{\hbar^2}{2m}\nabla^2$ becomes negligible. In this limit, the eigenstates of the total Hamiltonian are well approximated by the local dressed spin eigenstates $|\xi_n\rangle$, which are the eigenstates of $\vec{\Omega}_{\text{eff}} \cdot \vec{F}$. Under this approximation, the spinor wavefunction adiabatically follows the local eigenstate $|\xi_n(\vec{r})\rangle$, with the effective field defined by

$$\vec{\Omega}_{\text{eff}}(\vec{r}) = \Omega(r)\cos\phi\,\hat{e}_x - \Omega(r)\sin\phi\,\hat{e}_y + \delta\,\hat{e}_z,$$

where $\Omega(r)$ is the spatially dependent Raman coupling strength and δ is the Raman detuning.

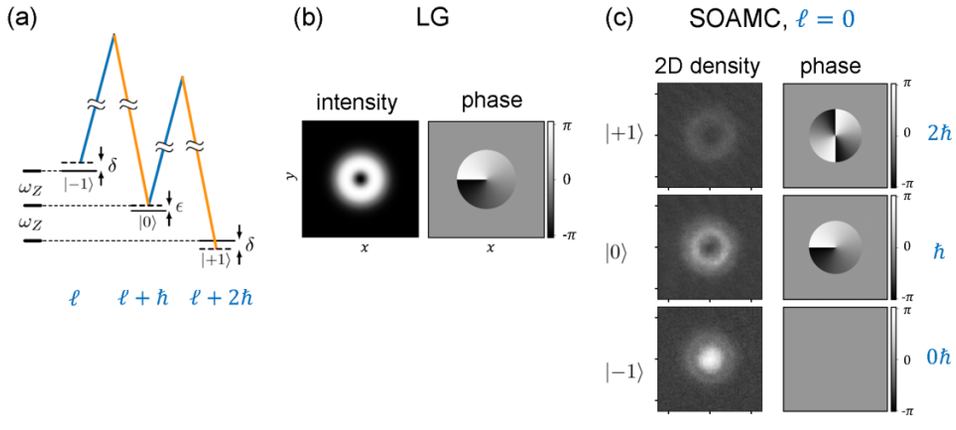


Figure 1: (a) Atomic energy levels: The three spin states with angular momentum $m_F = -1, 0, 1$ are split under an external magnetic field, with energy separations denoted as ω_2. The two Raman beams, LG (blue) and G (orange), have a frequency difference $\Delta\omega = \omega_L - \omega_2$. When the Raman detuning $\delta = \Delta\omega - \omega_2$ is sufficiently small, Raman coupling between the m_F and $m_F + 1$ spin states is allowed. (b) Intensity and phase spatial distributions of the Laguerre-Gaussian (LG) beam used in the Raman process. (c) Experimentally measured 2D atomic density distributions in the (x, y) plane for the three spin states. The relative phase distributions are shown, corresponding to atoms carrying orbital angular momentum of 0, $\hbar$, and $2\hbar$. The center point $r = 0$ is a phase singularity where the atomic density vanishes. The Raman detuning is $\delta/2\pi = 250\,\text{Hz}$. .(Source: [10])

The total wavefunction of a dressed atom can thus be written as

$$\langle \vec{r}|\Psi\rangle = \varphi_n(\vec{r})|\xi_n(\vec{r})\rangle,$$

where $\varphi_n(\vec{r})$ describes the external spatial part of the wavefunction, and $|\xi_n(\vec{r})\rangle$ is the normalized spinor component. The amplitude of the spatial wavefunction satisfies $|\varphi_n(\vec{r})| = \sqrt{n_a(\vec{r})}$, where n_a is the atomic density.

Projecting the full Hamiltonian onto the dressed state $|\xi_n\rangle$, we obtain the effective Hamiltonian governing the dynamics of φ_n:

$$H_{\text{eff}}^{(n)} = -\frac{\hbar^2}{2m}\nabla_{(r,z)}^2 + \frac{(L_z - rA_n)^2}{2mr^2} + V(r) + \varepsilon_n + W_n. \tag{5}$$

Here, $L_z = -i\hbar\partial_\varphi$ is the angular momentum operator with eigenvalue l, $A_n(r) = \frac{i\hbar}{r}\langle\xi_n|\partial_\varphi\xi_n\rangle$ is the azimuthal gauge potential, $V(r)$ is the spin-independent trapping potential, $\varepsilon_n = n\sqrt{\Omega(r)^2 + \delta^2}$ is the eigenenergy of the dressed state, and $W_n \approx \hbar^2/(2mr^2)$ is the geometric scalar potential.

We label the dressed states in increasing energy order as $|\xi_{-1}\rangle$, $|\xi_0\rangle$, and $|\xi_1\rangle$. The lowest energy state $|\xi_{-1}\rangle$ is given by a spin rotation parametrized by the Euler angles:

$$|\xi_{-1}(\vec{r})\rangle = e^{i(\theta+\gamma)} \begin{pmatrix} e^{i\phi}\dfrac{1-\cos\beta}{2} \\ -\dfrac{\sin\beta}{\sqrt{2}} \\ e^{-i\phi}\dfrac{1+\cos\beta}{2} \end{pmatrix}, \tag{6}$$

where $\beta(r) = \tan^{-1}\left(\frac{\Omega(r)}{\delta}\right)$ is the polar angle of the effective field $\vec{\Omega}_{\text{eff}}$, and $\theta + \gamma$ is a gauge phase.

By choosing the gauge such that $\theta + \gamma = \phi$, the corresponding azimuthal gauge potential becomes

$$A_{-1}(r) = \frac{\hbar}{r}\cos(\beta - 1). \tag{7}$$

In this gauge, the angular momentum of φ_{-1} is l, and the total mechanical angular momenta of the spin components $m_F = -1, 0, 1$ in the full state $\varphi_{-1}|\xi_{-1}\rangle$ are $l, l+\hbar, l+2\hbar$, respectively.

In this work, we analyze the of ^{87}Rb condensates subjected to such light-induced synthetic gauge fields. Our experimental system uses a pair of Raman beams—one Gaussian and the other Laguerre–Gaussian—to couple spin states and impart OAM to the atoms. We characterize the resulting dynamics through in-situ image analysis, using azimuthal analysis, and radial profiling to analyze ring breaking, formation of torus and get an understanding of quench dynamics.

2 Bose-Einstein Condensates

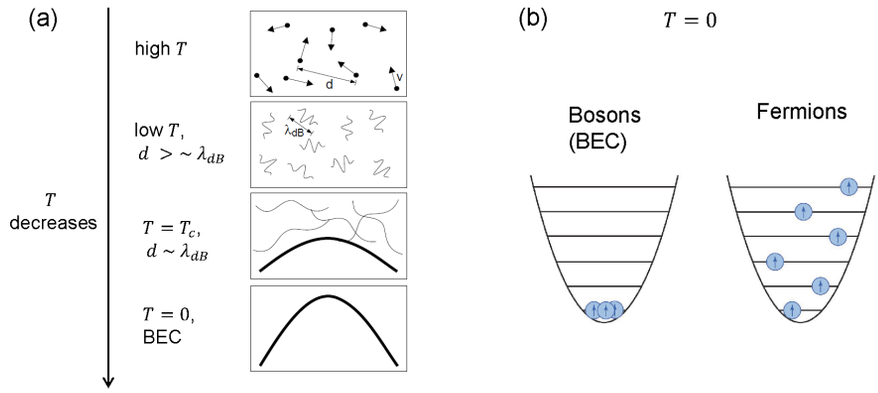


Figure 2: a). figure showing formation of BEC below T_c. b). distribution of atoms in various levels in bosons and fermions. (Source: [11])

Bose-Einstein Condensates (BECs) represent a unique phase of matter that arises when a dilute gas of bosonic atoms is cooled to ultralow temperatures such that a macroscopic number of atoms occupy the lowest quantum state. This phenomenon, predicted by Satyendra Nath Bose and Albert Einstein in the 1920s, was first realized experimentally in 1995 in dilute alkali gases like ^{87}Rb and ^{23}Na.

2.1 Quantum Statistical Foundation

Bosons are particles that obey Bose-Einstein statistics and have integer spin. The Bose-Einstein distribution for the mean occupation number of a single-particle state with energy ϵ at temperature T is given by:

$$\bar{n}(\epsilon) = \frac{1}{e^{(\epsilon-\mu)/k_B T} - 1}, \tag{8}$$

where μ is the chemical potential. As $T \to 0$, $\mu \to \epsilon_0$, the ground state energy. For bosons, unlike fermions, there is no restriction on the occupation number, and below a certain critical temperature, a macroscopic occupation of the ground state occurs:

$$N_0 \sim N\left(1 - \left(\frac{T}{T_c}\right)^3\right), \quad \text{for } T < T_c. \tag{9}$$

The critical temperature for an ideal Bose gas in a three-dimensional harmonic trap is:

$$T_c = \frac{\hbar\bar{\omega}}{k_B}\left(\frac{N}{\zeta(3)}\right)^{1/3}, \quad \bar{\omega} = (\omega_x\omega_y\omega_z)^{1/3}, \tag{10}$$

where $\zeta(3) \approx 1.202$ and ω_i are the harmonic trapping frequencies.

2.2 Macroscopic Wavefunction and Gross-Pitaevskii Equation

At $T \ll T_c$, nearly all atoms reside in the ground state and can be described by a single macroscopic wavefunction $\psi(\mathbf{r}, t)$ such that:

$$n(\mathbf{r}, t) = |\psi(\mathbf{r}, t)|^2, \tag{11}$$

and normalized as:

$$\int |\psi(\mathbf{r}, t)|^2 d^3\mathbf{r} = N. \tag{12}$$

The dynamics of this condensate are governed by the Gross-Pitaevskii equation (GPE), a nonlinear Schrödinger equation that incorporates mean-field interactions:

$$i\hbar\frac{\partial\psi(\mathbf{r}, t)}{\partial t} = \left[-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g|\psi(\mathbf{r}, t)|^2\right]\psi(\mathbf{r}, t), \tag{13}$$

where:

- m is the atomic mass,
- $V_{\text{ext}}(\mathbf{r})$ is the external trapping potential,
- $g = \dfrac{4\pi\hbar^2 a_s}{m}$ is the interaction strength,
- a_s is the s-wave scattering length.

2.3 Stationary Solutions and Thomas-Fermi Approximation

The time-independent form of the GPE is obtained by assuming $\psi(\mathbf{r}, t) = \psi(\mathbf{r})e^{-i\mu t/\hbar}$:

$$\mu\psi(\mathbf{r}) = \left[-\frac{\hbar^2}{2m}\nabla^2 + V_{\text{ext}}(\mathbf{r}) + g|\psi(\mathbf{r})|^2\right]\psi(\mathbf{r}), \tag{14}$$

where μ is the chemical potential.

In the regime where interaction energy dominates over kinetic energy (i.e., $g|\psi|^2 \gg \hbar^2\nabla^2/2m$), the kinetic term can be neglected — this is the Thomas-Fermi approximation:

$$n(\mathbf{r}) = \frac{1}{g}\left(\mu - V_{\text{ext}}(\mathbf{r})\right), \quad \text{for } V_{\text{ext}}(\mathbf{r}) \leq \mu, \tag{15}$$

and $n(\mathbf{r}) = 0$ otherwise.

For a harmonic trap $V_{\text{ext}}(\mathbf{r}) = \frac{1}{2}m(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2)$, the condensate takes the shape of an inverted paraboloid.

2.4 Healing Length and Sound Velocity

Two important length and energy scales in a BEC are:

Healing Length: This is the characteristic length scale over which the condensate recovers from local perturbations and is given by:

$$\xi = \frac{\hbar}{\sqrt{2mgn_0}}, \tag{16}$$

where n_0 is the peak condensate density. It also characterizes the vortex core size.

Speed of Sound: Small excitations in a BEC propagate like sound waves. In the Bogoliubov approximation, the speed of sound is:

$$c_s = \sqrt{\frac{gn_0}{m}}. \tag{17}$$

2.5 Phase Coherence and Superfluidity

BECs exhibit long-range phase coherence, which is evident in interference experiments. The phase of the macroscopic wavefunction $\psi(\mathbf{r}) = \sqrt{n(\mathbf{r})}e^{i\theta(\mathbf{r})}$ governs superfluid flow. The superfluid velocity is given by:

$$\mathbf{v}_s = \frac{\hbar}{m}\nabla\theta(\mathbf{r}). \tag{18}$$

Because the wavefunction must be single-valued, the circulation is quantized:

$$\oint \mathbf{v}_s \cdot d\mathbf{l} = \frac{h}{m}l, \quad l \in \mathbb{Z}, \tag{19}$$

leading to quantized vortices in rotating BECs or under synthetic magnetic fields.

3 Atomic Structure of ^{87}Rb

Rubidium-87 is an alkali metal atom with a single valence electron, which gives it a hydrogen-like structure. The total angular momentum of this electron, denoted by $\mathbf{J}$, arises from the combination of its orbital angular momentum $\mathbf{L}$ and its intrinsic spin $\mathbf{S}$, where $S = \frac{1}{2}$.

The fine structure of ^{87}Rb consists of a ground state $5^2S_{1/2}$ with $L = 0$ and $J = \frac{1}{2}$, and two excited states: $5^2P_{1/2}$ ($L = 1$, $J = \frac{1}{2}$) and $5^2P_{3/2}$ ($L = 1$, $J = \frac{3}{2}$). The optical transitions between the ground state and these excited states are known as the D_1 and D_2 lines, respectively.

In addition to fine structure, the atom exhibits hyperfine splitting due to the interaction between the total electronic angular momentum $\mathbf{J}$ and the nuclear spin $\mathbf{I}$, where $I = \frac{3}{2}$. The total angular momentum including the nucleus is given by

$$\mathbf{F} = \mathbf{J} + \mathbf{I}.$$

As a result, the ground state $5^2S_{1/2}$ and the excited state $5^2P_{1/2}$ (both with $J = \frac{1}{2}$) split into two hyperfine levels with $F = 1$ and $F = 2$. The $5^2P_{3/2}$ state, having $J = \frac{3}{2}$, splits into four levels: $F = 0$, 1, 2, and 3.

When the atom is placed in a magnetic field, the Zeeman effect causes each hyperfine level F to further split into $2F + 1$ magnetic sublevels. The interaction of the atom with an external magnetic field $\mathbf{B}$ is described by the Hamiltonian

$$H_B = \frac{\mu_B}{\hbar}(g_S\mathbf{S} + g_L\mathbf{L} + g_I\mathbf{I}) \cdot \mathbf{B},$$

where μ_B is the Bohr magneton, and g_S, g_L, and g_I are the g-factors for the spin, orbital, and nuclear contributions, respectively.

Assuming a weak magnetic field aligned along the z-axis, i.e., $\mathbf{B} = B_z\hat{e}_z$, the energy shift of a magnetic sublevel $|F, m_F\rangle$ is given by

$$\Delta E = \mu_B g_F m_F B_z,$$

where g_F is the Landé g-factor for the hyperfine level F. It can be approximated using

$$g_F \approx \left[1 + \frac{J(J+1) + S(S+1) - L(L+1)}{2J(J+1)}\right] \cdot \frac{F(F+1) - I(I+1) + J(J+1)}{2F(F+1)}.$$

For the $5^2S_{1/2}$ ground state of ^{87}Rb, the values of g_F are approximately $-\frac{1}{2}$ for $F = 1$ and $+\frac{1}{2}$ for $F = 2$.

4 Methods

To study the density distribution of the Bose-Einstein Condensates (BEC), we employed the absorption imaging technique. In this section, we describe the setup and methodology used for imaging the atomic cloud, as well as the steps involved in extracting the optical density (OD) from the images.

4.1 Absorption Imaging Method

The absorption imaging technique involves probing the atomic cloud with a resonant laser beam and recording the transmitted light with a CCD camera. The atoms in the BEC absorb part of the light, and this attenuation provides information about the atomic density. For an imaging beam with intensity I propagating along the z-direction , the intensity attenuation is described by

$$\frac{dI}{dz} = -n_{3D}(x, y, z)\sigma I$$

where $n_{3D}(x, y, z)$ is the 3D atomic density,$\sigma(I)$ is the scattering cross section with σ_0= $\frac{3\lambda^2}{2\pi}$ for imaging using cycling transition and I_{sat} is the saturation intensity. We assume $I << I_{sat}$ to get an intensity independent scattering cross section and the transmission through the dilute sample follows the Beer's Law.

The equation for optical density (OD) is given by:

$$\ln\left(\frac{I}{I_0}\right) = -OD = -\sigma n_{\text{col}}(x, y)$$

where:

- I_0 is the incident light intensity,
- I is the transmitted light intensity,
- σ is the absorption cross-section of the atoms for the probe transition. The scattering cross section σ is related by $\sigma = \hbar\omega\gamma_{\text{scatt}}/I$:

$$\gamma_{\text{scatt}} = \frac{\Gamma}{2} \cdot \frac{I/I_{\text{sat}}}{1 + 4(\delta/\Gamma)^2 + I/I_{\text{sat}}}, \quad \text{where } \frac{I}{I_{\text{sat}}} \equiv 2\left(\frac{\Omega}{\Gamma}\right)^2, \tag{20}$$

$$\sigma = \frac{\sigma_0}{1 + 4(\delta/\Gamma)^2 + I/I_{\text{sat}}}, \quad \text{with } \sigma_0 = \frac{\hbar\omega\Gamma}{2I_{\text{sat}}}. \tag{21}$$

Here, I is the incident light intensity, Ω is the Rabi frequency, I_{sat} is the saturation intensity, and σ_0 is the on-resonance scattering cross section and Γ is the longitu-

dinal decay rate due to spontaneous emission and γ_{scatt} is the photon scattering rate.

- $n_{\text{col}}(x, y)$ is the column density of the atomic cloud at position (x, y).

Using this relationship, we can calculate the OD and determine the atomic density profile of the BEC.

4.2 Image Acquisition Process

To obtain the OD, three images are recorded during the experiment:

- $I_{\text{atom}}(x, y)$: This image shows the intensity of the probe beam passing through the atomic cloud.
- $I_{\text{light}}(x, y)$: This image is taken when the probe beam passes through an empty region, without any atoms present.
- $I_{\text{bg}}(x, y)$: This background image captures the noise and ambient light, including any scattered light from the imaging system.

The OD of the atomic cloud is then calculated using the formula:

$$OD(x, y) = \sigma n_{\text{col}}(x, y) = -\ln\left(\frac{I_{\text{atom}}(x, y) - I_{\text{bg}}(x, y)}{I_{\text{light}}(x, y) - I_{\text{bg}}(x, y)}\right)$$

This formula allows us to compute the optical density (OD) at each point in the image, from which the atomic density distribution is derived.

4.3 In-Situ and Time-of-Flight (TOF) Imaging

We use two types of imaging techniques to study the BEC:

- In-situ Imaging: In this method, the probe beam directly illuminates the atoms while they are still trapped in the potential. The resulting image gives a snapshot of the density profile in the equilibrium state.
- Time-of-Flight (TOF) Imaging: In this technique, the trap is turned off, allowing the atoms to fall freely and expand. The atoms undergo ballistic expansion during the free-fall period, and the resulting cloud is larger, allowing for better resolution in the image.

TOF imaging provides a more detailed view of the BEC's spatial distribution due to the expansion of the cloud, which increases the resolution of the imaging system. My analysis primarily focuses on in-situ images, which capture the density profile of the condensate without expansion.In-situ imaging gives us spatial density profile of the distribution while TOF imaging gives us the momentum distribution.

4.4 Destructive Nature of Absorption Imaging

It is important to note that the absorption imaging process is destructive. During imaging, atoms absorb photons from the probe laser, which causes them to heat up and lose their condensate properties. Therefore, the atomic cloud cannot be re-imaged after the first exposure. The imaging process destroys the BEC, and the atoms need to be reloaded into the trap, followed by the cooling process, to prepare a new BEC for the next round of measurements.

5 Experimental Set-Up

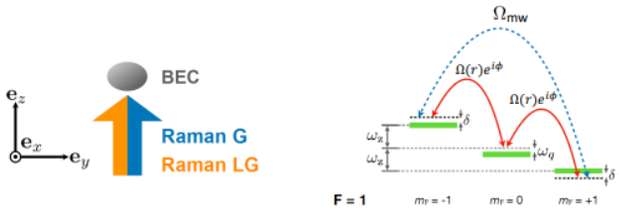


Figure 3: Experimental setup used in the measurement of toroidal density profiles.(Courtesy: Prof.Y-J Lin)

We implement Raman coupling using a pair of laser beams:a Gaussian beam and a Laguerre Gaussian beam carrying an orbital angular momentum of $\hbar$. The beams are at $\lambda = 790$ nm where the light shifts from the D1 and D2 lines cancel. These beams couple bare spin states $|m_F\rangle$ to $|m_{F\pm1}\rangle$.The Gaussian and Laguerre-Gaussian beams have frequency ω_L and $\omega_L + \Delta\omega_L$ and are polarised along e_y and e_x respectively. A vortex phase plate with a phase winding number of 1 and a radial index of 0 is used to produce the LG beam.
The spatial dependence of the Raman coupling in our system is described by

$$\Omega(r) = \Omega_M \sqrt{e} \left(\frac{r}{r_M}\right) e^{-r^2/2r_M^2}, \tag{22}$$

where r_M denotes the radial position at which the coupling strength is maximized. In our experiment, this maximum coupling corresponds to $\Omega_M/2\pi = 3.8$ kHz and occurs at $r_M = 17$ μm. The Raman detuning is defined as $\delta = \Delta\omega_L - \omega_Z$, where $\Delta\omega_L$ is the frequency offset between the Raman beams, and ω_Z is the effective linear Zeeman shift.

In addition to the Raman coupling, we apply a resonant two-photon microwave drive that couples the spin states $|m_F = 1\rangle$ and $|m_F = -1\rangle$ with an amplitude of $\Omega_{\rm mw}/2\pi = 3.15(0.25)$ kHz. When the system is at zero Raman detuning ($\delta = 0$), the full atom–light interaction Hamiltonian takes the form:

$$V = \frac{\Omega(r)}{\sqrt{2}} e^{i\phi}|1\rangle\langle 0| + \frac{\Omega(r)}{\sqrt{2}} e^{i\phi}|0\rangle\langle -1| + \frac{\Omega_{\rm mw}}{2} e^{i\theta_{\rm mw}}|-1\rangle\langle 1| + \text{H.c.} - \omega_q|0\rangle\langle 0|, \tag{23}$$

where ω_q represents the effective quadratic Zeeman shift. This includes light-induced shifts from the microwave drive. The phase of the microwave field, $\theta_{\rm mw}$, is stabilized relative to the phase difference between the Raman lasers, allowing control over the toroidal phase profile with high reproducibility.

Prior to turning on the microwave coupling, the atomic ensemble is initialized in

the lowest-energy Raman-dressed spin state $|\xi_{-1}\rangle$, which corresponds to the eigenstate of the Hamiltonian V in the absence of microwave driving ($\Omega_{\rm mw} = 0$).The dressed spin states in our system are denoted by $|\xi_n\rangle$, with $n = 0, \pm 1$ labeling the three branches of the Raman-dressed manifold. The Bose–Einstein condensate is initialized in the lowest-energy dressed state $|\xi_{-1}\rangle$, with the full condensate wavefunction expressed as $\varphi|\xi_{-1}\rangle$, where φ represents the external (spatial) part of the wavefunction.

For a finite detuning $\delta \neq 0$, the lowest-energy dressed spinor state has the form:

$$|\xi_{-1}(r)\rangle = \begin{pmatrix} e^{i2\phi}\dfrac{1-\cos\beta}{2} \\ -e^{i\phi}\dfrac{\sin\beta}{\sqrt{2}} \\ \dfrac{1+\cos\beta}{2} \end{pmatrix}, \tag{24}$$

where the mixing angle β is defined via $\tan\beta = \Omega(r)/\delta$. We adopt a gauge in which the initial condensate wavefunction φ carries no vorticity.

Our experiment begins with a thermal cloud of ^{87}Rb atoms prepared in the bare Zeeman state $|F = 1, m_F = -1\rangle$, confined within a crossed optical dipole trap supplemented by a repulsive (blue-detuned) optical sheet along the vertical direction. To initialize the system in the dressed state $|\xi_{-1}\rangle$, we adiabatically ramp up the Raman coupling over a duration of 10 ms, holding the detuning fixed at $\delta_i/2\pi = 2.8$ kHz. This is followed by a linear sweep of the detuning to zero over 40 ms. We then perform evaporative cooling by gradually lowering the dipole trap intensity, ultimately producing a condensate of $N = (8 \pm 1) \times 10^4$ atoms in $|\xi_{-1}\rangle$. The trapping frequency along the vertical ($\hat{z}$) direction is approximately 370 Hz.

Torus Formation

At zero Raman detuning ($\delta = 0$), the spatially varying Raman coupling $\Omega(r)$ gives rise to a dressed-state light shift that forms a ring-shaped potential minimum for atoms in the lowest-energy dressed state $|\xi_{-1}\rangle$. In this regime, the external potential experienced by the condensate is cylindrically symmetric, leading to a uniform distribution of atomic density around the ring. However in our experimental setup, we do not have a perfectly cylindrical symmetry as there are some fluctuations.

A two-photon microwave coupling $\Omega_{\rm mw}$ with a controlled phase $\theta_{\rm mw}$ leads to cyclic coupling between the dressed spin states $m_F = \pm 1$. To understand the atomic density distribution under cyclic coupling between the dressed spin states when the microwave coupling is turned on, we draw an analogy to a particle in a one dimensional periodic potential. In such systems, Bloch's theorem states that eigenstates of the Hamiltonian

can be expressed as:

$$\psi_{n,k}(x) = e^{ikx} u_{n,k}(x),$$

where $u_{n,k}(x)$ has the periodicity of the lattice. Due to the periodicity, momentum is only conserved *modulo* a reciprocal lattice vector $G = \frac{2\pi}{d}$. Consequently, states with momenta differing by integer multiples of G, i.e., $k \rightarrow k + nG$, can be coupled by a periodic potential. In presence of only Raman coupling, we have $|1, \ell + 2\rangle\,, \quad |0, \ell + 1\rangle\,, \quad |-1, \ell\rangle$, but when the microwave field is turned on, it leads to coupling between states $|1, \ell + 2\rangle$ and $|-1, \ell + 2\rangle$ and also coupling between states $|-1, \ell\rangle$ and $|1, \ell\rangle$. The microwave field can couple same spin states whose OAM differ by 2n,where n is an integer, i.e., for example, in the state $m_F = -1$, it couples between OAM :

$$\ell \rightarrow \ell \pm 2n.$$

This angular momentum coupling is analogous to the momentum coupling in a spatially periodic lattice. Thus, the eigenstates contain the ℓ and $\ell + 2n$ components and the fourier transform leads to a periodicity in the azimuthal direction which is directly analogous to the coupling between k and $k + n\frac{2\pi}{d}$ in a periodic lattice.

On diagonalizing V, we see that there are two minimas and two maximas with the energy minima corresponding to the OD maxima. The initially uniform ring breaks into a toroidal structure characterized by two distinct depletion regions along the azimuthal direction on turning on the microwave field.

Experimentally, the ring evolving into the torus on application of the microwave field is visible in *in situ* optical density (OD) images. Initially, the atomic OD is distributed around the ring. The distribution is not completely uniform as the external potential experienced by our condensate is not perfectly symmetric. Our t_{on} for the experiment is typically 50 ms with holdtime t_h of order less than 1 ms.

To quantify this ring breaking into the torus on applying the microwave field, we integrate the OD along the radial direction and examine the resulting angular density distribution as a function of azimuthal angle ϕ. In the symmetric ring configuration, this distribution is flat; in contrast, in the toroidal phase, the angular OD profile exhibits two distinct peaks located at $\phi = \mu_1, \mu_2$, separated by density minima. The full widths at half maximum (FWHM) of these peaks, denoted σ_1 and σ_2, provide a measure of the angular localization .

6 Image Analysis and Processing

The analysis of BEC images was carried out using custom Python code.In the appendix, I have included all the code for reference, as well as Henry's code as it serves as the foundational framework for many of the techniques used in this analysis.

To evaluate the quality of OD vs ϕ(azimuthal angle) fits, a Python routine was developed to compare two models: a double Gaussian with a fixed offset and one with a constrained free offset (0–3). Each model was fit to the OD data, and their performance was compared using residuals and the Akaike Information Criterion (AIC). The code identified the better-fitting model, flagged anomalous cases, and saved all fit parameters, residuals, and diagnostic plots for further analysis.

For the analysis of quench dynamics, it was noticed that using a double Gaussian with a free offset (constrained between 0 and 3) along with a suitable shift(45° in this case), we get a better 1D double gaussian fit with a lower residual and lower AIC compared to a model with fixed offset.

For the analysis of the simulation results, we have used fixed offset as the simulations are of the perfect conditions where the OD offset is zero. So for the simulation data, we have used fixed offset double gaussian fit.

The code in Appendix 9.1 was developed to process the raw images, generate optical density profile and fit a 1D double gaussian model after sorting the files according to their holdtime. In this analysis, the raw optical density (OD) profiles were first rotated to span the azimuthal angle range from 45° to 405°. This rotation helps align the data properly for the subsequent fitting process, ensuring a more accurate double Gaussian fit.

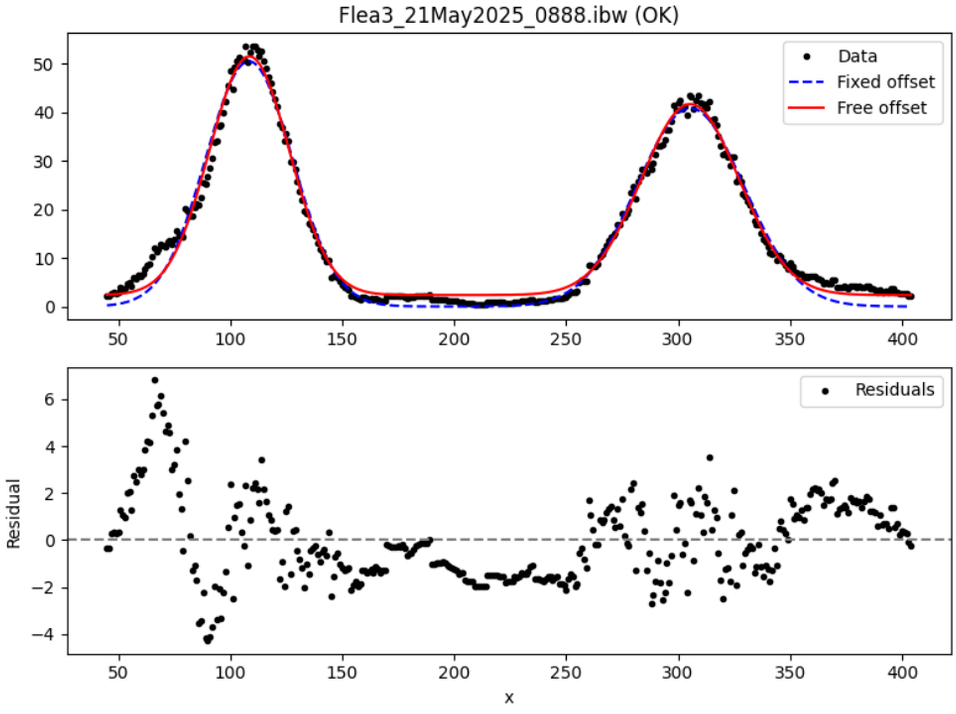


Figure 4: shows integrated OD vs ϕ and finds the best fit between free offset between 0 and 3 and fixed offset.It also plots the residual of what is determined to be the best fit.

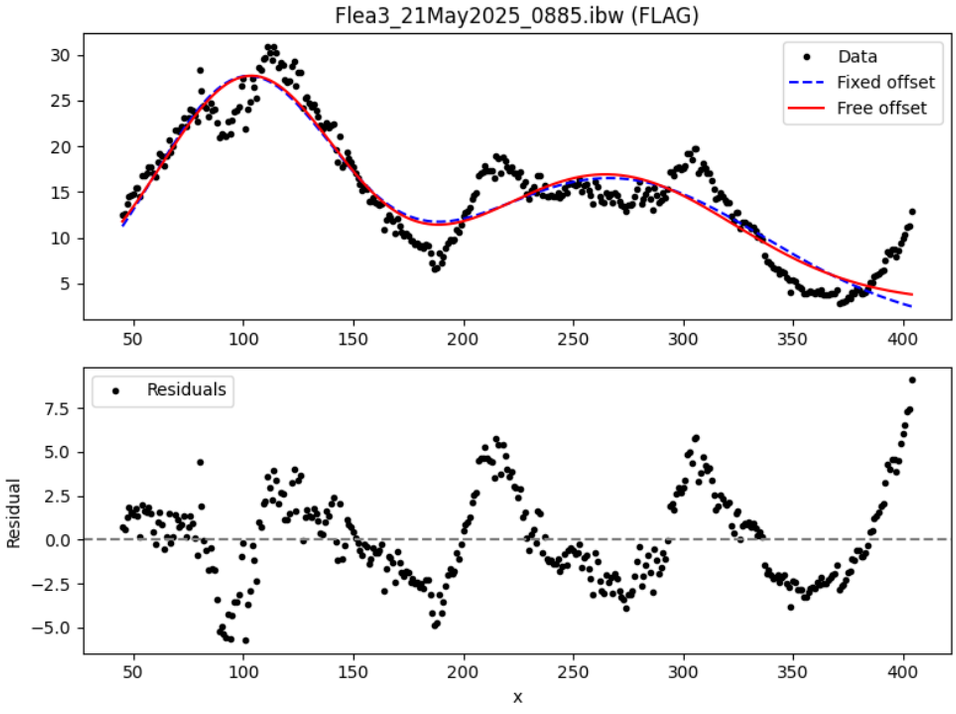


Figure 5: shows the code flagging plots that are considered to be bad fits based on the parameters specified.

For some of the flagged plots, we implemented skewed + normal Gaussian fit to get a better fit with a lower residual and lower AIC and for other plots with three peaks, triple gaussian fit was implemented. The code for the following have been provided in the appendix 9.2.

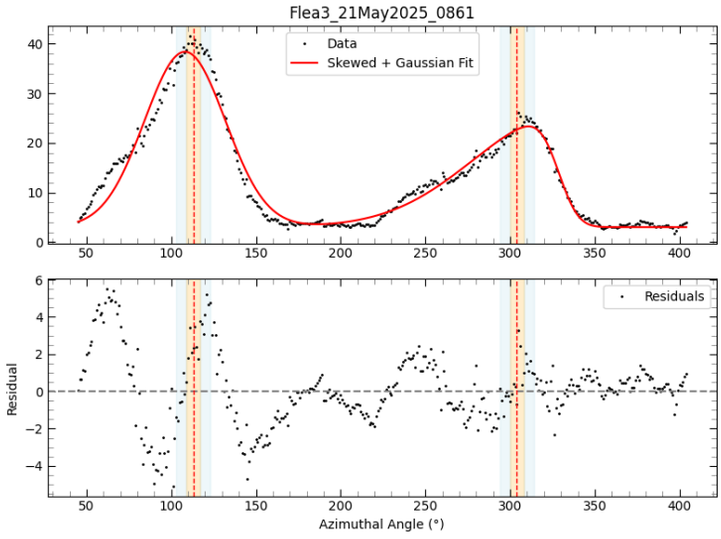


Figure 6: shows integrated OD vs ϕ with FWHM reference lines marked and fits the data to a normal Gaussian + a skewed Gaussian and also plots its residual.

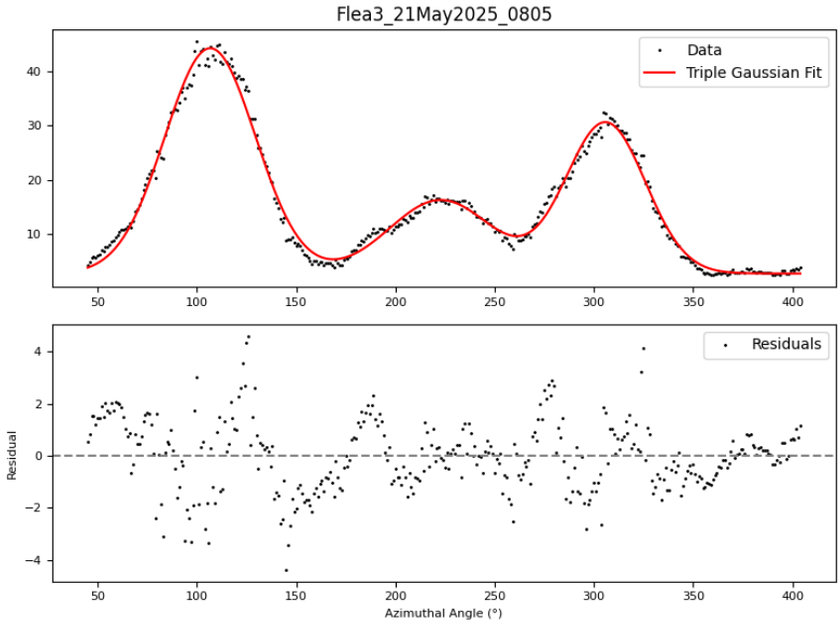


Figure 7: shows integrated OD vs ϕ and fits the data to a triple Gaussian and also plots its residual.

The image shown below is analysed with the center coordinates set at (534, 735). This particular image corresponds to a quench experiment, and the analysis was performed over a dataset consisting of 40 files ,861-900 taken during 21st May,2025. The processing involved extracting azimuthal density profiles and fitting the data to characterize the dynamics during the quench.

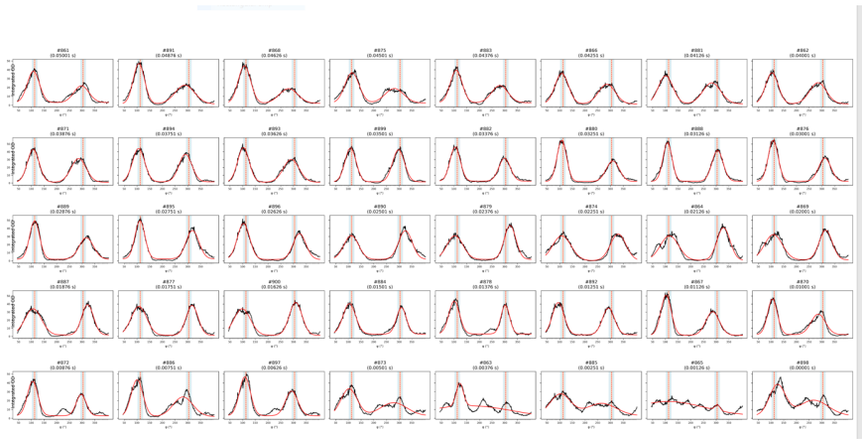


Figure 8: Quench experiment image analyzed using data taken on 21st May 2025, with center coordinates (534, 735). The analysis was done for 40 files and the files above are sorted according to holdtime with top left being the maximum(50 ms) and bottom right the least time (0.01 ms).

The FWHM was calculated and plotted along with reference lines for further analysis and radial FWHM lines were plotted in the OD image as well for further analysis.

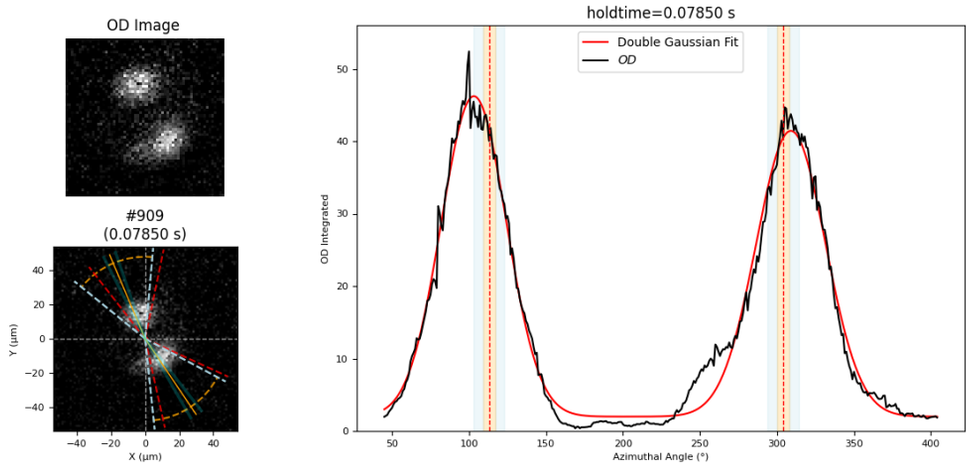


Figure 9: OD image with FWHM lines plotted radially along with reference lines. The double gaussian is fit to the OD data and the FWHM is marked there as well.

The code given in Appendix 9.3 was used to generate grayscale plots of the optical density (OD) images corresponding to image files numbered 861 to 900, taken on 21st May 2025. These images are arranged in a grid sorted by decreasing hold time, with the maximum hold time of 50 ms located at the top-left, and the minimum hold time of 0.01 ms at the bottom-right. Each subplot represents a processed OD image, overlaid with radial red lines indicating the full width at half maximum (FWHM) of the density distribution. This visualization provides a concise representation of the condensate evolution as a function of hold time, showing the ring breaking into a torus as the microwave field is turned on as a function of holdtime.

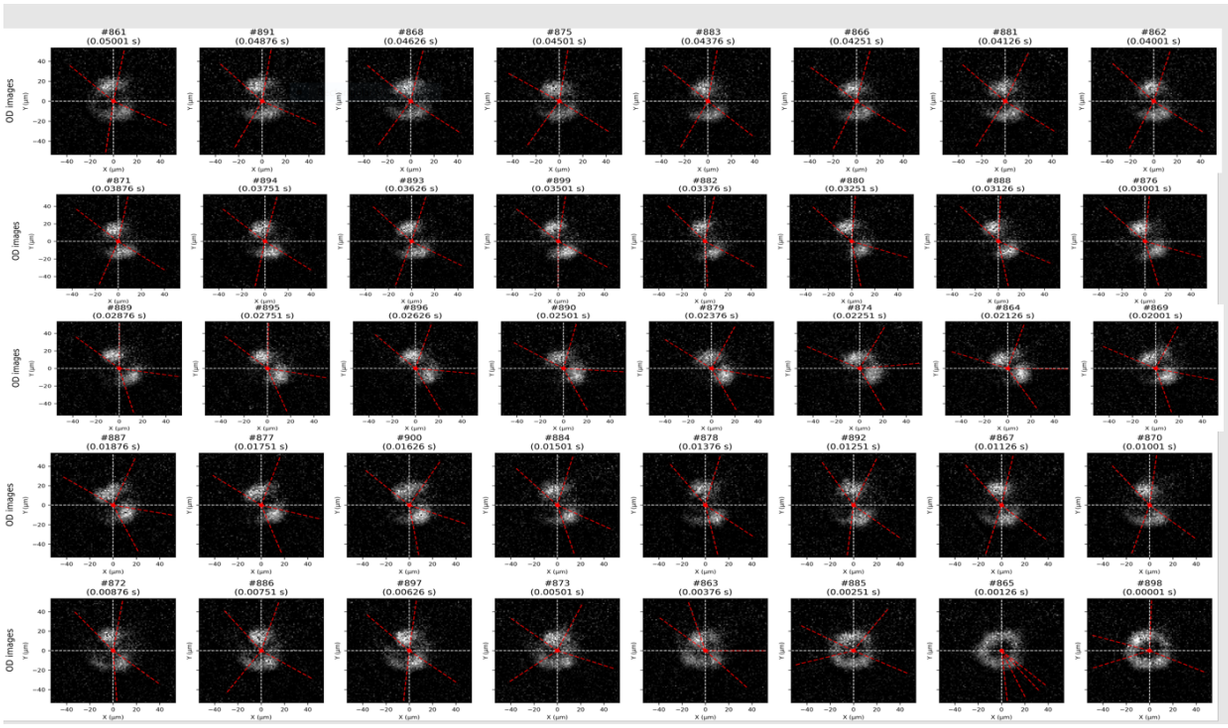


Figure 10: OD images with FWHM marked, using data taken on 21st May 2025, with center coordinates (534, 735). The analysis was done for 40 files and the files above are sorted according to holdtime with top left being the maximum (50 ms)and bottom right the least time(0.01 ms).

The code given in Appendix 9.4 plots OD vs r at the two peak positions, $\mu 1$ and $\mu 2$ and fit a 1d gaussian model to the data and plot the peak position of the 1d gaussian with the red dot. The code supports batch processing of image series and automatically saves fit results and parameters and diagnostic plots for further analysis. When it is plotted at only $\phi = \mu 1$ or $\mu 2$, the data is noisy, but it can be refined further and made a lot smoother if we use the OD values vs r at angles $\phi = \mu 1 \pm bin$ and $\phi = \mu 2 \pm bin$, and then fit the average to a double Gaussian.

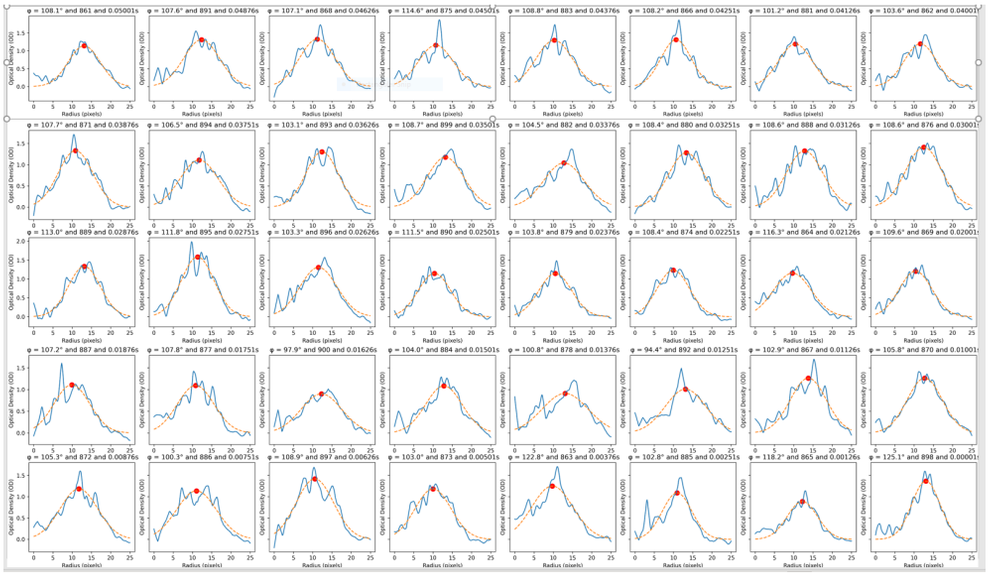


Figure 11: shows OD vs r at the first peak position $\mu 1 \pm bin$ for files 861-900 sorted by holdtime and fits a 1d gaussian to the averaged OD data, plotting the peak of the fitted gaussian with the red dot .

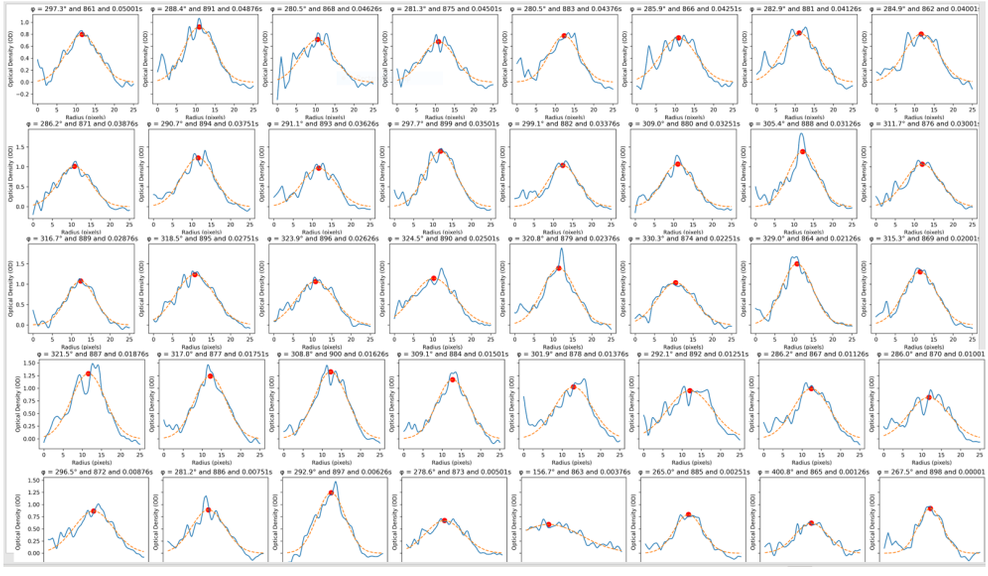


Figure 12: shows OD vs r at the second peak position $\mu 2 \pm bin$ for files 861-900 sorted by holdtime and fits a 1d gaussian to the averaged OD data ,plotting the peak of the fitted gaussian with the red dot.

7 Results and Discussion

Based on the analysis and numerical calculations, the following observations are made:

1. When the microwave field is turned on *sufficiently slowly*, the overall shape of the torus is found to be near independent of the holdtime (t_h) and the turn-on time (t_{on}). Here t_{on} is typically 50 ms and t_h is of the order of 0.1 ms. Following the application of the microwave field for $t_{\rm on} \gtrsim 35$ ms (where t_{on} is the turn-on time of the microwave field) ,the OD develops two angularly localized dips, indicating a redistribution of atomic density into a toroidal profile with azimuthal modulation. This torus-shaped density profile becomes stable for longer ramp times (e.g., $t_{\rm on} = 50$ ms) and thus our $t_{on} = 50$ ms.

2. When the microwave field is turned on *suddenly*, we observe quench dynamics in the torus. Sinusoidal oscillations of around freq= 50-60 Hz are observed with amplitude oscillations of around 20° in the azimuthal width of the torus as a function of the hold time for the numerical simulation data under ideal condition. For the experimental data, we see two azimuthal peaks and further analysis needs to be done.

3. On varying the microwave phase $\theta_{\rm mw}$ continuously from 0° to 180°, we see that the torus orientation changes in the counterclockwise direction. When $\theta_{\rm mw}$ is changed from 0° to 180°, the torus undergoes a rotation from 0° to 90°.

8 Conclusion

This project provided involved the azimuthal analysis of Bose–Einstein condensates (BECs) subjected to light-induced synthetic gauge fields. By developing Python-based tools for image processing and data analysis, I was able to automate the extraction and batch processing of the key features from both experimental and numerical simulation datasets. Using azimuthal analysis of the experimental images and the simulation data, we characterized the transformation of ring-shaped condensates into torus upon the application of microwave coupling, and analyzed the resulting density modulations using azimuthal and radial profiles. The comparative fitting of experimental data using various Gaussian models allowed for evaluation of quench dynamics and angular localization. Further analysis is being done to get a better understanding of the quench dynamics .

9 Appendix

9.1 Code for Double Gaussian Fit Comparison : With Free Offset constrained between 0 and 3 and Fixed Offset

```
#to compare fitting models-free offset with offset constrained between
      0 and 3 and fixed offset using aic and residual and to print all
      the parameters.
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


def gaussian(x, amp, mu, sigma):
    return amp * np.exp(-((x - mu)**2) / (2 * sigma**2))

def double_gaussian_fixed_offset(x, amp1, mu1, sigma1, amp2, mu2,
      sigma2):
    return gaussian(x, amp1, mu1, sigma1) + gaussian(x, amp2, mu2,
          sigma2)

def double_gaussian_free_offset(x, amp1, mu1, sigma1, amp2, mu2, sigma2
      , offset):
    return gaussian(x, amp1, mu1, sigma1) + gaussian(x, amp2, mu2,
          sigma2) + offset

def batch_fit(image_folder,image_prefix="Flea3_21May2025_",image_ext=".
      ibw", save_plots=True, output_summary="fit_summary_auto.csv"):
    results = []

    ibw_files = sorted([f for f in os.listdir(image_folder) if f.
          endswith(image_ext) and f.startswith(image_prefix)])

    for filename in ibw_files:
        filepath = os.path.join(image_folder, filename)
        print(f"Processing:{filename}")

        try:
            phi, OD_sum, _, _ = analyze_ibw_file(filepath, (534, 735),
                  (90, 90), 25)
            x = np.arange(45, 405, 1)
            y = np.concatenate([OD_sum[45:], OD_sum[:45]])

            guess = [50, 125, 8, 50, 300, 5]
```

```
            popt_fixed, _ = curve_fit(double_gaussian_fixed_offset, x,
                y, p0=guess)
            yfit_fixed = double_gaussian_fixed_offset(x, *popt_fixed)
            residual_fixed = np.sum((y - yfit_fixed)**2)
            aic_fixed = len(x) * np.log(residual_fixed / len(x)) + 2 *
                len(popt_fixed)
        except Exception as e:
            print(f"{filename}:Fixed-offset_fit_failed:_{e}")
            popt_fixed = [np.nan] * 6
            residual_fixed, aic_fixed = np.nan, np.nan
            yfit_fixed = np.zeros_like(y)

        try:
            guess_free = guess + [1]
            bounds = ([0, 0, 0, 0, 0, 0, 0], [np.inf, np.inf, np.inf,
                np.inf, np.inf, np.inf, 3])
            popt_free, _ = curve_fit(double_gaussian_free_offset, x, y,
                 p0=guess_free, bounds=bounds)
            yfit_free = double_gaussian_free_offset(x, *popt_free)
            residual_free = np.sum((y - yfit_free)**2)
            aic_free = len(x) * np.log(residual_free / len(x)) + 2 *
                len(popt_free)
            offset_fitted = popt_free[-1]
        except Exception as e:
            print(f"{filename}:Free-offset_fit_failed:{e}")
            popt_free = [np.nan] * 7
            residual_free, aic_free, offset_fitted = np.nan, np.nan, np
                .nan
            yfit_free = np.zeros_like(y)

        max_residual_free = np.max(np.abs(y - yfit_free)) if not np.
            isnan(popt_free).any() else np.nan
        rmse_free = np.sqrt(residual_free / len(x)) if not np.isnan(
            residual_free) else np.nan
        mask_range = (x >= 160) & (x <= 200)
        avg_residual_160_200 = np.mean(np.abs(y[mask_range] - yfit_free
            [mask_range])) if np.any(mask_range) else np.nan

        suspicious = False
        flag_reasons = []

        if aic_free > aic_fixed:
            suspicious = True
            flag_reasons.append("aic_free>aic_fixed")
        if avg_residual_160_200 > 1.95:
            suspicious = True
```

```
            flag_reasons.append("avg_residual_160_200>1.95")

        best_fit = "free" if aic_free < aic_fixed else "fixed"
        yfit_best = yfit_free if best_fit == "free" else yfit_fixed

        if save_plots:
            plt.figure(figsize=(6,3))
            plt.subplot(2, 1, 1)
            plt.plot(x, y, "k.",markersize=2, label="Data")
            plt.plot(x, yfit_fixed, "b--", label="Fixed offset")
            plt.plot(x, yfit_free, "r-", label="Free offset")
            plt.title(f"{os.path.basename(filename)} ({"FLAG" if
                  suspicious else "OK"})")
            plt.legend()

            plt.subplot(2, 1, 2)
            plt.plot(x, y - yfit_best, "k.", label="Residuals")
            plt.axhline(0, color="gray", linestyle="--")
            plt.xlabel("x")
            plt.ylabel("Residual")
            plt.legend()

            plt.tight_layout()
            plot_name = os.path.splitext(os.path.basename(filename))[0]
                   + "_fit45shifted1.png"
            plt.savefig(plot_name)
            plt.close()

        results.append({
            "file": filename,
            "best_fit": best_fit,
            "amp1_fixed": popt_fixed[0], "mu1_fixed": popt_fixed[1], "
                  sigma1_fixed": popt_fixed[2],
            "amp2_fixed": popt_fixed[3], "mu2_fixed": popt_fixed[4], "
                  sigma2_fixed": popt_fixed[5],
            "res_fixed": residual_fixed, "aic_fixed": aic_fixed,
            "amp1_free": popt_free[0], "mu1_free": popt_free[1], "
                  sigma1_free": popt_free[2],
            "amp2_free": popt_free[3], "mu2_free": popt_free[4], "
                  sigma2_free": popt_free[5],
            "offset_free": offset_fitted, "res_free": residual_free, "
                  aic_free": aic_free,
            "max_residual_free": max_residual_free, "rmse_free":
                  rmse_free,
            "avg_residual_160_200": avg_residual_160_200,
            "suspicious": suspicious, "flag_reasons": "; ".join(
                  flag_reasons)
```

```
        })

    df = pd.DataFrame(results)
    df.to_csv(output_summary, index=False)
    print(f"\nSaved summary to {output_summary}")
```

9.2 Plotting 1D double gaussian model with free offset b/w 0 and 3 to the OD vs ϕ data for files sorted by holdtime

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

def double_gaussian(x, A1, mu1, sigma1, A2, mu2, sigma2,offset):
   gauss1 = A1 * np.exp(-((x - mu1)**2) / (2 * sigma1**2))
   gauss2 = A2 * np.exp(-((x - mu2)**2) / (2 * sigma2**2))
   return gauss1 + gauss2+offset

def analyze_ibw_file(filename, circle_center, ROI, radius):
    phi, _, r, delta_r = std_r_vs_phi(filename=filename, circle_center=
          circle_center, ROI=ROI, radius=radius)
    _, OD_sum, _, _ = OD_vs_angle(filename=filename, circle_center=
          circle_center, ROI=ROI, radius=radius)
    return phi, OD_sum,r,delta_r

def manualfit(filename):
    phi, OD_sum, _, _ = analyze_ibw_file(filename, (534, 735), (90, 90)
          , 25)
    x= np.arange(45,405,1)
    y=np.concatenate([OD_sum[45:], OD_sum[:45]])
    initial_guess = [ 50,125,8,50, 300,5,1]


    bounds = ([0, 0, -np.inf, -np.inf, 0, 0, 0],[np.inf, np.inf, np.inf
          , np.inf, np.inf, np.inf, 3])


    params, covariance = curve_fit(double_gaussian, x, y, p0=
          initial_guess,bounds=bounds)
```

```
    errors = np.sqrt(np.diag(covariance))
    A1, mu1, sigma1, A2, mu2, sigma2,offset= params
    dA1, dmu1, dsigma1, dA2, dmu2, dsigma2,doffset= errors
    y_fit = double_gaussian(x, *params)
    return(A1,A2,mu1,mu2,sigma1,sigma2,offset,dA1,dA2,dmu1,dmu2,dsigma1
          ,dsigma2,doffset)

#to plot the fitted double gaussian fits and the original data in a
      line image sorted by holdtime or upulsetime

def plot_filtered_profiles(csv_path, data_folder, t1, t2,
      starting_file_number=791, image_prefix="Flea3_21May2025_",
      image_ext=".ibw", output_file="filtered_profiles.png"):
    df = pd.read_csv(csv_path)


    df["index"] = df["index"].astype(int)
    df["holdtime"] = df["holdtime"].astype(float)

    # Filter and sort
    filtered = df[(df["holdtime"] >= t1) & (df["holdtime"] <= t2)].
          sort_values(by="holdtime", ascending=False)
    n = len(filtered)
    ncols = n

    fig, axes = plt.subplots(1, ncols, figsize=(4*ncols, 3), sharey=
          True)
    if ncols == 1:
        axes = [axes]

    x = np.arange(45, 405, 1)
    highlight_regions = [(112.95 - 4, 112.95 + 4), (303.97 - 4, 303.97
          + 4)]
    highlight_regions2=[(112.95-10,112.95-4),(112.95+4,112.95+10)
          ,(303.97-10,303.97-4),(303.97+4,303.97+10)]

    ax_idx = 0


    for _, row in filtered.iterrows():
        file_index = row["index"].astype(int)
        holdtime = row["holdtime"]
        file_number = starting_file_number + file_index
        filename = f"{image_prefix}{int(file_number):04d}{image_ext}"
        filepath = os.path.join(data_folder, filename)
```

```
        try:
            _, OD_sum, _, _ = analyze_ibw_file(filename, (534, 735),
                (90, 90), 25)
            y = np.concatenate([OD_sum[45:], OD_sum[:45]])
            initial_guess = [ 50,125,8,50, 300,5,1]


            bounds = ([0, 0, -np.inf, -np.inf, 0, 0, 0],[np.inf, np.inf
                , np.inf, np.inf, np.inf, np.inf, 3])
            params, _ = curve_fit(double_gaussian, x, y, p0=
                initial_guess,bounds=bounds)
            ax = axes[ax_idx]
            ax.plot(x, y, 'k-')
            ax.plot(x, double_gaussian(x, *params), label="Double␣
                Gaussian␣Fit", color="red")
            ax.set_title(f"#{file_number}\n({holdtime:.5f}␣s)",
                fontsize=12) #for holdtime
            #ax.set_title(f"#{file_number}\n({holdtime:.5f} )",
                fontsize=12)  #for prog
            #ax.set_title(f"#{file_number}\n({holdtime:.5f} s)",
                fontsize=12)  #for upulsetime
            ax.axvline(303.97, color="red", linestyle="--", linewidth
                =1)
            ax.axvline(112.95, color='red', linestyle="--", linewidth
                =1)
            for start, end in highlight_regions:
                ax.axvspan(start, end, color="#FFCC99", alpha=0.4)
            for start, end in highlight_regions2:
                ax.axvspan(start, end, color="lightblue", alpha=0.4)
            ax.set_xticks([50, 100, 150, 200, 250, 300, 350])
            ax.set_xticks(np.arange(50, 401, 5), minor=True)
            ax.tick_params(axis="both", which="major", labelsize=7)
            ax.tick_params(axis="both", which="minor", length=2)
            ax.set_xlabel("phi(  )", fontsize=8)
            ax_idx += 1
        except Exception as e:
            ax.set_title(f"{file_number}", fontsize=10)
            print(f"Skipped{filename}:␣{e}")

    fig.supylabel("Integrated␣OD")
    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.savefig(output_file, dpi=300)
    plt.close()
    print(f"Saved{output_file}")
```

9.3 Code to plot OD images along with FWHM lines marked for files sorted by holdtime

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os

from scipy.optimize import curve_fit, root_scalar

def find_fwhm_bounds_gaussian(A, mu, sigma):
    half_max = A / 2

    def gauss(x):
        return A * np.exp(-((x - mu)**2) / (2 * sigma**2))

    def shifted(x):
        return gauss(x) - half_max

    left_bracket = [mu - 5 * sigma, mu]
    right_bracket = [mu, mu + 5 * sigma]

    try:
        left = root_scalar(shifted, bracket=left_bracket, method='
            brentq').root
        right = root_scalar(shifted, bracket=right_bracket, method='
            brentq').root
        return left, right
    except ValueError:
        print(f"Warning: Could not find FWHM bounds for peak at mu = {
            mu}")
        return None, None

def plot_filtered_ODprofiles(csv_path, data_folder, t1, t2,
      starting_file_number, image_prefix="Flea3_21May2025_", image_ext=
      ".ibw", output_file="filtered_profiles.png"):
    df = pd.read_csv(csv_path)


    df["index"] = df["index"].astype(int)
    df["holdtime"] = df["holdtime"].astype(float)

    # Filter and sort
    filtered = df[(df["holdtime"] >= t1) & (df["holdtime"] <= t2)].
          sort_values(by="holdtime", ascending=False)
    n = len(filtered)
```

```
    ncols = n

    fig, axes = plt.subplots(1, ncols, figsize=(2.7 * ncols, 3), sharey
          =True, gridspec_kw={"wspace": 0.02}, constrained_layout=True)
    if ncols == 1:
        axes = [axes]

    x = np.arange(45, 405, 1)
    highlight_regions = [(112.95 - 4, 112.95 + 4), (303.97 - 4, 303.97
          + 4)]
    highlight_regions2=[(112.95-10,112.95-4),(112.95+4,112.95+10)
          ,(303.97-10,303.97-4),(303.97+4,303.97+10)]

    ax_idx = 0


    for _, row in filtered.iterrows():
        file_index = row["index"].astype(int)
        holdtime = row["holdtime"]
        file_number = starting_file_number + file_index
        filename = f"{image_prefix}{int(file_number):04d}{image_ext}"
        #filepath = os.path.join(data_folder, filename)
        print(filename)
        #print(filepath)


        try:
            _, OD_sum, _, _ = analyze_ibw_file(filename, (534, 735),
                  (90, 90), 25)
            y = np.concatenate([OD_sum[45:], OD_sum[:45]])
            _,_,mu1,mu2,sigma1,sigma2,_,_,_,_,_,_,_,_,_=manualfit(
                  filename)



            start1, end1= find_fwhm_bounds_gaussian(A1,mu1, sigma1)
            start2, end2 = find_fwhm_bounds_gaussian(A2,mu2, sigma2)
            circle_center=(534,735)
            ROI=(90,90)
            radius=25
            extent_range=54 #um



            ax = axes[ax_idx]
            ax.imshow(OD_calculation(filename, center=circle_center,
                  ROI=ROI), cmap="Greys_r", origin="lower",extent=[-
```

```
                extent_range, extent_range, -extent_range,
                extent_range],vmax=2, vmin=0)
            ax.set_aspect("equal")
            #ax.set_title(f"#{file_number}\n({holdtime:.5f} s)",
                fontsize=12) #for holdtime and upulsetime
            ax.set_title(f"#{file_number}\n({holdtime:.5f}␣)", fontsize
                =12)  #for prog
            angles_deg = [start1,start2,end1,end2]
            for phi_deg in angles_deg:
              theta = np.deg2rad(phi_deg)
              x_end = extent_range * np.cos(theta)
              y_end = extent_range * np.sin(theta)
              ax.plot([0, x_end], [0, y_end], linestyle="--", color="
                  red", linewidth=1.5, alpha=0.8)

              ax.axhline(0, color="white", linestyle="--", linewidth=1,
                   alpha=0.6)
              ax.axvline(0, color="white", linestyle="--", linewidth=1,
                   alpha=0.6)
            _, OD_sum,_,_=analyze_ibw_file(filename, circle_center
                =(534, 735), ROI=(90, 90), radius=25)
            y=np.concatenate([OD_sum[45:],OD_sum[:45]])
            x=np.arange(45,405,1)

            ax.plot(0, 0, "ro", markersize=5)
            ax.set_xlabel("X(micron)")
            ax.set_ylabel("Y(micron)")


            ax_idx += 1
        except Exception as e:
            ax.set_title(f"{file_number}", fontsize=10)
            print(f"Skipped{filename}:{e}")

    fig.supylabel("OD␣images")
    plt.savefig(output_file, dpi=300)
    plt.close()
    print(f"Saved_line_image{output_file}")
```

9.4 Code to plot OD vs r at the two peak positions,$\mu 1$ and $\mu 2$ and to fit a 1D gaussian to it

```
#to plot OD vs r for peak positions using 1d double gaussian fit.
from scipy.optimize import curve_fit
from scipy.ndimage import map_coordinates
def gaussian_oned(r, A, r0, sigma):
    return A * np.exp(-(r - r0)**2 / (2 * sigma**2))

def OD_vs_radius_single_angle(filename_or_OD, center, ROI, phi_center,
      max_radius=25, num_r=200, output_dir=None):
    if isinstance(filename_or_OD, str):
        OD = OD_calculation(filename_or_OD, shape=(1280, 1024), ROI=ROI
             , center=center)
    else:
        OD = filename_or_OD

    cy, cx = ROI[1] // 2, ROI[0] // 2
    r_vals = np.linspace(0, max_radius, num_r)


    phi_rad = np.deg2rad(phi_center)


    x_coords = cx + r_vals * np.cos(phi_rad)
    y_coords = cy + r_vals * np.sin(phi_rad)
    coords = np.vstack([y_coords, x_coords])
    od_line = map_coordinates(OD, coords, order=1, mode="nearest")


    initial_guess = [np.max(od_line), r_vals[np.argmax(od_line)], 5]  #
          Amplitude, peak position, initial sigma
    popt, pcov = curve_fit(gaussian_oned, r_vals, od_line, p0=
         initial_guess)
    perr = np.sqrt(np.diag(pcov))

    A, r0, sigma = popt


    if output_dir:
        os.makedirs(output_dir, exist_ok=True)

        plt.figure(figsize=(6, 4))
        plt.plot(r_vals, od_line, label=f"OD_Profile_for{filename}")
        plt.plot(r_vals, gaussian_oned(r_vals, *popt), label="Gaussian␣
             Fit", linestyle="--")
```

```
        plt.scatter(r0, gaussian_oned(r0, *popt), color="red" ,label=f"
            Peak at r = {r0:.2f}")
        plt.title(f"OD Profile and Gaussian Fit at phi={phi_center:.1f}
            ")
        plt.xlabel("Radius (pixels)")
        plt.ylabel("Optical Density (OD)")
        plt.legend()
        plt.grid(True)
        plt.tight_layout()

        base = os.path.basename(filename_or_OD).split(".")[0]
        plt.savefig(os.path.join(output_dir, f"{base}_fit_phi{
            phi_center:.1f}.png"))
        plt.close()

    # Return the peak position and fit parameters
    return {"r_max": r0, "r0_err": perr[1], "od_line": od_line, "popt":
        popt, "r_vals": r_vals}

def batch_fit_and_save(csv_path, data_folder,output_file,
    starting_file_number, t1, t2, ROI, center,shape=(1280, 1024),
    output2_csv="peakfits.csv", show_plot=False,image_prefix="
    Flea3_21May2025_", image_ext=".ibw", plot_output_dir="plots",
    plot_output_dir2="plots2"):

    df = pd.read_csv(csv_path)
    df["index"] = df["index"].astype(int)
    df["holdtime"] = df["holdtime"].astype(float)
    filtered = df[(df["holdtime"] >= t1) & (df["holdtime"] <= t2)].
        sort_values(by="holdtime",ascending=True)
    n = len(filtered)
    ncols = n

    fig, axes = plt.subplots(1, ncols, figsize=(3.1 * ncols, 3), sharey
        =True, gridspec_kw={"wspace": 0.04}, constrained_layout=True)
    if ncols == 1:
        axes = [axes]

    results1 = []
    results=[]
    ax_idx = 0
    for _, row in filtered.iterrows():
        file_index = int(row["index"])
        holdtime_val = row["holdtime"]
        file_number = starting_file_number + file_index
        filename = f"{image_prefix}{int(file_number):04d}{image_ext}"
        filepath = os.path.join(data_folder, filename)
```

```
        ax = axes[ax_idx]

        print(f"Processing␣{filename}␣(holdtime␣=␣{holdtime_val})")

        try:
            print(filename)
            fit = manualfit(filepath)
            fit2 = OD_vs_radius_single_angle(filepath, ROI=ROI, center=
                  center, phi_center=fit["mu1"], output_dir=
                  plot_output_dir2)
            fit3=OD_vs_radius_single_angle(filepath, ROI=ROI, center=
                  center, phi_center=fit["mu2"], output_dir=
                  plot_output_dir2)
            fit4=OD_vs_radius_single_angle(filepath, center, ROI,
                  phi_center=fit["mu1"], max_radius=25, num_r=200,
                  output_dir=None)
            phi_center = fit["mu1"]
            od_line=fit4["od_line"]
            r_vals=fit4["r_vals"]
            r0=fit4["r_max"]
            popt=fit4["popt"]
            OD = OD_calculation(filepath, shape=(1280, 1024), ROI=ROI,
                  center=center)


            ax.plot(r_vals, od_line, label=f"OD_Profile_for{filename}")
            ax.plot(r_vals, gaussian_oned(r_vals, *popt), label="
                  Gaussian␣Fit", linestyle="--")
            ax.scatter(r0, gaussian_oned(r0, *popt), color="red", label
                  =f"Peak␣at␣r␣=␣{r0:.2f}",s=75)
            ax.set_title(f"phi={phi_center:.1f} _and_ {file_number}
                  _and_{holdtime_val:.5f}s")
            ax.set_xlabel("Radius(pixels)")
            ax.set_ylabel("Optical␣Density(OD)")

            ax_idx+=1


            results1.append({"filename": filename, "holdtime":
                  holdtime_val, "r1_max": fit2["r_max"],"r2_max":fit3["
                  r_max"],"mu1":fit["mu1"],"mu2":fit["mu2"]})#for
                  holdtime
            #results1.append({"filename": filename, "upulsetime":
                  holdtime_val, "r1_max": fit2["r_max"],"r1_err": fit2
                  ["r0_err"],"r2_max": fit3["r_max"],"r2_err": fit3["
```

```
                r0_err"],"mu1":fit["mu1"],"mu2":fit["mu2"]})#for
                upulsetim e
            #results1.append({"filename": filename, "r1_max": fit2["
                r_max"],"r2_max":fit3["r_max"],"mu1":fit["mu1"],"mu2
                ":fit["mu2"]})

        except Exception as e:
            print(f"Failed to process {filename}:{e}")
    plt.savefig(output_file,dpi=300)
    plt.close()

    pd.DataFrame(results1).to_csv(output2_csv,index=False)
    print(f"\nBatch fit complete.")
```

9.5 Code for skewed Gaussian fit

```
#for skewed gaussian fit
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit, root_scalar
from scipy.stats import skewnorm
from scipy.optimize import minimize_scalar

def find_skewnorm_peak(mu, sigma, alpha):
    pdf = lambda x: -skewnorm.pdf(x, alpha, loc=mu, scale=sigma)
    result = minimize_scalar(pdf, bounds=(mu - 5*sigma, mu + 5*sigma),
          method="bounded")
    return result.x if result.success else np.nan



def skewed_plus_gaussian(x, A1, mu1, sigma1, alpha1, A2, mu2, sigma2,
      offset):
    skewed = A2 * skewnorm.pdf(x, alpha1, loc=mu2, scale=sigma2)
    gaussian = A1 * np.exp(-((x - mu1) ** 2) / (2 * sigma1 ** 2))
    return skewed + gaussian + offset

def skewnorm_mode(mu, sigma, alpha):
    delta = alpha / np.sqrt(1 + alpha**2)
    return mu + sigma * delta * np.sqrt(2 / np.pi)

def fwhm_skewed_gaussian(A, mu, sigma, alpha, resolution=2000):

    from scipy.stats import skewnorm
```

```
    from scipy.optimize import root_scalar

    def skewed_pdf(x):
        return A * skewnorm.pdf(x, alpha, loc=mu, scale=sigma)

    try:

        peak = find_skewnorm_peak(mu, sigma, alpha)
        if np.isnan(peak):
            raise ValueError("Peak finding failed.")


        half_max = skewed_pdf(peak) / 2


        def half_max_func(x):
            return skewed_pdf(x) - half_max


        left = root_scalar(half_max_func, bracket=[peak - 5*sigma, peak
              ], method="brentq").root
        right = root_scalar(half_max_func, bracket=[peak, peak + 5*
              sigma], method="brentq").root
        fwhm = right - left
        return left, right, fwhm

    except Exception as e:
        print(f"[FWHM] Failed to compute FWHM: {e}")
        return None, None, None




def fit_ibw_file(filepath):
    try:
        phi, OD_sum, _, _ = analyze_ibw_file(filepath, (534, 735), (90,
               90), 25)
        x = np.arange(45, 405, 1)
        y = np.concatenate([OD_sum[45:], OD_sum[:45]])

        A1, mu1, sigma1, alpha1 = 40, 110, 20, 0
        A2, mu2, sigma2 = 20, 300, 15
        offset = 1

        p0 = [A1, mu1, sigma1, alpha1, A2, mu2, sigma2, offset]
```

```
        bounds = ([0, 0, 1, -20, 0, 0, 1, 0], [np.inf, 400, 100, 20, np
            .inf, 400, 100, 3])

        popt, _ = curve_fit(skewed_plus_gaussian, x, y, p0=p0, bounds=
            bounds)
        y_fit = skewed_plus_gaussian(x, *popt)
        residuals = y - y_fit

        # FWHM calculations

        left, right, fwhm_skewed = fwhm_skewed_gaussian(popt[4], popt
            [5], popt[6], popt[3])
        fwhm_symmetric = 2.35482 * popt[2]

        return x, y, y_fit, residuals, popt, fwhm_skewed,
            fwhm_symmetric

    except Exception as e:
        print(f"Fit failed for {filepath}: {e}")
        return None, None, None, None, None, None, None

def save_fit_plot(x, y, y_fit, residuals, filename, output_dir):
    base = os.path.splitext(os.path.basename(filename))[0]
    plt.figure(figsize=(8, 6))
    plt.subplot(2, 1, 1)
    plt.plot(x, y, "k.", markersize=2, label="Data")
    plt.plot(x, y_fit, "r-", label="Skewed + Gaussian Fit")
    plt.axvline(304, color="red", linestyle="--", linewidth=1)
    plt.axvline(112.95, color="red", linestyle="--", linewidth=1)
    plt.axvspan(304 - 4, 304 + 4, color="orange", alpha=0.2)
    plt.axvspan(112.95 - 4, 112.95 + 4, color="orange", alpha=0.2)
    plt.axvspan(304 - 10, 304 - 4, color="lightblue", alpha=0.2)
    plt.axvspan(304 + 4, 304 + 10, color="lightblue", alpha=0.2)
    plt.axvspan(112.95 - 10, 112.95 - 4, color="lightblue", alpha=0.2)
    plt.axvspan(112.95 + 4, 112.95 + 10, color="lightblue", alpha=0.2)
    plt.title(base)
    plt.legend()
    plt.minorticks_on()
    plt.tick_params(axis="both", which="both", direction="in", top=True
        , right=True)
    plt.tick_params(axis="both", which="minor", length=4, color="gray")
    plt.tick_params(axis="both", which="major", length=6)

    plt.subplot(2, 1, 2)
    plt.plot(x, residuals, "k.", markersize=2, label="Residuals")
    plt.axhline(0, color="gray", linestyle="--")
    plt.xlabel("Azimuthal Angle")
```

```
    plt.ylabel("Residual")
    plt.axvline(304, color="red", linestyle="--", linewidth=1)
    plt.axvline(112.95, color="red", linestyle="--", linewidth=1)
    plt.axvspan(304 - 4, 304 + 4, color="orange", alpha=0.2)
    plt.axvspan(112.95 - 4, 112.95 + 4, color="orange", alpha=0.2)
    plt.axvspan(304 - 10, 304 - 4, color="lightblue", alpha=0.2)
    plt.axvspan(304 + 4, 304 + 10, color="lightblue", alpha=0.2)
    plt.axvspan(112.95 - 10, 112.95 - 4, color="lightblue", alpha=0.2)
    plt.axvspan(112.95 + 4, 112.95 + 10, color="lightblue", alpha=0.2)
    plt.legend()
    plt.minorticks_on()
    plt.tick_params(axis="both", which="both", direction="in", top=True
          , right=True)
    plt.tick_params(axis="both", which="minor", length=4, color="gray")
    plt.tick_params(axis="both", which="major", length=6)
    plt.tight_layout()

    outpath = os.path.join(output_dir, f"{base}_fit.png")
    plt.savefig(outpath)
    plt.close()
    print(f"Saved␣plot:␣{outpath}")



def batch_fit_ibw_skewed_plus_gaussian(input_dir="ibw_data", output_dir
      ="fit_output", summary_file="fit_summary.csv"):
    os.makedirs(output_dir, exist_ok=True)
    results = []

    for fname in os.listdir(input_dir):
        if not fname.endswith(".ibw"):
            continue

        filepath = os.path.join(input_dir, fname)
        x, y, y_fit, residuals, popt, fwhm_skewed, fwhm_symmetric =
              fit_ibw_file(filepath)

        if popt is not None:
            save_fit_plot(x, y, y_fit, residuals, filepath, output_dir)
            results.append({
                "file": fname,
                "A1": popt[0], "mu1": popt[1], "sigma1": popt[2], "
                      alpha1": popt[3],
                "A2": popt[4], "mu2": popt[5], "sigma2": popt[6],
                "peak2_mode": find_skewnorm_peak(popt[5], popt[6], popt
                      [3]),
                "offset": popt[7],
```

```
                "FWHM_skewed": fwhm_skewed,
                "FWHM_symmetric": fwhm_symmetric
            })

    df = pd.DataFrame(results)
    excel_path = os.path.join(output_dir, summary_file)
    df.to_csv(excel_path, index=False)
    print(f"Fit_summary_saved_to:_{excel_path}")
```

9.6 Code for Triple Gaussian Fit

```
#fitting it to a triple gaussian and print all the parameters in a csv
      file.
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit


def triple_gaussian(x, A1, mu1, sigma1, A2, mu2, sigma2, A3, mu3,
      sigma3, offset):
    g1 = A1 * np.exp(-((x - mu1)**2) / (2 * sigma1**2))
    g2 = A2 * np.exp(-((x - mu2)**2) / (2 * sigma2**2))
    g3 = A3 * np.exp(-((x - mu3)**2) / (2 * sigma3**2))
    return g1 + g2 + g3 + offset


def fit_triple_gaussian_ibw(filepath):
    try:
        phi, OD_sum, _, _ = analyze_ibw_file(filepath, (534, 735), (90,
               90), 25)
        x = np.arange(45, 405, 1)
        y = np.concatenate([OD_sum[45:], OD_sum[:45]])


        A1, A3, A2 =40,10,25     #guess for 2nd 2 peaks
        mu1, mu3, mu2 = 120, 225, 310     #guess for 2nd 2 peaks
        #A3, A1, A2 =20,30,40 #guess for 1st two peaks case
       # mu3, mu1, mu2 = 75, 125, 325 #guess for 1st two peaks case
        sigma1 = sigma2 = sigma3 = 20
        offset = 1

        p0 = [A1, mu1, sigma1, A2, mu2, sigma2, A3, mu3, sigma3, offset
              ]
        bounds = (
```

```
            [0, 0, 1, 0, 0, 1, 0, 0, 1, 0],
            [np.inf, 400, 100, np.inf, 400, 100, np.inf, 400, 100, 3]
        )

        popt, _ = curve_fit(triple_gaussian, x, y, p0=p0, bounds=bounds
            )
        y_fit = triple_gaussian(x, *popt)
        residuals = y - y_fit
        return x, y, y_fit, residuals, popt

    except Exception as e:
        print(f"Fit failed for {filepath}: {e}")
        return None, None, None, None, None


def save_fit_plot(x, y, y_fit, residuals, filename, output_dir):
    base = os.path.splitext(os.path.basename(filename))[0]
    plt.figure(figsize=(8, 6))
    plt.subplot(2, 1, 1)
    plt.plot(x, y, "k.", markersize=2, label="Data")
    plt.plot(x, y_fit, "r-", label="Triple Gaussian Fit")
    plt.title(base)
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.plot(x, residuals, "k.", markersize=2, label="Residuals")
    plt.axhline(0, color="gray", linestyle="--")
    plt.xlabel("Azimuthal Angle")
    plt.ylabel("Residual")
    plt.legend()
    plt.tight_layout()

    outpath = os.path.join(output_dir, f"{base}_triple_fit.png")
    plt.savefig(outpath)
    plt.close()
    print(f"Saved plot: {outpath}")


def batch_fit_ibw_triple_gaussian(input_dir, output_dir, summary_file):
    os.makedirs(output_dir, exist_ok=True)
    results = []

    for fname in os.listdir(input_dir):
        if not fname.endswith(".ibw"):
            continue

        filepath = os.path.join(input_dir, fname)
```

```
        x, y, y_fit, residuals, popt = fit_triple_gaussian_ibw(filepath
            )

        if popt is not None:
            save_fit_plot(x, y, y_fit, residuals, filepath, output_dir)
            results.append({
                "file": fname,
                "A1": popt[0], "mu1": popt[1], "sigma1": popt[2],"fwhm1
                    ":2*popt[2]*(np.sqrt(2*np.log(2))),
                "A2": popt[3], "mu2": popt[4], "sigma2": popt[5],"fwhm2
                    ":2*popt[5]*(np.sqrt(2*np.log(2))),
                "A3": popt[6], "mu3": popt[7], "sigma3": popt[8],"fwhm3
                    ":2*popt[8]*(np.sqrt(2*np.log(2))),
                "offset": popt[9]
            })

    df = pd.DataFrame(results)
    excel_path = os.path.join(output_dir, summary_file)
    df.to_csv(excel_path, index=False)
    print(f"Fit_summary_saved_to:{excel_path}")
```

References

[1] H.-R. Chen, K.-Y. Lin, P.-K. Chen, N.-C. Chiu, J.-B. Wang, C.-A. Chen, P. Huang, S.-K. Yip, Y. Kawaguchi, and Y.-J. Lin, Phys. Rev. Lett. **121**, 113204 (2018).

[2] C.-L. Hung, *In Situ Probing of Two-Dimensional Quantum Gases*, Ph.D. thesis, University of Chicago (2011).

[3] K. Jimenez-Garcia, *Artificial Gauge Fields for Ultracold Neutral Atoms*, Ph.D. thesis, NIST, University of Maryland, College Park (2012).

[4] V. Galitski, G. Juzeliūnas, and I. B. Spielman, Phys. Today **72**(1), 38 (2019).

[5] T. Ozawa and H. M. Price, Nat. Rev. Phys. **1**, 349 (2019).

[6] Y. Yan, S.-L. Zhang, S. Choudhury, and Q. Zhou, Phys. Rev. Lett. **123**, 260405 (2019).

[7] C.-H. Li, Y. Yan, S.-W. Feng, S. Choudhury, D. B. Blasing, Q. Zhou, and Y. P. Chen, PRX Quantum **3**, 010316 (2022).

[8] R. P. Anderson, D. Trypogeorgos, A. Valdés-Curiel, Q.-Y. Liang, J. Tao, M. Zhao, T. Andrijauskas, G. Juzeliūnas, and I. B. Spielman, Phys. Rev. Res. **2**, 013149 (2020).

[9] C. J. Foot, *Atomic Physics* (Oxford University Press, Oxford, 2005).

[10] *Physics Bimonthly*, No. 44, Vol. 5, p. 25.

[11] *Physics Bimonthly*, No. 44, Vol. 5, p. 19.

[12] Henry Su, code for azimuthal analysis (unpublished, 2025).