ChatGPT

Quantifying Droplet–Sediment Impact via High-Speed Imaging

Introduction

Understanding the dynamics of a water droplet impacting submerged sediments requires precise high-speed visualization and analysis. In this study setup, a single ~600 µm diameter water droplet is released from heights of 30–90 cm into a water-filled container with a bed of glass beads (simulating sediment) at the bottom. Upon impact, several phenomena occur in rapid succession: the droplet accelerates to a terminal velocity, the water's surface deforms (forming a bulge or crater and possibly a crown splash), and the sediment bed is disturbed or resuspended. Accurately quantifying each of these – (1) the droplet's velocity, (2) the size/shape of the surface displacement, and (3) the extent/distribution of sediment disturbance – is critical for understanding the impact physics. High-speed video at 10,000 fps (640×800 resolution) provides the raw data, but robust image analysis methods are needed to extract meaningful metrics from these rapid events. This report reviews **traditional image-processing techniques** and **modern deep-learning approaches** for each aspect, and offers recommendations on experimental design, imaging tools, and key parameters for reliable quantification.

Key challenges include the droplet's small size and high speed (requiring sharp imaging and tracking), the subtle deformation of the water surface (a transient bulge/crown that must be captured against a transparent medium), and the potentially diffuse sediment disturbance (a cloud of fine particles). We explore how classical methods (OpenCV-based tracking, edge detection, background subtraction, etc.) compare with advanced methods (object detection networks, segmentation models, optical flow) in terms of accuracy and efficiency. We also suggest improvements in **experimental design** (lighting, background, calibration) to optimize data quality. Summary tables are provided for a clear comparison of methods, tools, and trade-offs.

Droplet Velocity Measurement

Traditional Tracking Methods: The straightforward way to measure droplet velocity from high-speed video is to track the droplet's position frame-by-frame and compute its displacement over time. In practice, this can be done by segmenting the droplet in each frame and finding its centroid. For example, using OpenCV, one can apply intensity thresholding and contour detection to isolate the droplet (which, under backlighting, appears as a dark sphere) and then record its pixel coordinates [1]. The distance moved between successive frames (knowing the frame rate and a spatial calibration) yields the instantaneous velocity. Yu *et al.* (2016) describe this approach for rain droplets: the geometric center of each droplet image is located, and the velocity is calculated from the spacing of these centers in time [1]. This method is simple and accurate when the droplet has good contrast against the background and remains approximately spherical. Since the droplet here is small and fast, ensuring a high-contrast background (e.g. bright, uniform backlight) and using a short exposure to avoid motion blur are vital. Calibration (e.g. using a ruler or the known 600 µm drop size) should be done to convert pixel distances to physical units. Figure 1 shows how a high-speed sequence of a drop can be analyzed: the droplet's path is evident, and one can fit a line or polynomial to its vertical position vs. time to compute velocity (and even acceleration, if needed).

- **Optical Flow and Feature Tracking:** If simple thresholding is challenging (for instance, if the droplet is very faint or there is background noise), optical flow algorithms or feature tracking can be applied. Optical flow (e.g. Lucas-Kanade or Farnebäck in OpenCV) computes motion vectors for regions of the image and can indicate the droplet's movement between frames. This approach might help if the droplet's edges are indistinct, by effectively *following* the pattern of pixels that move downward. Similarly, one could use feature tracking (identifying a distinctive patch of the droplet or an interface and tracking that). However, for a single well-defined object like a droplet, these methods are generally less direct than centroid tracking – they may be unnecessary if segmentation is reliable. They also can be sensitive to other moving features (e.g. splashes or moving sediment later on) and thus require isolating the droplet in time (for velocity measurement, one would focus on frames *before* the drop impacts the water/sediment). In summary, traditional image processing (thresholding, blob detection, edge finding) is typically sufficient for droplet velocity and has the advantage of not requiring training data. **Table 1** summarizes methods for droplet velocity measurement, the tools involved, and their pros/cons.

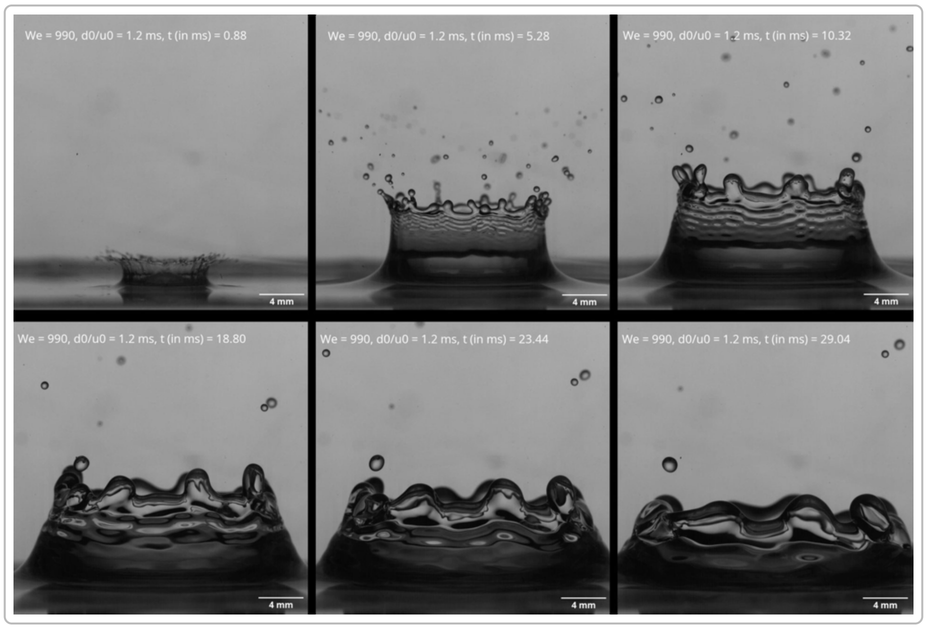
We = 990, d0/u0 = 1.2 ms, t (in ms) = 0.88
We = 990, d0/u0 = 1.2 ms, t (in ms) = 5.28
We = 990, d0/u0 = 1.2 ms, t (in ms) = 10.32
We = 990, d0/u0 = 1.2 ms, t (in ms) = 18.80
We = 990, d0/u0 = 1.2 ms, t (in ms) = 23.44
We = 990, d0/u0 = 1.2 ms, t (in ms) = 29.04
4 mm

Figure 1: Example high-speed image sequence of a water droplet (Weber number ≈990) impacting a water pool, creating an underwater cavity and an above-surface crown [2]. *By tracking the droplet's position in each frame prior to impact, one can compute its fall velocity. In this sequence (side view, backlit), the droplet pushes down to form a crater and a crown within ~10 ms; precise timing and displacement data can be extracted from such frames.* (Scale bar = 4 mm)

Modern Deep-Learning Approaches: Recent advances allow using object detection and tracking models to measure velocities, which can be especially useful if image quality is low or the droplet is hard to see. One approach is to use a pre-trained or custom-trained object detector (like YOLOv4/v5 or Mask R-CNN) to automatically locate the droplet in each frame. In a 2024 study by Burke *et al.* comparing deep detectors, YOLOv4 was shown to reliably detect droplets across varying image conditions, outperforming Mask R-CNN [3]. Once detected, the droplet's bounding box or mask provides its position and size. Combining this with a tracking algorithm (such as DeepSORT or a simple Kalman filter to link detections frame-to-frame) yields the trajectory. Durve *et al.* (2021) demonstrated a YOLO + DeepSORT pipeline that tracks droplets in complex flows with high accuracy, even using **synthetic training data** to avoid laborious manual labeling [4]. The advantage of deep learning is robustness: a well-trained model can handle variations in lighting, droplet appearance, or noise that might confound simple thresholding. For instance, if the droplet has nearly the same refractive index as the surrounding fluid (making it nearly invisible), classical methods might fail unless one artificially alters the optical setup. Deep neural networks, however, can learn subtle visual cues. In one microfluidics study, the Segment Anything Model (SAM) was used to detect droplet edges far more reliably than the Hough-transform method, especially in low-contrast images [5]. Traditional Hough detection often forces researchers to introduce artificial contrast (e.g. by refractive-index mismatch to create a visible outline), which *adds error* in locating the true interface [6]. A deep model can avoid such compromises and directly pinpoint the droplet. The trade-off is the need for training data: one might need to label droplet positions in a subset of frames or generate simulated images for training. Additionally, inference with a heavy model can be slower, but with modern GPUs, methods like YOLO can operate near real-time (and certainly fast enough for offline analysis of 10k fps footage) [7]. In this single-droplet scenario, deep learning is likely **not strictly necessary** if the imaging is optimized (since a single thresholded object is easy to track), but it could improve confidence in edge cases (e.g. very faint droplet or if multiple objects were present).

Table 1 below compares techniques for measuring droplet velocity:

Method	Tools/Techniques	Pros	Cons
Threshold & Centroid Tracking (static image processing)	OpenCV (threshold + `findContours` or blob detector); MATLAB/ImageJ equivalents	Simple and robust if droplet has high contrast; Provides direct position vs. time data [1]; No training data needed	Requires controlled lighting/background; Fails if droplet is not distinguishable (e.g. transparent in medium); Manual tuning of threshold may be needed
Optical Flow / Feature Tracking	OpenCV optical flow (Lucas-Kanade, Farnebäck); point trackers (`cv::calcOpticalFlowPyrLK` for features)	Can track motion even when shape is not distinct (follows texture); Useful if droplet cannot be easily segmented	May pick up other moving features or noise; Less direct – yields velocity field from which droplet motion must be extracted; Typically not needed for a single clear droplet

Method	Tools/Techniques	Pros	Cons
Deep Object Detection + Tracking (e.g. YOLOv4 with DeepSORT)	PyTorch/TensorFlow (YOLOv4/v5 detector); DeepSORT tracker or similar; or Mask R-CNN for segmentation	High accuracy and automation; Robust to noise, low contrast, and complex backgrounds [3] [4]; Once trained, can process many frames quickly (even real-time on GPU)	Significant setup: requires training data or synthetic data generation; Higher computational load (GPU often required); Overkill for one object in a simple scene unless contrast issues are severe
Segmentation Networks (e.g. U-Net or SAM)	Meta AI **SAM** (no training, prompt-based segmentation); U-Net or Mask R-CNN (trained on droplet masks)	Can precisely outline droplet for sub-pixel accuracy of centroid; SAM can be used with minimal effort to get droplet mask [5]; Useful if droplet edges need to be measured (e.g. shape, not just position)	SAM requires user prompt per frame (not fully automatic); Training a U-Net/ Mask R-CNN needs many labeled masks; Not necessary if only centroid/velocity is needed; Complex models may be slower per frame

Experimental Design Tips (Velocity): To optimize velocity measurements, ensure the droplet is **clearly visible** throughout its fall. Using backlighting with a diffusive screen will make the droplet appear as a dark silhouette against a bright background, ideal for thresholding. The camera and lens should be focused at the droplet's path (depth of field should cover from just below the release point to the water surface). If possible, use a telecentric lens or minimize perspective distortion so that the droplet's vertical motion translates linearly to pixel motion without scale changes. Markers or a scale (e.g. millimeter grid) in the frame can help convert pixel displacement to real units (mm). Starting the recording just before release and knowing the frame timestamp of impact can allow calculation of average acceleration and impact speed. If the droplet reaches a steady terminal velocity before impact, one might verify that against theoretical values or literature for consistency. In summary, a straightforward setup with good contrast and calibration should allow traditional tracking to yield highly accurate velocity data, while deep learning offers a fallback for more challenging imaging conditions.

Surface Displacement (Bulge/Crown) Analysis

When the droplet strikes the water surface, it causes a rapid deformation: the formation of a bulge or upward crown and a downward cavity. Even a small droplet can push down the water to form a dimple; from 30–90 cm drop height, a 600 μm droplet has enough inertia to create a noticeable disturbance of the air-water interface. Prior experiments have documented that as the droplet impacts, it *"pushed down below the surface to form a crater, while nearly at the same time a wall of liquid rose above the surface, forming a* *crown”* [8] . In our scenario (a deep water container), we expect an underwater cavity and a small surface bulge; the bulge might not be a dramatic crown with droplets (given the small drop size), but its height and radius are important. Quantifying the **size and shape** of this surface displacement involves capturing the interface profile in each frame.

Traditional Image Processing: The classic approach is to detect the water surface contour in the high-speed images and measure its deviation from the undisturbed state. If the imaging is done from the side with a backlight (as assumed), the water-air interface appears as a sharp line or boundary between bright and dark regions. One can utilize edge detection techniques like the Canny filter to trace this boundary [9] . In practice, a combination of thresholding and edge detection works well: Otsu’s method (automatic threshold) can differentiate the dark region of water+background from the bright region above the water [9] , and then Canny can find the outline of the bulge (the outline being where the water surface departs from its initial flat level). Another method is to subtract a reference image of the initial flat surface: by subtracting the pre-impact frame from a frame during impact, the difference image will highlight the bulging region (as a bright or dark patch). This can be binarized to get the bulge shape. Once the interface is identified, key geometric metrics can be measured: **maximum bulge height** (peak elevation above the original water level), **cavity depth** (how far the droplet pushes the surface down at the deepest point, if it forms a crater), and the **lateral extent** of the bulge (essentially the base diameter of the bulge or crown). High-speed imaging studies often extract these parameters frame-by-frame to see their evolution [2] . For example, MIT researchers captured the changing width and depth of the underwater cavity and the height/diameter of the rising crown over time using in-house image processing [2] . In our case, we would plot bulge height vs. time to find the peak, and similarly cavity depth vs. time. Assuming axial symmetry, the cross-sectional shape can even be revolved to estimate the displaced volume of water (volume of the bulge or cavity) if needed.

To implement this with OpenCV or similar tools, one could: (a) apply a ROI around the expected surface region, (b) use edge detection to get the interface line, (c) fit a smooth curve if necessary (polynomial or spline) to reduce noise, and (d) find key points like the apex. Dash *et al.* (2020) used a polynomial fit on the detected interface to measure contact angles in a droplet-on-solid impact [9] – a similar smoothing could improve accuracy of the bulge profile. It’s important to have sufficient resolution in the region of interest; the vertical resolution (800 px tall) should be concentrated around the surface if possible. If the entire 800 px frame covers the full drop path, the actual bulge might occupy only a small portion – considering cropping or using a zoom lens to focus on the air-water interface could improve measurement precision. Table 2 outlines methods for capturing the interface shape and their trade-offs.

Advanced/Modern Approaches: Deep learning can enhance interface detection, especially in cases of ambiguous visuals (e.g., poor contrast between water and background). One possible approach is to use a segmentation model to label each pixel as “water” or “air.” A custom-trained U-Net (with a few manually labeled frames of the water surface) could learn to output the exact outline of the water surface even when contrast is low or there is splashing. Another new approach is the **Segment Anything Model (SAM)** by Meta AI, which is a general segmentation tool that can produce masks given a prompt (like a point or rough box on the object of interest). For instance, by prompting SAM with a point on the crown or cavity region, it might output a mask covering the entire crown shape. Researchers have found SAM to provide *“superior detection and reduced error in droplet diameter measurement compared to Hough Transform, being more robust to low contrast between fluid phases”* [5] . This suggests SAM (or similar deep models) can pick up the true interface without needing to artificially enhance it (as mentioned earlier, classical methods might require adding a dye or refractive index difference to see the interface, which can introduce errors [6]).

Deep learning could also help measure more complex features of the bulge: for example, the **crown wall thickness** or inner flow that Bourouiba *et al.* measured [10]. Determining wall thickness might require either multiple camera angles or tracer particles, but a neural network could potentially infer it from subtle shading if trained appropriately. However, for a single small droplet impact, the bulge shape is relatively simple (likely a dome shape without an extreme crown), and classical edge detection should suffice. The benefit of deep learning might be more pronounced if the surface deformation is very subtle or if one aims to automate analysis over hundreds of videos with varying lighting (the network can generalize where threshold values might need manual adjustment per video).

One can also consider *optical methods* outside of conventional imaging: e.g., **schlieren or shadowgraphy** techniques can visualize density gradients and might highlight the droplet and the deforming interface. If available, a schlieren setup could capture the bulge as a refractive index change. Another method is **digital image projection**: projecting a grid or fringe pattern onto the water surface and analyzing its distortion (Moiré or fringe reflection techniques) to reconstruct surface topography. These methods can achieve very precise 3D surface measurements, but they require more complex setup and calibration, so they may be beyond the scope of a straightforward droplet experiment. Still, it’s worth noting that if ultra-high accuracy of the surface shape is needed, there are specialized techniques in fluid optics (often used in wave measurement or droplet splash studies) that could be employed.

For most purposes, the key output will be metrics like *max bulge height*, *max cavity depth*, *bulge lateral size*, and possibly *duration* of the deformation (how quickly it rises and subsides). If the droplet causes oscillations (capillary waves), one could measure the oscillation frequency or damping rate from the interface motion as well.

Table 2 provides a summary of methods for measuring the surface displacement geometry:

Method	Tools/Techniques	Pros	Cons
Edge Detection & Thresholding (classic contour extraction)	OpenCV (Canny edge detector; Otsu threshold) [9]; ImageJ (find edges)	Captures clear interface outlines; well-established method for droplet shapes [9]; High spatial precision if resolution is sufficient (can detect small bulge)	Requires good contrast between water and background (refractive or brightness difference); May pick up noise or secondary edges (needs smoothing); Fails if interface is very faint or if reflections cause false edges
Background Subtraction / Differencing	Reference frame subtraction; OpenCV `absdiff()` then threshold	Isolates the *change* due to the droplet – highlights the bulge region clearly against static background; Simple to implement and no complex parameters	Needs a stable reference (camera absolutely fixed, no ambient changes); Any lighting fluctuations or camera vibration will interfere; Only gives the *affected region* not the exact interface line (additional processing needed to get precise shape)

Method	Tools/Techniques	Pros	Cons
Manual or Semi-Manual Tracking of interface points	Marking key points on interface in each frame (e.g. using ImageJ or Tracker software)	Human insight can identify the interface even in ambiguous frames; Good for small number of frames or validation of automated method (manual measurement of bulge height/width)	Labor-intensive and subjective; Not feasible for thousands of frames; Less reproducible and lower temporal resolution (might skip frames to reduce labor)
Deep Segmentation (U-Net, Mask R-CNN trained on interface)	Python (PyTorch/ TensorFlow) with custom training on annotated masks; or fine-tuned Mask R-CNN on water surface class	Highly robust to noise and lighting variation once trained; Can capture complex shapes (e.g. crown with droplets) in one pass; Provides pixel-level mask of water vs air which yields all geometric metrics (height, width, area)	Requires a training dataset (which might involve labeling water surface in many images or generating synthetic ones); Training is time-consuming and requires expertise; Could overfit if not enough variety – may struggle if experimental conditions change (different lighting, etc.)
General Segmentation Models (e.g. SAM – Segment Anything Model)	SAM by Meta AI (no training required, uses prompts per image) [5]	No need to train a model for this specific task; With minimal input (points on the bulge area) it can delineate the entire bulge/crown shape automatically; Proven to outperform traditional Hough in droplet edge detection [5]	Not fully automatic across a video – requires providing prompts (though this could be semi-automated by using the previous frame's mask as a prompt for the next); Might sometimes split or merge regions if the prompt is not well-chosen; Because it's very general, it might not capture thin features like a very fine crown rim unless prompted carefully
Special Optical Methods (schlieren, Moiré fringes, etc.)	Schlieren imaging setup; Projected grid and camera (for Moiré or fringe reflection)	Can measure interface deflection with extremely high precision (down to micron scale in some cases); Schlieren can reveal even invisible density changes (helpful if droplet fluid has slightly different temperature or refractive index)	Additional complex setup required (aligned mirrors, knife edges for schlieren, or projector for fringe); Typically single-purpose and needs calibration; May be sensitive to vibrations and require a very controlled environment

Experimental Design Tips (Surface Measurement): To accurately capture the surface deformation, use a **high-contrast background** technique. One recommended setup is to place a diffused backlight opposite the camera, so the water and any submerged parts appear dark and the background is bright – the air-water interface will then appear as a well-defined silhouette. Minimize glare on the water surface by using polarizing filters or careful light positioning (glare can confuse edge detection). Ensure the camera is exactly level with the undisturbed water surface if you want to easily define the "zero" level in pixels (this can often be done by marking the water level before the drop). It's also useful to perform a static calibration: record an image of a ruler placed at the water surface to get pixel-to-mm scaling in the vertical direction at that plane. If the crown or bulge is expected to be axi-symmetric, you can assume the side-view profile is representative; if not (e.g., for very oblique impacts, which we don't have here), multiple views would be needed.

Triggering and synchronization are also important – the most severe surface deformation happens within the first few milliseconds of impact [8], so the camera's 10,000 fps is essential. Make sure the droplet enters the frame with some margin before impact so you have baseline frames and initial contact frames. If possible, use a camera with a faster frame rate or a burst mode around impact to capture even finer temporal resolution of the bulge formation (10 kfps might be sufficient given the time scales, but a 100 kfps recording of just the first 1–2 ms of impact, if available, could capture the very instant of cavity formation).

Finally, consider the fluid properties: since this is water on water, surface tension and viscosity effects will dominate the crown formation. If the droplet were a different fluid, the shape could differ; here, since it's the same fluid, the droplet will merge and primarily create a symmetric bulge. The metrics you gather (peak height, etc.) can be compared to dimensionless numbers like the Weber number (We) or Froude number to see if they collapse on known scaling laws. This can validate that your measurements are reasonable. For example, literature shows a Weber ~990 drop (much larger than ours) produces a sizable crown 【33†image】; our Weber might be smaller, but by comparing with known results we can ensure the bulge height scales appropriately with impact velocity.

Sediment Disturbance Quantification

The third challenge is to measure how the sediment bed is disturbed by the droplet impact. Even though the droplet is small, its impact can transmit energy through the water to the bed, potentially displacing beads, creating a local scour, or resuspending particles into the water column. In analogous scenarios like raindrops on soil or shallow pools, impacts *"dislodge surface particles"* and can even launch them into the air [11]. In our submerged case, we expect a localized eruption or agitation of the beads near the impact point. Quantifying the **extent** (how far and how deep) and **distribution** of this disturbance requires analyzing the motion of numerous small particles in the video frames.

Traditional Image Analysis: A practical starting point is **background subtraction** or frame differencing to identify regions of motion. Since the camera view is fixed on the sediment bed, one can take an image of the undisturbed bed (either a frame before the droplet hits, or an average of several pre-impact frames) as the "background." Once the droplet impacts, any movement of sediment will change the pixel intensities in those areas. By subtracting the background image from each subsequent frame, we obtain a "foreground" mask of changed pixels. Thresholding this difference highlights where movement occurs. Researchers have successfully used OpenCV's background subtraction in similar contexts – for example, to obtain moving sediment particles on a riverbed surface from video [12]. In our experiment, the difference images over time would show a growing patch of disturbance that might initially expand outward and upward, then dissipate as particles settle. By combining these masks over time (for instance, taking a logical OR of all motion masks or summing them), we can determine the total **affected area** on the bed. This could be represented as a radius of influence (e.g., the furthest distance from the impact point that any movement was detected) and as an area (in mm^2) of bed that was disturbed.

Another classic approach is to track **individual beads** (if the bead size and image resolution allow). If the glass beads are of order tens to hundreds of microns, they may appear as small specks in the images. One can use a blob detection or feature detection on each frame to identify bead centroids. Then, using a nearest-neighbor match or a tracking algorithm, link the positions of the same bead across frames (this is Particle Tracking Velocimetry, PTV). There are open-source libraries like `trackpy` in Python that are designed for tracking particles in fluid flows. PTV would yield trajectories for each identifiable bead – from which you could compute how far each bead traveled, its velocity, and where it settled. This gives the most detailed distribution of displacements (e.g., you could plot a histogram of bead travel distances or maximum elevation reached). However, PTV can be challenging if the particle density is high or if the particles are very small (they might appear and disappear or overlap). In a dense disturbance (many beads moving at once), PTV may not resolve individual paths well; background subtraction might be more robust for an overall picture.

An alternative to tracking every bead is **Particle Image Velocimetry (PIV)** or optical flow applied to the moving sediment as a continuum. PIV is commonly used in fluids to measure velocity fields by analyzing the motion of tracer particles between frame pairs. Here, the sediment particles can act as tracers of the flow (or of their own movement). Using PIV software (like OpenPIV or MATLAB's PIVlab), one can input two successive frames and obtain a grid of velocity vectors. Repeating over the sequence builds a time-varying velocity field of the sediment-laden flow. From this, one can derive quantities like the maximum velocity of sediment motion, and observe how the "cloud" of moving particles expands. For example, a high-speed droplet impact on a bed might generate a ring-like flow – PIV could reveal vortices or the propagation of that ring. The **extent** of disturbance could be quantified by the region where the flow velocity exceeds some small threshold. This is essentially measuring how far the kinetic effects reach. The caveat is that PIV (and optical flow) measures fluid motion as well – if water is moving but not all of it has sediment, you might measure flow beyond where sediment actually moves. Still, it's a useful tool to identify the dynamic range of the impact.

Using simpler optical flow (like OpenCV's Farnebäck dense flow) can similarly highlight moving areas: one can compute the flow field and then create a mask of all regions with movement above noise level. This might be quicker than full PIV and can run directly in Python on the video frames.

Modern/Deep Learning Approaches: Deep learning can assist in two ways: by improving the detection of moving particles (segmentation/classification) or by yielding more accurate motion estimation. For detection, one could train a convolutional neural network to classify regions of a frame as "sediment disturbed" vs "undisturbed." This could be done with a CNN sliding over the image or as a segmentation model labeling each pixel. However, creating a ground truth segmentation of disturbed sediment is labor-intensive (you'd have to manually label what pixels have moving particles in many frames). Instead, a more feasible deep approach is to use **deep optical flow** models. Networks like RAFT or PWC-Net (pre-trained on generic motion) can take two frames and output a very detailed flow field. These often outperform classical optical flow in capturing fine motions. Applying such a model to consecutive frames of the experiment could produce a high-fidelity map of particle motions, even with motion blur or low contrast. From the flow field, one would then extract similar metrics – e.g., region of significant motion, speed distribution, etc.

Another deep approach is object detection at the particle level: for instance, a trained YOLO network could potentially detect individual sediment grains or clumps of grains in motion. This would be like treating each moving bead as an object. In practice, unless the beads are large and distinct, this might not work well; it's more suited if you had, say, larger gravel pieces where you want to detect each piece. In our case, the disturbance might look more like a cloud. A segmentation network might better handle that by labeling the whole cloud region.

One interesting possibility is using deep learning to **estimate concentration** of sediment from images. If one had calibration data (images with known concentrations of suspended particles), a neural network could potentially learn to map image intensity or texture to a concentration value. Then you could get a quantitative measure of how much sediment mass is suspended at various times and locations. This goes beyond simple visualization – it would turn the image into data about sediment quantity. There is research in water treatment and environmental monitoring that uses imaging to estimate sediment or turbidity; a deep model could improve accuracy by accounting for lighting, overlapping particles, etc. Again, that requires training data or at least a careful calibration.

For a comprehensive analysis, one might combine methods: use deep optical flow to get the motion vectors and then integrate or threshold those to define the disturbed region. Deep learning could also help filter out spurious changes (like if lighting changes cause false positives in background subtraction). A trained model might distinguish actual sediment movement from noise.

In summary, traditional methods like frame differencing give a quick map of disturbance, whereas deep methods can potentially extract richer and cleaner information, at the cost of more setup. Table 3 compares several approaches for analyzing sediment disturbance:

Method	Tools/Techniques	Pros	Cons
Background Subtraction & Difference Mask	Static background image or running average; OpenCV `BackgroundSubtractor` (MOG2/KNN) [12]	Simple implementation; Highlights all moving regions clearly (gives an "area of disturbance"); Can accumulate changes over time to get total affected zone	Sensitive to lighting changes and noise (requires stable illumination); Cannot distinguish depth (just 2D area in view); May include non-sediment motion (e.g. the droplet or bubble) unless masked out

Method	Tools/Techniques	Pros	Cons
Particle Tracking (PTV)	Trackpy (Python); custom centroid detection + nearest-neighbor matching	Yields detailed trajectories of individual sediment grains; Can compute distribution of displacements, velocities, etc.; Very quantitative (each particle's data)	Difficult if particle density is high or particles are small (tracking can fail when paths cross or particles vanish); Computationally intense for many particles; Requires good image resolution of each bead
Particle Image Velocimetry (PIV) / **Optical Flow** (classical)	OpenPIV library; PIVlab (MATLAB); OpenCV optical flow (Farnebäck)	Provides a full **velocity field** of the sediment-laden flow; Captures the dynamics (vortices, jet, etc.) not just the extent; Well-proven in fluid mechanics experiments	Resolution of vectors limited by interrogation window (PIV divides image into patches); Cannot easily separate actual sediment motion from overall water motion; May smooth out localized effects (gives average motion in each area)
Deep Optical Flow (e.g. RAFT)	Pre-trained RAFT model on frame pairs (via PyTorch)	Extremely accurate motion detection, even for small or textureless regions; Can handle large displacements better than basic optical flow; Produces dense motion vectors that can be thresholded to find moving regions	Requires using a neural network (more setup and computing power); Might need adaptation if scene differs greatly from training data (though RAFT is quite general); Still does not inherently differentiate sediment vs fluid – just gives motion everywhere

Method	Tools/Techniques	Pros	Cons
Deep Motion Segmentation (CNN-based)	Custom CNN or background subtraction via deep autoencoder (unsupervised)	Potential to **learn** the specific appearance of moving sediment vs static bed (could filter out false positives due to lighting); Once trained, could directly output a binary mask of disturbed area each frame; Unsupervised models (autoencoders) can learn background and detect novel motion	Hard to get training labels for supervised segmentation; Unsupervised models might classify any unusual change as foreground (which is similar to classical subtractors anyway); Complexity likely unjustified unless classical methods are failing due to noise
Object Detection for Beads/Clusters	YOLO or similar, trained to detect individual beads or clumps in motion	Could track larger sediment pieces or clusters even when they move in groups; Deep detectors can be scale- and rotation-invariant, potentially tracking particles through collisions or agglomeration	Impractical for very fine or numerous particles (model would need to detect hundreds of tiny objects with overlapping signals); Requires extensive training data (synthetic or annotated) to reliably detect beads in various states; Likely not feasible for sand-like fine sediment

Experimental Design Tips (Sediment): Capturing sediment motion is challenging – good experimental design can greatly improve the image analysis. **Lighting** is crucial: if using backlighting, note that a cloud of sediment will appear as a darker region (because particles block light). This is good for background subtraction (dark cloud vs bright background). Make sure the background illumination is uniform across the bed area so that intensity changes truly reflect sediment presence. Another approach is front-lighting from the top or side, which might make particles sparkle or show up bright against a darker background. Depending on bead material (glass beads might be transparent), you might consider mixing a small fraction of colored or opaque tracer particles to make the motion more optically visible. For instance, a few percent of the beads could be black or fluorescent – their motion could be tracked and considered representative of the bulk movement. If using fluorescing tracers, you could employ a filter to only image those, thereby visualizing the sediment cloud distinctly (requires UV illumination and appropriate camera filter).

Minimize water movement or vibrations before the drop so that any detected motion is solely from the droplet impact. It may be useful to take a “difference” not just from the initial frame, but also after everything settles, to see net displacement. For example, after the event, once particles resettle, take an image of the final bed and compare it to the initial bed image – this could show if a crater or rearrangement remains. If a permanent crater forms in the sediment, you can measure its diameter and depth (possibly by looking at the side view or feeling the bed). However, given the tiny scale (0.6 mm droplet), any sediment displacement might be subtle. Using high-speed video primarily tells us about the transient cloud of sediment in the water. If quantifying the **mass** of sediment resuspended is of interest, consider an independent measurement: for instance, you could use turbidity sensors or even carefully remove and weigh sediment that got suspended. But within the images, a proxy for mass could be the integrated intensity of the sediment cloud (darker cloud = more particles, if calibrated).

Calibration in this context can mean knowing the size of one pixel in the plane of the sediment bed (to measure distances of spread) and possibly the depth of field (to know how many layers of particles might be captured – though with one camera it's hard to separate depth). If the bed disturbance is axisymmetric, you can just measure radius in the 2D image (side view radius corresponds to actual radius if the cross-section is through the center). Ensure the droplet ideally impacts at the center of the frame and directly above the point you analyze, to maintain symmetry in the side view.

Finally, consider the time aspect: you may want to quantify not just the maximum disturbance but also how quickly sediment resettles. This could be done by tracking the total area of moving sediment vs. time or the average intensity of the sediment cloud vs. time. A rapid rise and exponential decay in turbidity might be observed. These are additional metrics that describe the distribution in time (e.g., the **duration** of suspension or the settling rate). They complement the spatial metrics (radius, area, etc.).

Conclusion and Recommendations

Most Accurate & Reliable Methods: For each aspect of the droplet-sediment interaction, a combination of careful experimental setup and appropriately chosen analysis methods will yield accurate quantification. For the **droplet velocity**, a straightforward centroid tracking after thresholding is typically very accurate (error on velocity can be just a few percent [13] [14]). Given controlled lighting and calibration, this method is both simple and reliable. Deep learning-based tracking can match or exceed this accuracy in more complicated scenes, but for a single droplet on a uniform background, the traditional approach is sufficient and practically noise-free. We recommend using the classical tracking as a baseline, and possibly verifying a subset of data with a deep learning approach (to ensure no systematic bias – e.g., one could run a YOLO detector on a few frames to see if it agrees on the droplet position). Ensuring the camera is calibrated and the frame timestamps are known (to compute velocity = distance / time) is essential; using the high frame rate to your advantage, you can get a smooth velocity curve rather than just an average speed.

For the **surface displacement (bulge)**, edge detection on high-speed images has been proven in literature to capture crater and crown dimensions [2]. The key to accuracy is image quality: a high-contrast silhouette will allow sub-pixel interpolation of the interface position, whereas a low-contrast edge could introduce uncertainty. If the interface is difficult to see, consider using a slight dye in the droplet (if it doesn't alter physics significantly) to make the droplet fluid visible against the water – though since it's the same liquid, this won't help much for the air-water interface. Instead, focus on lighting and perhaps use a shallow angle of illumination to cast a shadow of the bulge. In terms of methods, a **hybrid approach** can be powerful: use threshold/edge detection to get an initial contour, then apply a polynomial fit or smoothing spline to refine it (reducing pixelation noise). This combines the objectivity of automation with some post-processing to improve reliability. Deep learning segmentation could provide a one-click solution for all frames, but the development overhead is only justified if you plan to analyze many impacts or if the images are too noisy for classical methods. If so, a model like SAM could be a quick win – you might interactively obtain the bulge outline for each impact event without full training.

For **sediment disturbance**, multiple methods should be used in complement. Our recommendation is to start with **background subtraction to get the overall affected area** (this gives a primary measure of extent). Then, if resolution allows, attempt **particle tracking or PIV** in the region of interest to get more detailed metrics like velocity of particles and distribution. PIV is likely to succeed even if individual particles can't be tracked, and will provide a velocity magnitude map. From that, one can derive secondary metrics: e.g., the maximum uplift velocity of sediment, which could correlate with particle size to see if some beads might escape the bed. PTV, if feasible, would directly give the distribution of displacements – for example, "the droplet impact moved beads up to ~5 mm from their original positions, with most beads moving ~1–2 mm". These kinds of statements are very valuable in interpreting the effect. If PTV is not feasible due to density, using deep optical flow (RAFT) is a good surrogate: it may "virtually" track particles in a dense flow by virtue of dense correspondence. The deep flow results could then be clustered to identify groups of moving particles.

Experimental Optimizations: We reiterate the importance of a **controlled lighting setup** and a clean, contrast-enhancing background. Use the highest practical resolution in the region of interest – if 640×800 is the maximum at 10 kfps, consider if a slightly lower frame rate could afford a higher resolution for certain experiments (for example, 5 kfps at higher resolution) to capture fine details of sediment motion, if the timescales allow. However, the initial impact is so fast that 10 kfps is likely needed; thus, one could do two sets of experiments: one focused on the immediate impact (at max fps, somewhat lower res) and another on the slower sediment settling (which could be recorded at lower fps but higher resolution or with a second camera). Additionally, using a high-speed **macro lens** or microscope lens for the sediment bed could give more detail on particle motion (though covering a smaller field of view). If the budget and logistics permit, a second synchronized high-speed camera viewing from **above** could directly capture the radial spread of the sediment cloud on the bed. This top view, combined with the side view, would enable 3D reconstruction of the disturbed volume of sediment (through simple geometry or even stereoscopic pairing of views).

Deep Learning in Perspective: Deep learning methods can significantly improve analysis *when* the imaging data is suboptimal or the scenario is complex (e.g., many overlapping phenomena). In this relatively controlled experiment, traditional methods supplemented by careful setup can achieve high accuracy. Deep learning shines in automating analysis for large data volumes and handling variations without constant re-tuning. If this graduate research will involve many droplet trials under different conditions (varying droplet sizes, different sediments, etc.), investing in a deep learning pipeline could save time long-term and provide consistency. A possible strategy is to create a modest training dataset from a representative subset of your videos – label droplet positions, water interface, and disturbed sediment regions in those frames – and train specialized models for each task (or a multi-task model). Then use those models to process all future videos quickly and uniformly. This would ensure objectivity (reducing user bias in picking thresholds or edges) and likely improve the precision of detection (e.g., sub-pixel accuracy from neural networks). The trade-off is the upfront effort and the requirement of computational resources.

In conclusion, by combining high-speed imaging with smart analysis techniques, one can comprehensively quantify the droplet impact process: the droplet's **velocity** profile (for example, confirming it reaches ~2–4 m/s before impact, consistent with theoretical free-fall or drag considerations), the **surface deformation** geometry (peak bulge on the order of perhaps a few hundred microns tall given the drop size, to be measured precisely), and the **sediment disturbance** footprint (likely a few millimeters in radius, depending on impact energy). Key parameters like those should be reported along with uncertainties. Using the methods and suggestions outlined – summarized in the tables above – will ensure that the measurements are accurate and repeatable. By optimizing both the experiment (lighting, calibration, camera settings) and the analysis (choosing the right image processing or deep learning tool for each job), the study can yield high-quality quantitative data on how a tiny droplet can mobilize sediments underwater. This not only serves the immediate research questions but also provides methodological insights for similar fluid–sediment interaction experiments in the future.

---

[1] [13] [14] amt.copernicus.org
https://amt.copernicus.org/preprints/amt-2015-396/amt-2015-396.pdf

[2] [8] [10] [11] High-speed videos show what happens when a droplet splashes into a pool - cee.mit.edu
https://cee.mit.edu/high-speed-videos-show-what-happens-when-a-droplet-splashes-into-a-pool/

[3] Two deep learning methods in comparison to characterize droplet sizes in emulsification flow processes | Journal of Flow Chemistry
https://link.springer.com/article/10.1007/s41981-024-00330-3

[4] [7] Tracking droplets in soft granular flows with deep learning techniques | The European Physical Journal Plus
https://link.springer.com/10.1140/epjp/s13360-021-01849-3

[5] [6] Enhancing Microdroplet Image Analysis with Deep Learning
https://www.mdpi.com/2072-666X/14/10/1964

[9] An image processing method to measure droplet impact and evaporation on a solid surface | Sādhanā
https://link.springer.com/article/10.1007/s12046-020-01520-0

[12] Study on the motion characteristics of bedload sediment particles ...
https://onlinelibrary.wiley.com/doi/10.1002/rvr2.90