U-LAG: Uncertainty-Aware, Lag-Adaptive Goal Retargeting for Robotic Manipulation

Anamika J H, Anujith Muraleedharan

***Abstract*— Robots often grasp with observations that are delayed, noisy, or stale. We present U-LAG, a mid-execution goal–retargeting layer that leaves the low-level controller unchanged while re-aiming task goals (pre-grasp hover, grasp, post-lift) as fresh observations arrive. Unlike visual servoing or online re-planning that redesign control or regenerate trajectories, U-LAG treats in-flight goal re-aiming as a pluggable module between perception and control. Our main technical contribution is UAR–PF, an uncertainty-aware retargeter that maintains a belief over object pose under sensing lag and selects goals that maximize expected capture, evaluated against ICP and nearest-geometry baselines within the same interface. We instantiate a reproducible Shift×Lag stress test in PyBullet/PandaGym for pick and place, where the object undergoes abrupt in-plane shifts while synthetic perception lag is injected during approach. Across 0–10 cm shifts and 0–400 ms lags, UAR–PF degrades gracefully relative to a no-retarget baseline and achieves higher success with modest end-effector travel and few aborts; simple operational safeguards further improve stability.**

I. INTRODUCTION

Perception delay and scene changes routinely invalidate contact goals during manipulation. Setpoints that were safe at plan time can be out of date by the time the end effector reaches them, yielding missed grasps and wasted motion. Classical visual servoing closes the loop directly on image or deep features [1]–[3]. Task and motion planning commits to goal waypoints and relies on replanning when disruptions are large [4], [5].

We advocate a simple separation of concerns: treat mid-execution goal re-aiming as a distinct module between perception and control. Rather than redesigning controllers, a retargeting layer updates pre-grasp, grasp, and post-lift goals whenever new observations arrive, and an unchanged Cartesian servo executes those goals. This viewpoint is complementary to visual servoing and planning.

We instantiate this idea as U-LAG (Fig. 1), a lag-adaptive, uncertainty-aware goal-retargeting layer that consumes RGB–D point clouds and outputs refreshed task goals for pick and place. The layer admits multiple realizations within a common interface, including deterministic nearest-geometry updates, point-to-point ICP alignment for registering earlier and current clouds [6], and a particle-filtered variant that maintains a belief over object pose under delay [7], [8]. To probe delay and disturbance in a controlled way, we introduce a Shift×Lag stressor in PyBullet and PandaGym [9]. At the approach trigger, we apply an in-plane object shift and hold actions to a lagged target for a fixed duration; the camera then re-captures and the retargeter refreshes goals. Over 0–10 cm shifts and 0–400 ms lags, U-LAG variants degrade gracefully versus a no-retarget baseline, with PF and ICP remaining robust in harder cells. Though evaluated on a fixed arm, the formulation transfers to mobile manipulation where ego motion amplifies perception delay [10], [11].

A. Related Work

Robotic systems have long closed the perception–action loop with visual servoing (VS), where image measurements directly drive velocity or pose commands [1]–[3]. Classic IBVS/PBVS continuously regulate feature errors, while learning variants replace hand-crafted features with deep descriptors or keypoints to improve generalization (e.g., DFVS/KOVIS) [3]. Predictive VS explicitly handles moving targets or delays by forecasting image features over a short horizon [5], [12]. These threads adapt the control law itself to changing observations. We keep the controller fixed and refresh pre, grasp, and post-contact goals via a mid-execution retargeter. A second strand updates goals by re-planning. Reactive grasping often re-optimize grasps or short trajectories online as new state estimates arrive, recovering from perturbations and clutter interactions (e.g., real-time grasp re-planning in clutter; fast online grasp synthesis) [13], [14]. Task-and-motion planning with feedback similarly interleaves planning and execution, periodically re-solving for waypoints as the world changes [4], [5]. These systems typically regenerate motion plans or grasps, which can be costly under tight control loops. U-LAG targets a lighter-weight middle ground: keep the motion primitive, but re-aim its goals immediately using a pluggable map from observations to updated setpoints.

Reliable retargeting needs uncertainty-aware state estimation; particle-filter/RBPF trackers maintain 6-DoF pose beliefs and smooth delayed/noisy measurements [7], [8]. Alignment methods such as point-to-point ICP (and its robust variants) remain a strong baseline for reconciling stale and fresh clouds after discrete shifts [6]. Those methods internalize lag within the controller design. Our method is orthogonal: keep the controller fixed; update goals using lagged, uncertainty-aware cues. Prior work does not, to our knowledge, elevate goal retargeting to a modular mid-execution layer or compare ICP, nearest-geometry, and PF realizations under controlled shift–lag perturbations.

B. Contributions

(1) **U-LAG**: a modular goal–retargeting layer between perception and control that re-aims pre/grasp/post waypoints under delay. (2) **UAR–PF**: an uncertainty-aware retargeter using a particle-filter belief to maximize expected capture

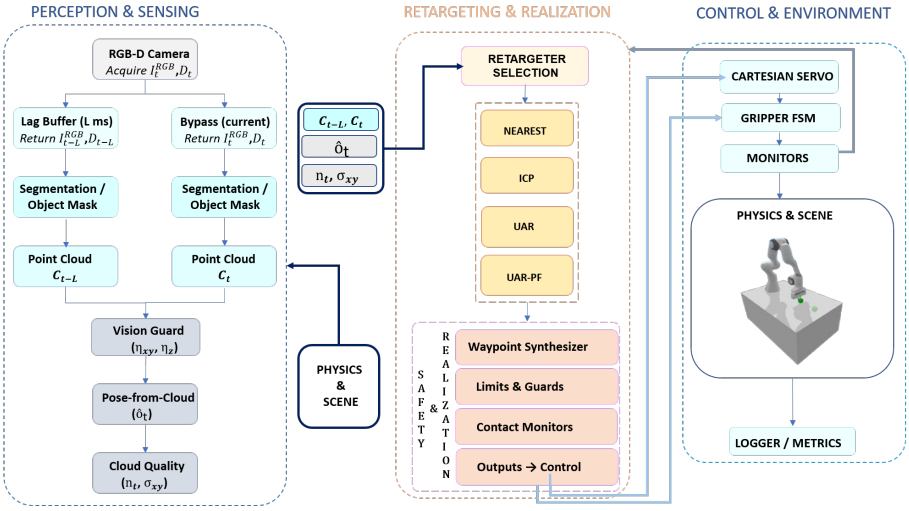


Fig. 1. **U-LAG system overview.** Left: perception & sensing. Center: retargeters (Nearest/ICP/UAR/UAR–PF) and shared guards. Right: unchanged Cartesian servo & gripper FSM executing waypoints.

under lag. (3) **Shift×Lag benchmark**: a reproducible pick-and-place stress test in PyBullet/PandaGym where UAR–PF reliably outperforms ICP/Nearest with low compute overhead.

II. METHODOLOGY

A. Problem Formulation

We consider a fixed-base manipulator interacting with a rigid object whose center at time t is $\mathbf{o}_t = [x_t, y_t, z_t]^\top$ in world frame $\mathcal{W}$. A task specifies a grasp goal $\mathbf{g}$ (position+orientation). Low-level control is an unchanged first-order Cartesian position servo at 100 Hz. Perception provides a depth image and segmented point cloud $C_t = \{\mathbf{p}_k\}$, but with latency L ms (freshest usable cloud is C_{t-L}). During the approach phase the object may shift abruptly in-plane, making precomputed goals stale. We insert a pluggable *retargeting* map

$$\mathcal{R}\colon \big(C_{t-L}, C_t, \text{state}\big) \mapsto \big(\mathbf{g}^{\text{pre}}, \mathbf{g}^{\text{grasp}}, \mathbf{g}^{\text{post}}\big), \qquad (1)$$

and keep control, kinematics, and gripper FSM fixed. We assume a calibrated camera and a reliable object segment (available in sim; any detector can provide it on hardware).

B. Sensing & State Estimation

A calibrated RGB–D camera provides a segmented point cloud $C_t = \{\mathbf{p}_k \in \mathbb{R}^3\}$ at control time t, subject to latency L (the freshest usable cloud is C_{t-L}). When C_t has not yet arrived, we fall back to the most recent cloud.

1) Pose proxy & cloud quality:

- **Pose proxy:** From C_t we trim depth outliers by percentile clipping (1st/99th) to get $\tilde{C}_t$, form an axis-aligned bounding box with corners $\boldsymbol{\ell}_t, \mathbf{h}_t$, and set the center $\hat{\mathbf{c}}_t = \frac{1}{2}(\boldsymbol{\ell}_t + \mathbf{h}_t)$. For a cube of edge a, the vertical coordinate uses the top face: $\hat{z}_t = h_t^z - \frac{a}{2}$, giving $\hat{\mathbf{o}}_t = [\hat{c}_t^x, \hat{c}_t^y, \hat{z}_t]^\top$ with a top-down grasp orientation. If $|\tilde{C}_t|$ is too small, the proxy is marked invalid.
- **Latency use:** Retargeters consume both the delayed cloud C_{t-L} and the freshest available cloud C_t (or the last received cloud if C_t is missing).
- **Quality metrics:** We summarize reliability with $n_t = |\tilde{C}_t|$ and lateral dispersion $\sigma_{xy} = \sqrt{\lambda_{\max}(\Sigma_{xy})}$ where $\Sigma_{xy} = \mathrm{cov}\big(\{[p_k^x, p_k^y]^\top : \mathbf{p}_k \in \tilde{C}_t\}\big)$; depth noise is $\sigma_z = \mathrm{std}(\{p_k^z : \mathbf{p}_k \in \tilde{C}_t\})$ (or a task-fixed constant when unstable). These drive UAR/UAR–PF margin inflation and the PF observation covariance.
- **Vision guard:** The proxy and quality metrics feed a simple guard that rejects gross outliers; see Sec. II-E.

C. Goal Geometry

Pick (pre-grasp, grasp, post-lift).: Let the sensed proxy be $\hat{\mathbf{o}}_t = [\hat{x}_t, \hat{y}_t, \hat{z}_t]^\top$ (Sec. II-B), and let a be the cube edge. Define the top plane as $z_{\text{top}} = \hat{z}_t + \frac{a}{2}$. Let offsets be a hover clearance δ_{hover}, an approach depth δ_{approach} (into the object from top plane toward its center), and a lift height δ_{lift}.

$$\mathbf{g}^{\text{pre}} = \big[\hat{x}_t,\ \hat{y}_t,\ z_{\text{top}} + \delta_{\text{hover}}\big]^\top, \qquad (2)$$

$$\mathbf{g}^{\text{grasp}} = \big[\hat{x}_t,\ \hat{y}_t,\ z_{\text{top}} - \delta_{\text{approach}}\big]^\top, \qquad (3)$$

$$\mathbf{g}^{\text{post}} = \big[\hat{x}_t,\ \hat{y}_t,\ z_{\text{top}} + \delta_{\text{lift}}\big]^\top. \qquad (4)$$

Uncertainty-aware scaling.: When cloud quality (n_t, σ_{xy}) is available, inflate δ_{hover} and δ_{approach} as $\delta' = \delta_0 + \lambda\sigma_{xy}$ (clamped to task bounds) to reduce collision risk under noise.

D. Retargeters $\mathcal{R}$

Each retargeter maps the delayed/fresh clouds (C_{t-L}, C_t) and controller state to updated waypoints $(\mathbf{g}^{\text{pre}}, \mathbf{g}^{\text{grasp}}, \mathbf{g}^{\text{post}})$, and is wrapped by the same vision guard and two-stage waypointing (Sec. II-E).

1) Nearest: Recompute all three waypoints directly from the latest pose proxy $\hat{\mathbf{o}}_t$ via the closed-form construction in Sec. II-C. Complexity $\mathcal{O}(1)$; negligible latency. Strong when segmentation is clean and lag is modest. Limitations: after large teleports or transient proxy bias, it can aim off-center.

2) ICP Retargeting (ICP): Register C_{t-L} to C_t with point-to-point ICP (standard outlier rejection and early stopping). Transport the stale pre-grasp through the estimated rigid transform; recompute grasp/post from the latest proxy (Sec. II-C). Complexity $\mathcal{O}(I\,m)$ with a small cap on ICP iterations I and cloud size m; adds noticeable latency only at high L. Strengths: robust when cloud overlap is high. Limitations: large teleports reduce overlap and can degrade alignment.

3) Uncertainty-Aware Retargeting (UAR): Inflate approach margins using the cloud dispersion σ_{xy} from Sec. II-B.1. For $\delta \in \{\delta_{\text{hover}}, \delta_{\text{approach}}\}$ use $\delta' = \text{clip}(\delta_0 + \lambda\,\sigma_{xy})$ (and optionally $\delta'_{\text{lift}} = \text{clip}(\delta_{\text{lift},0} + \frac{1}{2}\lambda\,\sigma_{xy})$), with clamps from Table I. Complexity $\mathcal{O}(1)$; improves capture by reducing side contact and under-reach under noisy clouds. Limits: without temporal smoothing it can still react to transient bias.

4) UAR with Particle Filter (UAR–PF): Maintain N particles over the object center with small Gaussian process noise; weight by a Gaussian likelihood whose covariance uses $(\sigma_{xy}^2, \sigma_{xy}^2, \sigma_z^2)$ (Sec. II-B.1); low-variance resample when $\text{ESS} < N/2$. Instantiate Sec. II-C at the belief mean and apply the same UAR margin inflation. Complexity $\mathcal{O}(N)$ with $N \in [64, 128]$; adds only tens of milliseconds in our setting. Strengths: smooths guarded/noisy measurements and remains stable under large lags and intermittent dropouts.

E. Reliability mechanisms

We add two lightweight, implementation-agnostic safeguards; they do not change the low-level controller and are not claimed as contributions.

a) Vision guard.: Compare the fresh proxy $\hat{\mathbf{o}}_t$ to a trusted reference (e.g., previous estimate); if $\|\Delta\hat{\mathbf{o}}_{xy}\| > \tau_{xy}$ or $|\Delta\hat{z}| > \tau_z$, discard the frame and fall back. Thresholds are set large to catch only gross outliers.

b) Two-stage waypointing (UP→XY→DOWN).: Retargeted grasps are executed via a safe height z_{safe} before lateral motion and descent, reducing side contact. We enable the two-stage sequence when the sensed correction is large or near side surfaces; otherwise we go direct-to-center-Z. We perform exactly one retarget update immediately after the injected lag.

F. Control Interface

At each control tick t (period Δt), with EE pose $\mathbf{x}_t$ and active target $\mathbf{g}_t$, the Cartesian servo commands

$$\Delta\mathbf{x}_t = \text{clip}_{\Delta_{\max}}(\mathbf{g}_t - \mathbf{x}_t), \quad \text{clip}_{\Delta_{\max}}(\mathbf{v}) = \min\Big(1, \frac{\Delta_{\max}}{\|\mathbf{v}\|_2}\Big)\mathbf{v},$$

and updates $\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta\mathbf{x}_t$ ($\Delta_{\max}$ in Table I). We advance to the next waypoint when $\|\mathbf{x}_{t,xy} - \mathbf{g}_{t,xy}\|_2 \le \epsilon_{xy}$ and $|x_t^z - g_t^z| \le \epsilon_z$ (Table I). Latency L is simulated by holding the stale goal for $N = \lfloor L/\Delta t \rfloor$ ticks, then performing a single retarget update via $\mathcal{R}$ (Sec. II-D). The gripper opens at pre-grasp, closes at grasp, and opens at post-lift.

TABLE I
PARAMETERS AND DEFAULTS (PICK).

Category	Default / Description
Object & clearances	Cube edge $a = 0.04\,\text{m}$; vertical clearance 1 cm.
Waypoint offsets (pick)	$\delta_{\text{hover}} = 30\,\text{mm}$, $\delta_{\text{approach}} = 2\,\text{mm}$, $\delta_{\text{lift}} = 80\,\text{mm}$.
Two-stage tolerances	Arrival $(\epsilon_{xy}, \epsilon_z) = (15, 10)\,\text{mm}$; direct-grasp $(5, 6)\,\text{mm}$.
Two-stage activation	XY jump threshold 30 mm; side-band margin 2 mm; side proximity margin 3 mm; finger radius 7.5 mm.
Vision guard	Large thresholds (e.g., $\tau_{xy} = 0.45\,\text{m}$, $\tau_z = 0.45\,\text{m}$) to act only on outliers.
UAR	Margin slope λ (tuned per task); optional clamps $\underline{\delta}, \bar{\delta}$.
UAR–PF	Particles $N \in [64, 128]$; low-variance resample if $\text{ESS} < N/2$.

III. EXPERIMENTAL RESULTS

A. Setup and Protocol

a) Environment & Task: We use PyBullet/PandaGym with a Franka Panda and a single fixed RGB–D pinhole camera (known intrinsics/extrinsics, 60° FOV) providing depth and point clouds. Control runs at 100 Hz using a first-order Cartesian position servo (with a 3-DoF compatibility shim). We evaluate top-down pick-and-place of a 4 cm cube, where the retargeter updates pre-grasp, grasp, and post-lift waypoints executed by the unchanged servo.

b) Shift×Lag stressor: When the end-effector (EE) reaches the approach trigger, we (i) teleport the cube by a random in-plane vector of fixed magnitude $r \in \{0, 2, \ldots, 10\}$ cm, then (ii) hold actions against the stale target for a synthetic perception delay $L \in \{0, 100, 200, 300, 400\}$ ms. After the lag elapses, the camera re-captures and the retargeter updates the goals.

c) Baselines / Retargeters: We compare the four retargeters from Sec. II-D and a no-retarget baseline; all modes share the same vision guard and two-stage waypointing.

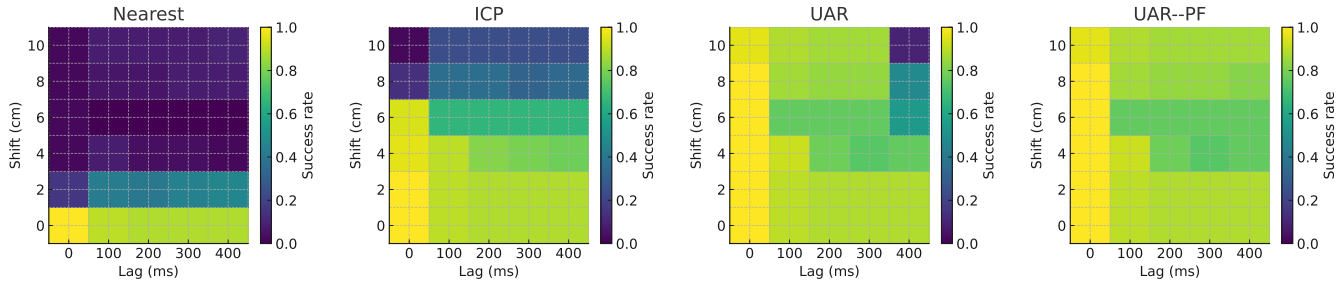
d) Grid / Trials: We sweep 6 shifts × 5 lags with 50 random seeds per cell. Each run logs success, retarget latency, end-effector (EE) travel, shift, lag, and mode to CSV.

e) Metrics: We report success rate with Wilson 95% confidence intervals (per cell, n=50), mean retarget latency, and mean EE travel.

f) Runtime: All retargeters run in a few milliseconds at our cloud sizes: Nearest/UAR $\mathcal{O}(1)$, ICP $\mathcal{O}(I\,m)$ with a small fixed cap $I \le 30$, and UAR–PF $\mathcal{O}(N)$. Compute overhead is negligible relative to the injected perception lag and the 100 Hz control loop.

B. Quantitative Results

a) Headline results.: UAR–PF maintains high success across the grid, including the hardest cells. At (r=10 cm, L=400 ms) it achieves 0.86 (95% CI 0.74–0.93). At (8 cm, 300 ms) it reaches 0.84 (0.71–0.92), while geometry-only methods degrade. Average EE travel is similar across modes, indicating robustness without longer paths.



(a) Nearest (b) ICP (c) UAR (d) UAR–PF

Fig. 2. **Pick & Place: success rate** over Shift×Lag grid (50 seeds/cell). A common color scale is used across panels.

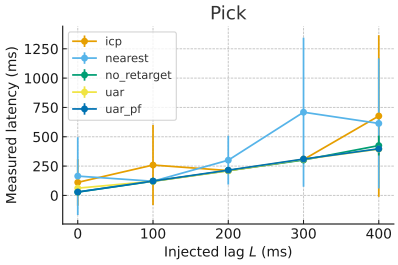


Fig. 3. **Pick: retarget latency vs. injected L.** Means with error bars (std). ICP shows additional overhead at high L.

b) Mode comparison.: Nearest works for small shifts but collapses after large teleports. *ICP* is competitive when cloud overlap is high but struggles for big teleports. *UAR* improves safety margins yet can overreact under the largest shift+lag. *UAR–PF* is best or tied-best overall due to belief smoothing.

C. Retarget Latency

Measured retarget latency closely tracks the injected lag L (Fig. 3); for UAR–PF, means are approximately $29\,\text{ms}$, $122\,\text{ms}$, $215\,\text{ms}$, $310\,\text{ms}$ and $397\,\text{ms}$ at $L = \{0, 100, 200, 300, 400\}$ ms. ICP adds extra cost at high L due to correspondence search and rigid solve.

D. Failure Modes

Main failures: over-guarded resets at large L (under-reach) and side contacts during XY correction; rarer: premature closure and fixture jams. Two-stage UP→XY→DOWN curbs side/jam events, and UAR–PF smoothing mitigates noisy/guarded updates.

IV. CONCLUSIONS

U-LAG is a lightweight goal-retargeting layer that updates pre-grasp, grasp, and post-lift waypoints online under delayed/noisy perception while leaving the low-level controller unchanged. We instantiate four retargeters (Nearest, ICP, UAR, and UAR–PF) with simple reliability hooks for stable grasp. On a Shift×Lag stress test in PyBullet/PandaGym for pick-and-place, all retargeters substantially outperform a no-retarget baseline; UAR–PF is most robust in the hardest regimes, achieving higher success with comparable EE travel and only tens of milliseconds of compute overhead. Future work includes hardware deployment, extensions to mobile manipulation, and learned uncertainty models for online retargeter selection.

REFERENCES

[1] S. Hutchinson, G. D. Hager, and P. I. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, 1996.

[2] F. Chaumette and S. Hutchinson, "Visual servo control. I. basic approaches," *IEEE Robotics & Automation Magazine*, vol. 13, no. 4, pp. 82–90, Dec. 2006.

[3] ——, "Visual servo control. II. advanced approaches," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 109–118, Mar. 2007.

[4] L. P. Kaelbling and T. Lozano-Pérez, "Hierarchical task and motion planning in the now," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 1470–1477.

[5] ——, "Integrated task and motion planning in belief space," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1194–1227, 2013.

[6] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[7] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian bayesian state estimation," *IEE Proceedings F – Radar and Signal Processing*, vol. 140, no. 2, pp. 107–113, 1993.

[8] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[9] G. Gallouédec, N. Cazin, E. Dellandrèa, and L. Chen, "panda-gym: Open-source goal-conditioned environments for robotic learning," arXiv preprint arXiv:2106.13687, 2021. [Online]. Available: https://arxiv.org/abs/2106.13687

[10] D. Falanga, S. Kim, and D. Scaramuzza, "How fast is too fast? the role of perception latency in high-speed sense and avoid," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1884–1891, 2019.

[11] R. Aldana-López, A. Gonzalez Forero, L. Alvarez-Icaza, and N. E. Leonard, "Latency vs precision: Stability-preserving perception–action loops," *Automatica*, vol. 152, p. 110972, 2023.

[12] J. Mahler, M. Matl, V. Satish, X. Wang, M. Van der Hoek, J. Evett, N. Ratliff, and K. Goldberg, "Dex-Net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[13] D. Morrison, J. Leitner, and P. Corke, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1272–1279, 2018.

[14] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.