Technical Report

A Vision-Based Analysis of Congestion Pricing in New York City

Mehmet Kerem Turkcan[1], **Jhonatan Tavori**[2], **Javad Ghaderi**[3], **Gil Zussman**[4], **Zoran Kostic**[5] **and Andrew Smyth**[6]

[1,2,3,4,5,6]*Columbia University, New York, USA*

Abstract

We examine the impact of New York City's congestion pricing program through automated analysis of traffic camera data. Our computer vision pipeline processes footage from over 900 cameras distributed throughout Manhattan and New York, comparing traffic patterns from November 2024 through the program's implementation in January 2025 until January 2026. By excluding anomalous periods such as holidays, we establish baseline traffic patterns and identify systematic changes in vehicle density across the monitored region.

Keywords: *computer vision, urban planning, congestion pricing, traffic analysis, smart cities, urban mobility*

Website: https://mkturkcan.github.io/congestionpricing/
DOI: https://www.doi.org/10.57967/hf/4448

1. Introduction

Urban congestion pricing has emerged as a key policy tool for managing traffic flow in major metropolitan areas, following successful implementations in Singapore (1975), London (2003), Stockholm (2006), Milan (2008), and Gothenburg (2013) [1]. On November 14, 2024, New York City announced its own congestion pricing program, effective January 5, 2025, marking the first such implementation in a North American city [2]. Although previous studies have examined congestion pricing effects through manual traffic counts, sensor networks, or limited camera deployments, a visual analysis of city-wide traffic patterns remains challenging due to the scale of data collection and processing needed.

We address this challenge by developing a computer vision pipeline that leverages New York City's existing network of 910 traffic cameras provided by webcams.nyctmc.org [3]. Our methodology enables systematic measurement of traffic density changes both within and around the designated Congestion Relief Zone, while addressing technical challenges inherent in processing low-resolution traffic camera feeds at scale. This approach offers three key contributions:

1. A single-stage object detection model (YOLO-LR) optimized for processing low-resolution (352×240) traffic camera footage available to the public, achieving improved detection accuracy compared to standard high-resolution models;
2. A scalable infrastructure for real-time processing of hundreds of simultaneous video feeds using distributed computing;
3. A methodology for analyzing traffic pattern changes that accounts for temporal variations and systematic biases in camera-based measurements.

By establishing this framework, we enable quantitative evaluation of large-scale traffic policy interventions using existing urban camera infrastructure. Our analysis provides insights into the effects of the implementation of congestion pricing and the spatial redistribution of traffic patterns throughout the metropolitan area.

Algorithm 1 Traffic Pattern Analysis

```
Require:
 1: $D = \{d_1, ..., d_n\}$                              ▷ Raw traffic count observations
 2: $t = \{t_1, ..., t_n\}$                              ▷ Timestamps for each observation
 3: $S = \{s_1, ..., s_m\}$                              ▷ Set of traffic count sources
 4: $w$                                                  ▷ Rolling window size
 5: $T_{split}$                                          ▷ Split timestamp
 6: procedure PROCESSTRAFFICDATA($D, t, S, w, T_{split}$)
 7:     for each source $s \in S$ do
 8:         $\bar{D}_s \leftarrow \text{RollingMean}(D_s, w)$   ▷ Smooth counts for source s
 9:     end for
10:     $H \leftarrow \{0, ..., 23\}$                    ▷ Hours of day
11:     $W \leftarrow \{\text{weekday}, \text{weekend}\}$   ▷ Day types
12:     $P \leftarrow \{\text{before}, \text{after}\}$     ▷ Split periods
13:     for each source $s \in S$ do
14:         for each $(h, w, p) \in H \times W \times P$ do
15:             $D_{s,h,w,p} \leftarrow \{d_i \in \bar{D}_s | \text{Hour}(t_i) = h, \text{DayType}(t_i) = w, t_i \in p\}$
16:             $\mu_{s,h,w,p} \leftarrow \text{Mean}(D_{s,h,w,p})$
17:         end for
18:     end for
19:     for each time window $[h_1, h_2] \in \text{TimeWindows}$ do
20:         for each $(s, w) \in S \times W$ do
21:             $\text{Peak}_{s,w,before} \leftarrow \max_{h \in [h_1, h_2]} \mu_{s,h,w,before}$
22:             $\text{Peak}_{s,w,after} \leftarrow \max_{h \in [h_1, h_2]} \mu_{s,h,w,after}$
23:             $\Delta_{peak,s,w} \leftarrow \text{Peak}_{s,w,after} - \text{Peak}_{s,w,before}$
24:         end for
25:     end for
26:     return $\{\Delta_{peak,s,w} | s \in S, w \in W\}$   ▷ Peak differences for each source
27: end procedure
28: where:
29:   $\text{TimeWindows} = \{[0, 23], [6, 9], [9, 15], [15, 18]\}$   ▷ Day, Morning, Midday, Afternoon
30:   $\text{RollingMean}(X, w) = \frac{1}{w} \sum_{i=0}^{w-1} x_{t-i}$   ▷ w-point moving average
```

2. Algorithm

2.1. Object Detection

We address the technical challenges of processing low-resolution traffic camera footage by developing YOLO-LR, an adaptation of YOLO optimized for low-resolution input. Of the 910 cameras in our dataset, 770 (84.6%) output footage at 352×240 pixels, significantly below standard object detection dataset resolutions. We train our model on the COCO dataset at this resolution, specifically focusing on five relevant classes: bicycle, car, motorcycle, bus, and truck. Images are resized and padded to 352×352 resolution. We show comparative evaluation metrics in Table 1 against the standard YOLO model trained at 640×640 resolution. Our results show that the YOLO-LR model demonstrates improved performance on low-resolution traffic footage.

2.2. Data Collection Infrastructure

Our data collection system employs a distributed architecture with 16 parallel workers for frame capture and processing. The pipeline processes captured frames in batches of 64 using the YOLO-LR model compiled using TensorRT, with automatic discarding of frames ex-

Table 1. Performance comparison of YOLO models across different classes in COCO val.

Class	Instances	YOLO-LR (imgsz=352)			YOLO11n (imgsz=352)			YOLO11n (imgsz=640)		
		R	mAP50	mAP50-95	R	mAP50	mAP50-95	R	mAP50	mAP50-95
All	3296	0.488	0.537	0.357	0.420	0.467	0.305	0.523	0.597	0.413
Bicycle	314	0.341	0.387	0.227	0.277	0.332	0.188	0.392	0.478	0.280
Car	1918	0.481	0.513	0.305	0.388	0.410	0.237	0.523	0.582	0.375
Motorcycle	367	0.537	0.605	0.373	0.512	0.577	0.338	0.585	0.672	0.440
Bus	283	0.678	0.740	0.595	0.601	0.652	0.537	0.707	0.772	0.647
Truck	414	0.403	0.437	0.287	0.321	0.363	0.224	0.408	0.480	0.326

ceeding a 100ms download threshold. All processing is performed on a single compute node equipped with dual NVIDIA A100 40GB GPUs. Detection results are stored in a SQLite database for subsequent analysis. All frames are discarded after processing.

2.3. Pattern Analysis

Our pattern analysis methodology addresses three key challenges in urban traffic analysis: temporal variability, spatial heterogeneity, and measurement noise. Algorithm 1 details our approach to quantifying traffic pattern changes through a multi-scale temporal analysis framework.

To quantify congestion dynamics, we introduce Peak Hour Differentials (PHD), a metric that captures intervention-induced changes in vehicle densities, while accounting for temporal dependencies and systematic biases in camera-based observations.

Given a traffic source s and time t, let $D_s(t)$ denote the vehicle count at time t, and define the rolling mean vehicle density over a window of size ω as:

$$\bar{D}_s(t) = \frac{1}{\omega} \sum_{i=0}^{\omega-1} D_s(t-i), \tag{1}$$

To assess congestion changes before and after intervention, we define the time-partitioned expectation of vehicle density for a given source s, hour h, and day type w as:

$$\mu_{s,h,w,p} = \mathbb{E}[D_s(t) \mid \text{Hour}(t) = h, \text{DayType}(t) = w, t \in p], \tag{2}$$

where p denotes either the pre- or post-intervention period.

For each source s and day type w, we compute peak densities over specified time windows $[h_1, h_2]$, where

$$\text{Peak}_{s,w,p} = \max_{h\in[h_1,h_2]} \mu_{s,h,w,p}. \tag{3}$$

The Peak Hour Differential (PHD), measuring congestion shifts post-intervention, is then given by

$$\text{PHD}_{s,w} = \text{Peak}_{s,w,\text{after}} - \text{Peak}_{s,w,\text{before}}. \tag{4}$$

This metric provides three key advantages: (i) it directly quantifies changes in peak congestion, the primary target of pricing interventions, while maintaining clear physical interpretation in terms of vehicle density, (ii) by computing peaks within defined windows rather than at fixed times, it captures potential shifts in peak timing induced by the intervention, and (iii) uses a rolling mean to counter short-term fluctuations and anomalies.

3. Limitations

Our approach has some important limitations: (i) stationary vehicles are included in traffic density calculations, which may affect measurements in areas with high street parking density, (ii) the baseline comparison period (starting mid-November) provides limited seasonal context, (iii) the current implementation analyzes aggregate traffic flow without distinguishing between individual lanes or travel directions, (iv) camera-based instantaneous vehicle counts serve as a proxy measure and may not directly correlate with actual transit times or congestion levels, (v) New York experienced unusually cold weather during January-February in 2025, which could have affected commuter behavior significantly.

References

[1] L. Lehe, "Downtown congestion pricing in practice", *Transportation Research Part C: Emerging Technologies*, vol. 100, pp. 200–223, 2019.

[2] Metropolitan Transportation Authority, *Congestion relief zone*, https://congestionreliefzone.mta.info/, Accessed: 2025-02-13.

[3] *Real-time traffic information*, https://webcams.nyctmc.org, Accessed: 2025-02-13.