Binary TTC: A Temporal Geofence for Autonomous Navigation
二進制 TTC：自主導航的時間地理圍欄

Abhishek Badki, Orazio Gallo, Jan Kautz, Pradeep Sen

1 NVIDIA
2 UC Santa Barbara

CVPR 2021 Best Paper Candidate
https://arxiv.org/abs/2101.04777
https://github.com/NVlabs/BiTTC
https://www.youtube.com/watch?v=uUQJcjyerM4

Abstract 摘要

Time-to-contact (TTC), the time for an object to collide with the observer's plane, is a powerful tool for path planning:
接觸時間 (TTC)，即物體與觀察者平面碰撞的時間，是路徑規劃的強大工具：

it is potentially more informative than the depth, velocity, and acceleration of objects in the scene—even for humans.
它可能比場景中物體的深度、速度和加速度提供更多信息——即使對於人類也是如此。

TTC presents several advantages, including requiring only a monocular, uncalibrated camera.
TTC 具有多種優勢，包括只需要一個單目、未校準的相機。

However, regressing TTC for each pixel is not straightforward, and most existing methods make over-simplifying assumptions about the scene.
然而，回歸每個像素的 TTC 並不簡單，大多數現有方法對場景做出了過度簡化的假設。

We address this challenge by estimating TTC via a series of simpler, binary classifications.
我們通過一系列更簡單的二元分類估計 TTC 來解決這一挑戰。

We predict with low latency whether the observer will collide with an obstacle within a certain time, which is often more critical than knowing exact, per-pixel TTC.
我們以低延遲預測觀察者是否會在特定時間內與障礙物發生碰撞，這通常比知道精確的每像素 TTC 更重要。

For such scenarios, our method offers a temporal geofence in 6.4 ms—over 25faster than existing methods.
對於這種情況，我們的方法在 6.4 毫秒內提供了時間地理圍欄——比現有方法快 25 倍。

Our approach can also estimate per-pixel TTC with arbitrarily fine quantization (including continuous values), when the computational budget allows for it.
當計算預算允許時，我們的方法還可以通過任意精細的量化（包括連續值）來估計每像素 TTC。

To the best of our knowledge, our method is the first to offer TTC information (binary or coarsely quantized) at sufficiently high frame-rates for practical use.
據我們所知，我們的方法是第一個以足夠高的幀速率提供 TTC 信息（二進製或粗量化）以供實際使用的方法。

This work was done while A. Badki was interning at NVIDIA.
這項工作是在 A. Badki 在 NVIDIA 實習期間完成的。

Project page: https://github.com/NVlabs/BiTTC

1. Introduction 前言

Path planning, whether for robotics or automotive applications, requires accurate perception, which in turn, benefits from depth information.
路徑規劃，無論是機器人還是汽車應用，都需要準確的感知，而這反過來又受益於深度信息。

Many modalities exist to infer depth.
存在許多模式來推斷深度。

Strategies such as lidar estimate depth accurately but only at sparse locations, in addition to being expensive.
光學雷達等策略可以準確估計深度，但僅限於稀疏位置，而且成本高昂。

Depth can also be estimated with strategies such as stereo [35], but these introduce issues such as calibration drift over time.
深度也可以通過立體聲 [35] 等策略進行估計，但這些會引入諸如校準隨時間漂移之類的問題。

An alternative is to use a monocular camera—an attractive, low-cost solution, with light maintenance requirements.
另一種選擇是使用單目相機——一種有吸引力的低成本解決方案，維護要求低。

The motion of the camera induces optical flow between consecutive frames, which carries information on the scene's depth.
相機的運動會導致連續幀之間的光流，它攜帶有關場景深度的信息。

Depth, however, can only be estimated in the constrained case of static scenes.
然而，深度只能在靜態場景的受限情況下進行估計。

For dynamic scenes, the 2D flow of a pixel is a function of its depth, its velocity, and the velocity of the camera.
對於動態場景，像素的 2D 流是其深度、速度和相機速度的函數。

Disentangling these three components is an under-constrained and challenging problem.
解開這三個組成部分是一個約束不足且具有挑戰性的問題。

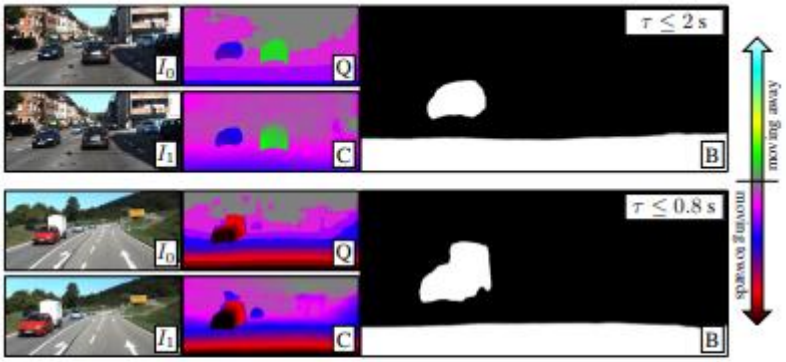


Figure 1: Given I_0 and I_1, our binary time-to-contact (TTC) estimation acts as a temporal geofence detecting objects that will collide with the camera plane within a given time, B. It only takes 6.4 ms to compute. Our method can also output quantized TTC, Q, and continuous TTC, C.

Figure 1: Given I0 and I1, our binary time-to-contact (TTC) estimation acts as a temporal geofence detecting objects that will collide with the camera plane within a given time, B .
圖 1：給定 I0 和 I1，我們的二進制接觸時間 (TTC) 估計充當時間地理圍欄檢測將在給定時間 B 內與相機平面碰撞的對象。

It only takes 6.4 ms to compute. Our method can also output quantized TTC, Q , and continuous TTC, C .
計算只需要 6.4 毫秒。 我們的方法還可以輸出量化的 TTC，Q 和連續的 TTC，C。

Previous approaches either ignore dynamic regions [36], or use strong scene priors [27, 45, 47, 44, 26] to hallucinate their depth.
以前的方法要么忽略動態區域 [36]，要么使用強場景先驗 [27、45、47、44、26] 來幻覺它們的深度。

Do we really need to disentangle them? The role of perception is to inform decisions.
我們真的需要解開它們嗎？ 感知的作用是為決策提供資訊。

An object moving towards the camera is more critical than another that is potentially closer, but moving away from the camera.
朝向相機移動的物體比可能更近但遠離相機的另一個物體更重要。

Differently put, predicting the time at which an object would make contact with the camera may be more valuable than knowing its actual depth, velocity, or acceleration [23].
換句話說，預測物體與相機接觸的時間可能比了解其實際深度、速度或加速度更有價值 [23]。

In fact, time-to-contact (TTC), the time for an object to collide with the camera plane under the current velocity conditions, is a traditional concept in psychophysics [39, 14] as well as computer vision [38, 5].
事實上，接觸時間 (TTC)，即在當前速度條件下物體與相機平面碰撞的時間，是心理物理學 [39, 14] 和計算機視覺 [38, 5] 中的傳統概念 .

TTC can be estimated from the ratio of an object's depth and its velocity relative to the camera, even when the problem of regressing either one independently is ill-posed.
TTC 可以根據物體的深度與其相對於相機的速度的比率來估計，即使獨立回歸的問題是不適定的。

However, TTC estimation has its own challenges, forcing most of the existing approaches to severely constrain their scope.
然而，TTC 估計有其自身的挑戰，迫使大多數現有方法嚴格限制其範圍。

For instance, they assume that the scene is static, or that a mask for dynamic objects is provided [16, 33].
例如，他們假設場景是靜態的，或者提供了動態對象的掩碼 [16, 33]。

The recent approach by Yang and Ramanan tackles some of these challenges by learning a mapping between optical flow and TTC directly, thus producing a per-pixel TTC estimate [43].
Yang 和 Ramanan 最近的方法通過直接學習光流和 TTC 之間的映射來解決其中的一些挑戰，從而產生每像素 TTC 估計 [43]。

However, it relies on accurate optical flow estimation and inherits its limitations, including its heavy computational load.
然而，它依賴於精確的光流估計並繼承了其局限性，包括其繁重的計算負載。

Unlike most existing approaches, we side-step the need to explicitly compute the optical flow.
與大多數現有方法不同，我們迴避了明確計算光流的需要。

Our learning-based approach estimates per-pixel TTC directly from images.
我們基於學習的方法直接從圖像估計每像素 TTC。

We leverage the relationship between an object's TTC and the ratio of the size of its image in different frames [5, 15].
我們利用對象的 TTC 與其圖像在不同幀中的大小比率之間的關係 [5, 15]。

However, because regressing this scale factor exactly is challenging, we focus on whether the size of the

object's image is increasing, indicating a collision at some time in the future, or decreasing, indicating that the object is moving away.
然而，因為準確地回歸這個比例因子是具有挑戰性的，所以我們關注對像圖像的大小是在增加，表明未來某個時間發生了碰撞，還是在減小，表明對象正在遠離。

More concretely, inspired by Badki et al. [1], we perform a series of binary classifications with respect to different scale factors, each corresponding to a specific TTC.
更具體地說，受 Badki et al. [1] 的啟發，我們針對不同的比例因子執行了一系列二元分類，每個分類因子對應一個特定的 TTC。

Each classification yields a binary TTC map with respect to the desired time threshold, Figure 1, insets B .
每個分類產生一個關於所需時間閾值的二進制 TTC 圖，圖 1，插圖 B。

Our binary map, efficient to compute, acts as a temporal geofence in front of the camera:
每個分類產生一個關於所需時間閾值的二進制 TTC 圖，圖 1，插圖 B。

it identifies objects within a given TTC, in 6.4 ms.1
它在 6.4 毫秒內識別給定 TTC 內的對象。

This is useful when a quick reaction time is important.
當快速反應時間很重要時，這很有用。

We can also estimate per-pixel TTC with arbitrary quantization, as shown in Figure 1, insets Q , or continuous, Figure 1, insets C .
我們還可以通過任意量化來估計每像素 TTC，如圖 1 中的插圖 Q 所示，或連續的，圖 1 中的插圖 C 。

These different levels of quantization, from binary to continuous, can be predicted with the same core network.
這些不同級別的量化，從二進製到連續，都可以使用相同的核心網絡進行預測。

In fact, quantization levels can be added, removed, or moved dynamically at inference time, based on the current needs of the autonomous agent.
事實上，可以根據自主代理的當前需求在推理時動態添加、刪除或移動量化級別。

Given the scarcity of TTC ground truth data, to impose additional inductive bias to our network we also introduce binary optical flow estimation as an auxiliary task.
鑑於 TTC 地面實況數據的稀缺性，為了對我們的網絡施加額外的歸納偏差，我們還引入了二進制光流估計作為輔助任務。

We achieve competitive performance for TTC estimation against existing methods, even stereo-based

methods, but we are from 25 to several orders of magnitude faster.
我們在 TTC 估計方面取得了與現有方法（甚至是基於立體的方法）的競爭性能，但我們的速度提高了 25 到幾個數量級。

2. Related Work 相關工作

While valuable for navigation, 3D information about the scene is challenging to gather with a monocular camera.
雖然對導航很有價值，但使用單目相機收集有關場景的 3D 資訊具有挑戰性。

Existing methods assume the scene to be static [9, 40, 17, 25], or estimate single-image relative depth, rather than metric depth [8, 11, 13, 10, 34].
現有方法假設場景是靜態的 [9, 40, 17, 25]，或者估計單幅圖像的相對深度，而不是度量深度 [8, 11, 13, 10, 34]。

Single-image relative depth can be “upgraded” to metric by computing the optical flow between multiple images [27, 45, 47, 44, 26], but this requires strong priors, and it is brittle for complex, large motions.
通過計算多幅圖像之間的光流 [27, 45, 47, 44, 26] 可以將單幅圖像的相對深度“升級”為度量，但這需要強大的先驗，並且對於複雜的大運動來說很脆弱。

Nevertheless, with depth maps and optical flow we can estimate scene flow, which captures both depth and velocity [37].
通過計算多幅圖像之間的光流 [27, 45, 47, 44, 26] 可以將單幅圖像的相對深度“升級”為度量，但這需要強大的先驗，並且對於複雜的大運動來說很脆弱。

A recent approach by Hur and Roth elegantly combines these principles by learning scene priors to decompose the depth and the velocity directly from the estimated optical flow information [18].
Hur 和 Roth 最近的一種方法通過學習場景先驗來直接從估計的光流信息中分解深度和速度，巧妙地結合了這些原則 [18]。

Instead of regressing depth and velocity, we focus on estimating their ratio directly from images, which yields timeto-contact (TTC).
我們沒有回歸深度和速度，而是專注於直接從圖像中估計它們的比率，從而產生接觸時間 (TTC)。

We do this via multiple binary classifications.
我們通過多個二元分類來做到這一點。

Here we discuss the state-of-the-art in terms of these two axes—TTC and regression via classification.
在這裡，我們根據這兩個軸來討論最先進的技術—TTC 和通過分類回歸。

2.1. Time-to-Contact 聯繫時間

Time-to-contact (TTC) was studied in psychophysics and psychology, even before it attracted the attention of the computer vision community.
接觸時間（TTC）在心理物理學和心理學中已經得到研究，甚至在它引起計算機視覺社區的注意之前。

Early work by Lee, for instance, suggested that TTC is sufficient for making decisions about braking, and is likely to be picked up by the driver faster than distance, speed, or acceleration of objects in the scene [23].
例如，Lee 的早期工作表明，TTC 足以做出有關製動的決定，並且駕駛員可能比場景中物體的距離、速度或加速度更快地接收到它[23]。

From a computational standpoint, TTC is appealing because it only depends on the ratio of depth and velocity, which can be estimated directly from images, even when estimating either one is an ill-posed problem [16].
從計算的角度來看，TTC 很有吸引力，因為它只取決於深度和速度的比率，這可以直接從圖像中估計，即使估計其中一個是不適定問題 [16]。

Several traditional approaches have been proposed to estimate TTC from the estimated optical flow [32, 33, 3].
已經提出了幾種傳統方法來從估計的光流中估計 TTC [32, 33, 3]。

Horn et al. proposed a direct method for TTC estimation that only uses the constant brightness assumptions [15, 16].
Horn et al. 提出了一種直接的 TTC 估計方法，該方法僅使用恆定亮度假設 [15, 16]。

While these approaches estimate the TTC for an object that is moving relative to the camera, they require masks for dynamic objects, which limits their practical impact.
雖然這些方法估計相對於相機移動的對象的 TTC，但它們需要動態對象的掩碼，這限制了它們的實際影響。

We propose a learning-based approach for TTC estimation that handles multiple dynamic objects—without needing any segmentation—and estimates per-pixel TTC.
我們提出了一種基於學習的 TTC 估計方法，該方法可以處理多個動態對象——無需任何分割——並估計每個像素的 TTC。

The elegant, closely related work by Yang and Ramanan estimates optical flow, uses it to compute the scaling factor of objects, and maps it to TTC [43].
Yang 和 Ramanan 的優雅、密切相關的工作估計了光流，使用它來計算對象的縮放因子，並將其映射到 TTC [43]。

Xu et al. also estimate the scaling of objects, but do so by modeling the change of objects' size explicitly for optical flow estimation [42].
徐等人。 還估計對象的縮放比例，但通過為光流估計明確建模對像大小的變化來實現[42]。

However, computing the full optical flow is timeconsuming, and estimating TTC from optical flow inherits its limitations.
然而，計算完整的光流很耗時，並且從光流中估計 TTC 繼承了它的局限性。

Instead, following Horn et al. [16], we compute TTC directly from the input images, side-stepping optical flow computation altogether.
相反，跟隨 Horn et al. [16]，我們直接從輸入圖像計算 TTC，完全側步光流計算。

Moreover, inspired by Badki et al. [1], we solve TTC estimation via a series of binary classifications.
此外，受 Badki et al. [1] 的啟發，我們通過一系列二元分類解決了 TTC 估計問題。

Each binary classification can be computed independently of the others at over 150 fps.
每個二元分類都可以以超過 150 fps 的速度獨立計算。

This can be thought of as a temporal geofence, detecting pixels or objects within a given TTC.
這可以被認為是一個時間地理圍欄，檢測給定 TTC 內的像素或對象。

For existing methods, including the method by Yang and Ramanan [43], this can only be achieved by computing the TTC for all the pixels or objects in the scene, and then thresholding it.
對於現有的方法，包括 Yang 和 Ramanan [43] 的方法，這只能通過計算場景中所有像素或對象的 TTC，然後對其進行閾值處理來實現。

###
1
On an NVIDIA Tesla V100 for 3841152 images.
在 NVIDIA Tesla V100 上用於 3841152 張圖像。

2.2. 3D Inference as a Classification Problem - 作為分類問題的 3D 推理

The idea of posing 3D regression as a classification task has a rich history.
將 3D 回歸作為分類任務的想法有著豐富的歷史。

Several learning-based approaches estimate depth via a multi-class classification task [21, 4, 46, 20].
幾種基於學習的方法通過多類分類任務估計深度 [21, 4, 46, 20]。

These are more accurate than other learning-based approaches that pose the problem as a regression task [30, 7].
這些比將問題作為回歸任務提出的其他基於學習的方法更準確 [30, 7]。

Badki et al. introduced a method that allows us to control the trade-off between latency and accuracy [1].

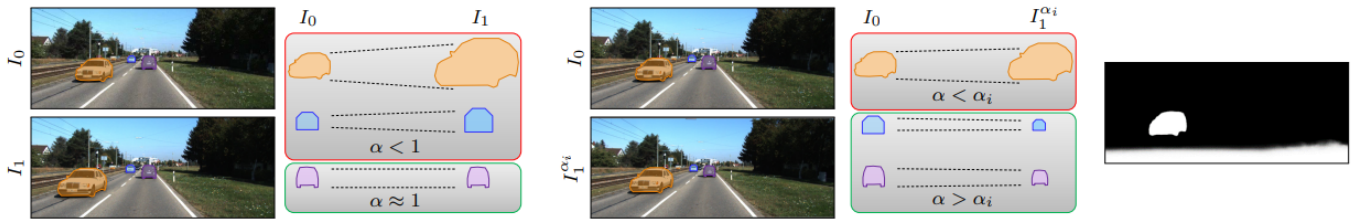
Badki et al. 引入了一種方法，允許我們控制延遲和準確性之間的權衡 [1]。

Instead of posing depth as a multi-class classification problem, they solve it via multiple binary classifications.
他們沒有將深度視為多類分類問題，而是通過多個二元分類來解決它。

Each classification provides useful information about the scene at high frame-rates.
每個分類都提供有關高幀率場景的有用信息。

Our approach is based on the same intuition.
我們的方法基於相同的直覺。

We are the first learning-based approach to estimate perpixel TTC directly from the input images, and to pose it as a (binary) classification problem.
我們是第一個直接從輸入圖像估計每像素 TTC 並將其作為（二元）分類問題的基於學習的方法。



(a) Inputs (b) Scaled inputs (c) Temporal geofence

Figure 2: **Intuition.** Given two images of a dynamic scene, I_0 and I_1, we define a temporal geofence to detect objects expected to cross the camera plane before a given time τ_i from the time of capture of I_0. Compare I_0 and I_1, (a). The images of the orange and blue cars are smaller in I_0 ($\alpha < 1$), while the image of the purple car is roughly unchanged. This allows us to predict that only the first two cars will collide with the camera plane. We propose to perform this comparison after scaling the source image by a factor α_i (corresponding to TTC value τ_i). The orange car is still larger in $I_1^{\alpha_i}$ ($\alpha < \alpha_i$), indicating that it will cross the camera plane *before* the specified τ_i. On the other hand, the blue and purple cars will not. Rather than regressing the exact scale factor, we classify a pixel as making contact before or after τ_i by classifying if the objects scale up or down in $I_1^{\alpha_i}$ with respect to I_0. This yields a binary TTC probability map for τ_i, (c).

Figure 2: Intuition. 圖 2：直覺

Given two images of a dynamic scene, I0 and I1, we define a temporal geofence to detect objects expected to cross the camera plane before a given time i from the time of capture of I0.
給定動態場景的兩個圖像 I0 和 I1，我們定義了一個時間地理圍欄來檢測預期在從捕獲 I0 的時間開始的給定時間 i 之前穿過相機平面的對象。

Compare I0 and I1, (a). The images of the orange and blue cars are smaller in I0 (α< 1), while the image of the purple car is roughly unchanged.
比較 I0 和 I1，(a)。 橙色和藍色汽車的圖像在 I0 中較小（ α <1），而紫色汽車的圖像大致不變。

This allows us to predict that only the first two cars will collide with the camera plane.
這使我們能夠預測只有前兩輛車會與相機平面發生碰撞。

We propose to perform this comparison after scaling the source image by a factor i (corresponding to TTC value τi).
我們建議在按因子 i（對應於 TTC 值 τ i）縮放源圖像後執行此比較。

The orange car is still larger in Iα i1 (α < α i), indicating that it will cross the camera plane before the specified τ i.
橙色汽車在 Iα i1 中仍然較大（α < α i），表明它將在指定的 τ i 之前穿過相機平面。

On the other hand, the blue and purple cars will not.
另一方面，藍色和紫色的汽車不會。

Rather than regressing the exact scale factor, we classify a pixel as making contact before or after τ i by classifying if the objects scale up or down in Iα i1 with respect to I0.
我們不是回歸精確的比例因子，而是通過分類對像在 Iα i1 中是否相對於 I0 放大或縮小，將像素分類為在 τ i 之前或之後進行接觸。

This yields a binary TTC probability map for τ i, (c).
這產生了 τ i, (c) 的二進制 TTC 概率圖。

3. Method 方法

Despite some time-to-contact (TTC) estimation methods dating back to the 1990s [38, 5], TTC never rose to the popularity of other techniques that are now mainstream, such as optical flow or stereo estimation.
儘管一些時間接觸 (TTC) 估計方法可以追溯到 1990 年代 [38, 5]，但 TTC 從未像現在主流的其他技術一樣流行，例如光流或立體估計。

This is due in part to its intrinsic limitations and to the challenges it poses, which we discuss below.
這部分是由於它的內在局限性和它帶來的挑戰，我們將在下面討論。

Note that in the rest of the paper we often attribute the properties of objects to the corresponding pixels, to simplify the discussion.
請注意，在本文的其餘部分，我們經常將對象的屬性歸因於相應的像素，以簡化討論。

For instance, we talk of “a pixel’s velocity” to indicate the projection on the image plane of the velocity of the object imaged by that pixel.
例如，我們談論“像素的速度”來表示該像素成像的物體的速度在圖像平面上的投影。

3.1. A Review on TimetoContact 聯繫時間回顧

Consider two frames of a static scene captured by a moving camera.
考慮由移動相機捕獲的靜態場景的兩幀。

Pixel-level correspondences allow us to compute depth, if the camera information is known.
如果已知相機信息，像素級對應允許我們計算深度。

In the more realistic case of dynamic scenes, however, the problem becomes ill-posed: the displacement of a pixel is the result of its depth, its velocity, and the camera velocity, all of which cannot be disambiguated without strong priors.
然而，在動態場景的更現實情況下，問題變得不適定：像素的位移是其深度、速度和相機速度的結果，所有這些都無法在沒有強先驗的情況下消除歧義。

In this case, the concept of time-to-contact (TTC) comes to the rescue.
在這種情況下，聯繫時間 (TTC) 的概念就派上用場了。

Given an object O, the TTC τ, i.e., the time at which object O will (or did) cross the camera plane, can be written as
給定一個物體 O，TTC τ，即物體 O 將（或確實）穿過相機平面的時間，可以寫為

$$\tau = -Z_O \Big/ \frac{dZ_O}{dt} = -Z_O / \dot{Z}_O, \qquad (1)$$

where the origin is at the camera, and we assume that the current velocity conditions will continue.
原點在相機處，我們假設當前的速度條件將繼續。

ZO is the depth of the object from the camera plane, and Z.O its relative velocity.
ZO 是物體離相機平面的深度，Z.O 是它的相對速度。

Equation 1 shows the first appealing feature of TTC: even if the depth and the velocity of the object cannot be estimated independently, can be computed from their ratio.
等式 1 顯示了 TTC 的第一個吸引人的特徵：即使物體的深度和速度無法獨立估計，也可以根據它們的比率計算。

However, we are interested in computing the TTC from pixel displacements alone.
然而，我們對僅從像素位移計算 TTC 感興趣。

To do that, we need an additional piece of information: the location of the focus-ofexpansion (FOE).
為此，我們需要一條額外的信息：擴展焦點 (FOR) 的位置。

Given two frames of a static scene captured by translating the camera, all the pixels in the image move along lines originating from the FOE, the image of the point towards which the camera is moving.
給定通過平移相機捕獲的靜態場景的兩幀，圖像中的所有像素沿著源自 FOE 的線移動，FOE 是相機向

其移動的點的圖像。

Under the same assumptions, the FOE coincides with the epipole.
在相同的假設下，FOE 與對極重合。

The relationship between the TTC, τ, and the velocity of the pixel is given by x. and y. where (x0, y0) is the FOE.
TTC、τ 和像素速度之間的關係由 x. 和 y. 給出，其中 (x0, y0) 是 FOE。

$$\dot{x} = \frac{x - x_0}{\tau} \quad \text{and} \quad \dot{y} = \frac{y - y_0}{\tau}, \qquad (2)$$

Equation 2 can be easily derived by differentiating the projection of a 3D point onto the image plane with respect to time [15].
通過將 3D 點在圖像平面上的投影相對於時間進行微分，可以很容易地推導出公式 2 [15]。

Note that the velocity (x. ,y.) can be computed from the optical flow (u, v), which allows us to write τ =((x − x0)/u).T and τ = ((y-y0)/v).T, where T is the time elapsed between the two frames.
請注意，速度 (x. ,y.) 可以從光流 (u, v) 計算出來，這允許我們寫出 τ=((x − x0)/u).T 和 τ= ((y-y0)/v).T，其中 T 是兩幀之間經過的時間。

$$\tau = \frac{x - x_0}{u} \cdot T \quad \text{and} \quad \tau = \frac{y - y_0}{v} \cdot T, \qquad (3)$$

Equation 3 shows a second compelling reason to use TTC: there is no need for camera calibration.2
公式 3 顯示了使用 TTC 的第二個令人信服的理由：無需進行相機校準。 2

There are also challenges, however.
然而，也存在挑戰。

If a rigid object is dynamic and translates with respect to the scene, its pixels move along lines centered around a different FOE.
如果剛性對像是動態的並且相對於場景平移，則其像素沿以不同 FOE 為中心的線移動。

To estimate the TTC using Equation 3, then, we need to localize the FOE of each dynamic object.
為了使用等式 3 估計 TTC，我們需要定位每個動態對象的 FOE。

Moreover, if an object is deformable, the FOE varies with each pixel, and for general motion, we need to further compensate for rotations.

此外，如果物體是可變形的，FOE 會隨著每個像素的變化而變化，對於一般運動，我們需要進一步補償旋轉。

This is why traditional approaches had to rely on oversimplifying assumptions.
這就是為什麼傳統方法不得不依賴過於簡單化的假設。

However, we can compute the size of the image of a fronto-parallel, planar, non-deformable object of size SO at distance ZO as [15] sO, where f is the focal length of the camera.
然而，我們可以將距離 ZO 處大小為 SO 的正面平行、平面、不可變形物體的圖像大小計算為 [15] sO，其中 f 是相機的焦距。

$$s_O = f S_O / Z_O, \qquad (4)$$

Since f and SO are constant, differentiating Equation 4 yields ZO/Z.O, which, plugged into Equation 1, allows us to estimate the TTC using only information about the size of an object's image and its rate of change: τ.
由於 f 和 SO 是常數，微分方程 4 產生 ZO/Z.O，將其插入方程 1，允許我們僅使用有關物體圖像大小及其變化率的資訊來估計 TTC：τ。

$$Z_O / \dot{Z}_O = -s_O / \dot{s}_O, \qquad (5)$$

$$\tau = s_O / \dot{s}_O. \qquad (6)$$

####
2
We assume a pin-hole camera model.
我們假設一個針孔相機模型。

Note that both Equation 3 and 6 effectively look at scaling.
請注意，公式 3 和 6 都有效地查看了縮放。

However, the latter is independent of the point with respect to which the scaling is performed, whereas for the former, that point is the FOE.
然而，後者與執行縮放的點無關，而對於前者，該點是 FOE。

Of course, under more realistic conditions (e.g., not planar or not fronto-parallel objects), Equation 6 becomes an approximation, which requires proper handling, as we show later.
當然，在更現實的條件下（例如，不是平面或不是正面平行的物體），方程 6 變成了一個近似值，這

需要適當的處理，我們將在後面展示。

3.2. TTC via Multiple Binary Classifications 通過多個二元分類的 TTC

In this section, we first provide the intuition motivating our method.
在本節中，我們首先提供激發我們方法的直覺。

We then describe binary TTC, the core of our method, which detects pixels predicted to collide with the camera plane within a given time.
然後我們描述了二進制 TTC，這是我們方法的核心，它檢測預測在給定時間內與相機平面碰撞的像素。

We further show how an arbitrary number of binary classifications can be combined to estimate, for each pixel, a quantized version of the TTC.
我們進一步展示瞭如何組合任意數量的二元分類來為每個像素估計 TTC 的量化版本。

To simplify the description, here we assume positive TTCs, i.e., we focus on objects moving towards the camera rather than away from it.
為了簡化描述，這裡我們假設正 TTC，即我們專注於向相機移動而不是遠離它的物體。

In Section 4, we describe a small adaptation of our method that allows us to seamlessly remove this distinction.
在第 4 節中，我們描述了我們方法的一個小改動，它使我們能夠無縫地消除這種區別。

3.2.1 Intuition 直覺

Given two images captured at times t0 and t1, Equation 1 can be approximated as
給定在時間 t0 和 t1 捕獲的兩個圖像，等式 1 可以近似為

$$\tau = -Z(t_0) \Big/ \left(\frac{Z(t_1) - Z(t_0)}{t_1 - t_0} \right) = \frac{t_1 - t_0}{1 - \frac{Z(t_1)}{Z(t_0)}}. \qquad (7)$$

If we assume fronto-parallel and planar objects that do not rotate, we can combine Equations 7 and 4, thus expressing the TTC as a function of observations in image space:
如果我們假設不旋轉的正面平行和平面物體，我們可以結合等式 7 和 4，從而將 TTC 表示為圖像空間中觀察的函數：

$$\tau = \frac{t_1 - t_0}{1 - \frac{s(t_0)}{s(t_1)}} = \frac{t_1 - t_0}{1 - \alpha}, \qquad (8)$$

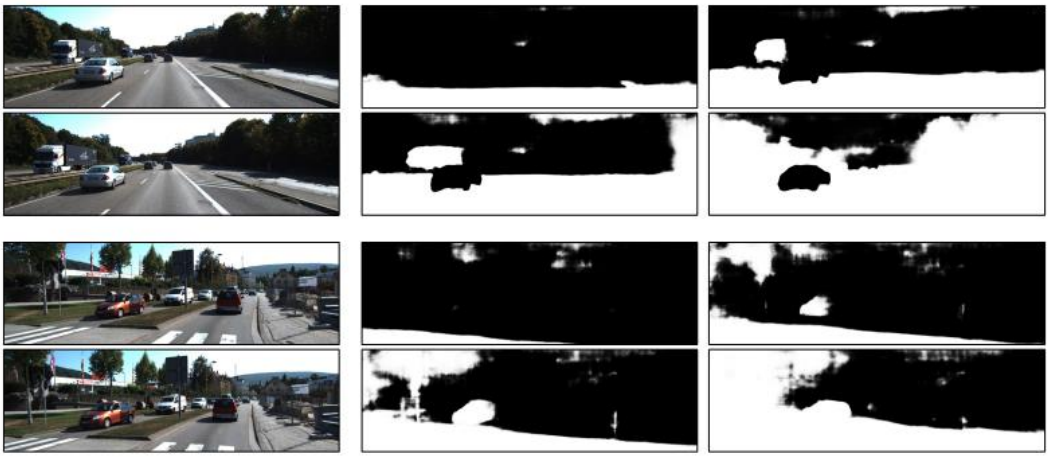
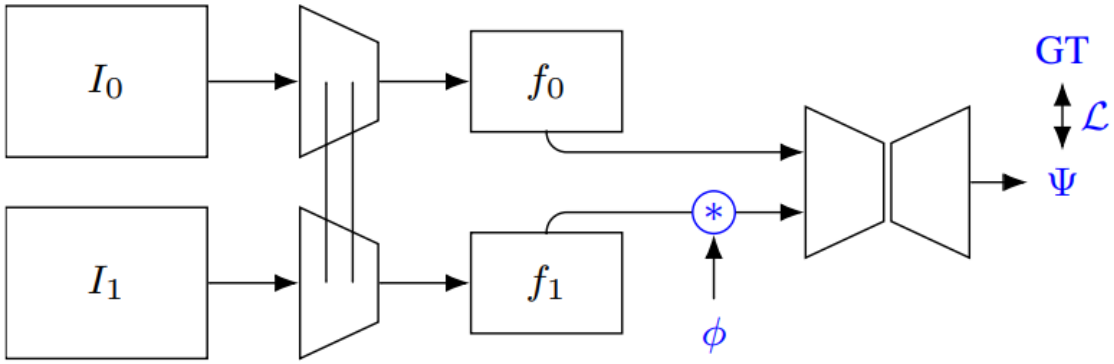


Figure 3: **Binary TTC maps.** Our method can directly identify the pixels that will contact the camera plane before a given time. From top to bottom, we show results for four TTC values for a case of highway driving and for a camera that rotates.

Figure 3: Binary TTC maps.
圖 3：二進制 TTC 映射。

Our method can directly identify the pixels that will contact the camera plane before a given time.
我們的方法可以直接識別在給定時間之前將接觸相機平面的像素。

From top to bottom, we show results for four TTC values for a case of highway driving and for a camera that rotates.
從上到下，我們展示了高速公路駕駛和旋轉相機的四個 TTC 值的結果。



Task		$(*)$	ϕ	Ψ	$\mathcal{L}$	GT
TTC	binary	scaling	α_i	B_{τ_i}	BCE	$\mathbb{1}_{\tau_* > \tau_i}$
	quantized	scaling	$\{\alpha_i\}$	$\{B_{\tau_i}\}$	BCE	$\{\mathbb{1}_{\tau_* > \tau_i}\}$
	continuous	scaling	$\{\alpha_i\}$	$\{B_{\tau_i}\}, \tau$	BCE, SL1	$\{\mathbb{1}_{\tau_* > \tau_i}\}, \tau_*$
Optical flow	horizontal	shifting	u_i	B_{u_i}	BCE	$\mathbb{1}_{u_* > u_i}$
	vertical	shifting	v_i	B_{v_i}	BCE	$\mathbb{1}_{v_* > v_i}$

Figure 4: **Architecture.** We perform all of our tasks with minor modifications to the same backbone. We preprocess the input images by extracting features and applying a task dependent operation, $(*)$, to the features of the second image. The input parameter ϕ, the loss, $\mathcal{L}$, the ground truth, GT, and the output, Ψ for each task are listed in the table. Note that $\{\cdot\}$ indicates a set, and $\mathbb{1}$ the indicator function.

Figure 4: Architecture.
圖 4：架構。

We perform all of our tasks with minor modifications to the same backbone.
我們通過對同一主幹進行微小修改來執行所有任務。

We preprocess the input images by extracting features and applying a task dependent operation, $*$, to the features of the second image.
我們通過提取特徵並對第二張圖像的特徵應用任務相關操作 $*$ 來預處理輸入圖像。

The input parameter φ, the loss, L, the ground truth, GT, and the output, Ψ for each task are listed in the table.
表中列出了每個任務的輸入參數 φ、損失 L、地面實況 GT 和輸出 Ψ。
Note that {.} indicates a set, and 1 the indicator function.
請注意，{.} 表示一個集合，而 1 表示指標函數。

where s is the size of the image of an object or region in the scene.
其中 s 是場景中對像或區域的圖像大小。

In other words, with Equation 8, the TTC can be computed from the scale factor α.
換句話說，使用等式 8，可以根據比例因子 α 計算 TTC。

This simplifies our task, compared with having to estimate per-pixel FOEs and optical flow.
與必須估計每像素 FOE 和光流相比，這簡化了我們的任務。

However, regressing α for each pixel explicitly, which requires defining the size of objects or image regions and tracking its change over time, is not straightforward.
然而，明確地為每個像素回歸 α，這需要定義對像或圖像區域的大小並跟踪其隨時間的變化，並不簡單。

Instead, we consider a pair of images, I0 and I1, and compute Iαi1 , a version of I1 scaled by a factor αi.
相反，我們考慮一對圖像，I0 和 I1，併計算 Iαi1，一個由因子 αi 縮放的 I1 版本。

This scaling factor αi corresponds to a unique TTC τi.
該比例因子αi 對應於唯一的 TTC τi。

The regions whose size matches between I0 and Iαi1 will collide with the camera plane exactly at τi.
大小在 I0 和 Iαi1 之間匹配的區域將恰好在 τi 處與相機平面碰撞。

Furthermore, we can expect regions that are larger (or smaller) in Iiα1 to collide with the camera before (or after) τi, see Figure 2.
此外，我們可以預期 Iiα1 中較大（或較小）的區域會在 τi 之前（或之後）與相機發生碰撞，見圖 2。

Instead of regressing the scale factor α directly, then, we propose to train a neural network to take such pairs of images and classify regions in Iαi1 as being larger or smaller than the corresponding regions in I0, where the correspondence is learned implicitly.
然後，我們建議訓練一個神經網絡，而不是直接回歸比例因子 α，以獲取這樣的圖像對，並將 Iαi1 中的區域分類為大於或小於 I0 中的相應區域，其中隱式學習了對應關係。

Our approach is inspired by the work of Badki et al. [1] for stereo.
我們的方法受到 Badki et al. [1]立體聲工作的啟發。

They also learn to classify the disparity of a pixel as being larger or smaller than a given disparity, instead of regressing the disparity directly.
他們還學習將像素的視差分類為大於或小於給定視差，而不是直接回歸視差。

3.2.2 Binary TTC

The inputs to our method are two images and scale factor αi corresponding to the TTC we want to analyze.
我們方法的輸入是兩個圖像和對應於我們要分析的 TTC 的比例因子 αi。

We first extract features f0 and f1 from both images, and scale f1 by αi, to obtain fαi1 .
我們首先從兩個圖像中提取特徵 f0 和 f1，並將 f1 縮放 αi，以獲得 fαi1。

Rather than classifying the objects as scaling up or down between f0 and fαi1, we train a lightweight network to directly classify whether each pixel's TTC is larger or smaller than τi, using a binary cross-entropy loss.
我們沒有將對象分類為 f0 和 fαi1 之間的放大或縮小，而是訓練一個輕量級網絡，使用二進制交叉熵損失直接分類每個像素的 TTC 是大於還是小於 τi。

That is, the network predicts a probability map Bτi which can be binarized by simple thresholding.
也就是說，網絡預測了一個概率圖 Bτi，它可以通過簡單的閾值化進行二值化。

We train directly to predict binary TTC instead of explicitly detecting if the size of the image of objects is getting larger or smaller with respect to αi for two reasons.
我們直接訓練以預測二值 TTC，而不是明確檢測物體圖像的大小相對於 αi 是變大還是變小，原因有兩個。

First, ground truth data for the scale factor is challenging to even define beyond a small neighborhood of pixels, unless objects move rigidly and only translate.
首先，比例因子的真實數據甚至很難定義一個小的像素鄰域，除非對象嚴格移動並且只能平移。

Moreover, as discussed before, Equation 8 is an approximation in the common case of objects that are not planar, and for non-rectilinear motions.
此外，如前所述，等式 8 是非平面物體和非直線運動的常見情況的近似值。

Equation 7, which establishes the relationship between the change in depth and the TTC, allows us to generate the ground truth data from existing datasets, as we explain in Section 4.1.
公式 7 建立了深度變化與 TTC 之間的關係，使我們能夠從現有數據集生成地面實況數據，如我們在第 4.1 節中所述。

A network trained to classify the TTC directly can learn the necessary priors to compensate for the approximations introduced by Equation 8.
經過訓練以直接對 TTC 進行分類的網絡可以學習必要的先驗以補償等式 8 引入的近似值。

Our core architecture is shown in Figure 4.
我們的核心架構如圖 4 所示

This very architecture is also used for all the tasks we describe below, with the caveat that the terms in blue differ for each task.

這種架構也用於我們下面描述的所有任務，但需要注意的是，藍色術語因每個任務而異。

3.2.3 Quantized TTC - 量化的 TTC

Our core approach naturally extends to estimating a coarsely quantized TTC for each pixel, which may be more useful than binary in certain scenarios.
我們的核心方法自然擴展到估計每個像素的粗量化 TTC，這在某些情況下可能比二進制更有用。

We note that Equation 9 is a complementary cumulative distribution function.
我們注意到方程 9 是一個互補的累積分佈函數。

Therefore, for two time-to-contact values, τj > τi, we can compute p(τi <τ(x, y)≤τj).
因此，對於兩個接觸時間值， τ j > τi，我們可以計算 p(τi <τ(x, y)≤τj)。

$$p(\tau_i < \tau(x,y) \leq \tau_j) = B_{\tau_i}(x,y) - B_{\tau_j}(x,y). \qquad (10)$$

Consider a set of TTC values {τi}i=1:N, and assume they are in increasing order.
考慮一組 TTC 值 {τ i}i=1:N，並假設它們按遞增順序排列。

After computing Equation 9 for each of the N TTC values, we can estimate the quantization bin in which the TTC of a pixel falls as Q(x, y).
在為 N 個 TTC 值中的每一個計算公式 9 之後，我們可以將像素的 TTC 落在其中的量化區間估計為 Q(x, y)。

$$Q(x,y) = \arg\max_i \left(p\left(\tau_i < \tau(x,y) \leq \tau_{i+1}\right) \right). \qquad (11)$$

The different TTC values can be spaced non-uniformly.
不同的 TTC 值可以不均勻地間隔開。

3.2.4 Continuous and Selective TTC - 連續和選擇性 TTC

While our method is specifically designed for binary and quantized TTC estimation, it can also estimate per-pixel, continuous TTC.
雖然我們的方法是專門為二進制和量化 TTC 估計而設計的，但它也可以估計每像素、連續的 TTC。

In principle, we can approximate continuous values by predicting quantized TCC (Section 3.2.3) with a larger set of TTC values {τi}i=1:N.
原則上，我們可以通過使用更大的一組 TTC 值 {τ i}i=1:N 預測量化的 TCC（第 3.2.3 節）來近似連續

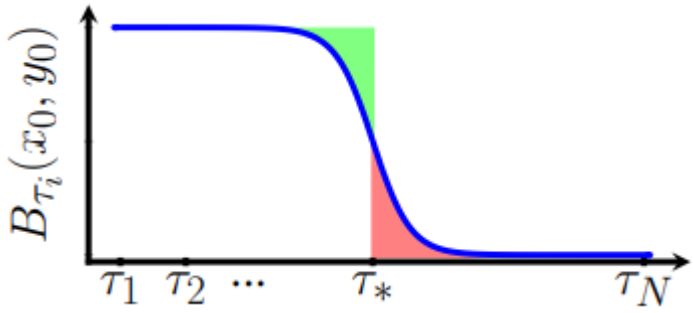
值。

This, however, fails to exploit the relationship between the binary classifications for different τi's, for a pixel.
然而，這未能利用像素的不同 τ i 的二元分類之間的關係。

We slightly modify the approach while still preserving its binary classification core, as shown in Figure 4.
我們稍微修改了該方法，同時仍保留其二元分類核心，如圖 4 所示。

Specifically, instead of taking consecutive pairs of probability maps and applying Equation 10, we stack them.
具體來說，我們不是採用連續的概率圖對並應用公式 10，而是將它們堆疊起來。

For a specific pixel (x0, y0) we generally see a progression as in the plot on the right.
對於特定像素 (x0, y0)，我們通常會看到如右圖所示的進展。



That is, Bτi (x0, y0) (the probability that the object will collide after τi) is consistently high for τi << τ* , and low for τi >> τ* , where τ* is the correct TTC value.
也就是說，對於 τ i << τ * 而言，B τ i (x0, y0)（在 τ i 之後物體碰撞的概率）始終很高，而對於 τ i >> τ * 而言則很低，其中 τ * 是正確的 TTC 值。

In the transition region, the network is uncertain, which is why aggregating information across multiple τi is beneficial.
在過渡區域，網絡是不確定的，這就是為什麼跨多個 τ i 聚合資訊是有益的。

Badki et al. [1], who obtain a similar curve for disparity, propose to estimate the transition point, in our case, by computing the area under the curve (AUC), AUC(x, y).
Badki et al. [1] 獲得了類似的視差曲線，建議通過計算曲線下面積 (AUC) AUC(x, y) 來估計過渡點，在我們的例子中。

$$\mathrm{AUC}(x,y) = \sum_i (\tau_{i+1} - \tau_i) \cdot B_{\tau_i}(x,y). \qquad (12)$$

To understand why Equation 12 yields the desired result, consider the case of a step function, i.e., aligns with a

quantization boundary: AUC= τ^*. 1 = τ^*.
要理解為什麼公式 12 會產生所需的結果，請考慮階躍函數的情況，即與量化邊界對齊：AUC= τ^*。1 = τ^*。

This relationship holds for the more common case of a smooth transition region: because the transition is generally symmetric around τ^*, the red and green areas in the plot have similar extent and compensate for each other.
這種關係適用於更常見的平滑過渡區域情況：因為過渡通常圍繞 τ^* 對稱，圖中的紅色和綠色區域具有相似的範圍並相互補償。

Because the AUC is differentiable, we can use it to fine-tune our network so that combining a set of probability values using the AUC operation yields continuous TTC values.
因為 AUC 是可微的，我們可以用它來微調我們的網絡，以便使用 AUC 操作組合一組概率值產生連續的 TTC 值。

3.2.5 A Note on Inference-Time Tradeoff - 關於推理時間權衡的說明

Binary, quantized, and continuous TTC estimation yield increasingly rich information for navigating an environment.
二進制、量化和連續 TTC 估計為導航環境提供了越來越豐富的資訊。

We note that, when estimating quantized and continuous TTC, the multiple binary classifications can be run in parallel, as they are independent of each other.
我們注意到，在估計量化和連續 TTC 時，多個二元分類可以並行運行，因為它們彼此獨立。

Therefore we can compute quantized and continuous TTC at roughly the same frame-rate as for binary quantization—just above 150 fps.
因此，我們可以以與二進制量化大致相同的幀速率（略高於 150 fps）計算量化和連續 TTC。

If multiple binary classifications cannot be run in parallel due to hardware limitations, the cost for computing quantized and continuous TTC grows linearly with the number of levels.
如果由於硬體限制而無法並行運行多個二元分類，則計算量化和連續 TTC 的成本隨級別數線性增長。

In such cases, one can decide the number of quantization levels dynamically to best leverage the trade-off between accuracy and latency.
在這種情況下，可以動態決定量化級別的數量，以最好地利用準確性和延遲之間的權衡。

For situations where a fast response time is critical, such as highway driving, binary TTC may be sufficient.
對於快速響應時間至關重要的情況，例如高速公路駕駛，二進制 TTC 可能就足夠了。

For slower navigation (e.g., for robotics or for parking lot driving), our method can be adapted to trade latency

for a finer quantization at inference time.
對於較慢的導航（例如，對於機器人或停車場駕駛），我們的方法可以適用於在推理時用延遲換取更精細的量化。

3.3. Dealing with the Lack of Training Data 處理缺乏訓練數據

As for any learning-based method, having access to a large amount of data is critical to properly train our network.
對於任何基於學習的方法，訪問大量數據對於正確訓練我們的網絡至關重要。

Unfortunately, there are no datasets with TTC ground truth data and few scene flow datasets that we can use to infer it (Section 4.1).
不幸的是，沒有包含 TTC 地面實況數據的數據集，也沒有我們可以用來推斷它的場景流數據集（第 4.1 節）。

To increase the training data, we leverage a closely related task: binary optical flow estimation.
為了增加訓練數據，我們利用了一個密切相關的任務：二元光流估計。

We use the same binary approach, but we shift the features (horizontally and vertically) rather than scaling them.
我們使用相同的二進制方法，但我們移動特徵（水平和垂直）而不是縮放它們。

We then classify the direction of the shift (left/right or up/down).
然後我們對移位的方向（左/右或上/下）進行分類。

For horizontal shifts, we seek to predict a probability map relative to a given shift ui Bui (x, y), where fui1 are the features extracted from the second image and shifted by ui.
對於水平位移，我們試圖預測一個相對於給定位移 ui Bui (x, y) 的概率圖，其中 fui1 是從第二幅圖像中提取的特徵並被 ui 位移。

$$B_{u_i}(x,y) = p(u(x,y) > u_i; f_0, f_1^{u_i}), \qquad (13)$$

The equation for the vertical shift vi is analogous.
垂直位移 vi 的等式是類似的。

Using optical flow to pre-train our network and continuing using it as an auxiliary task yields a stronger inductive bias, as we discuss in Section 5.
正如我們在第 5 節中討論的那樣，使用光流來預訓練我們的網絡並繼續將其用作輔助任務會產生更強的歸納偏差。

Although optical flow is an auxiliary task, we show results in Figure 5 to allow for a visual evaluation.
儘管光流是一項輔助任務，但我們在圖 5 中顯示了結果以進行視覺評估。

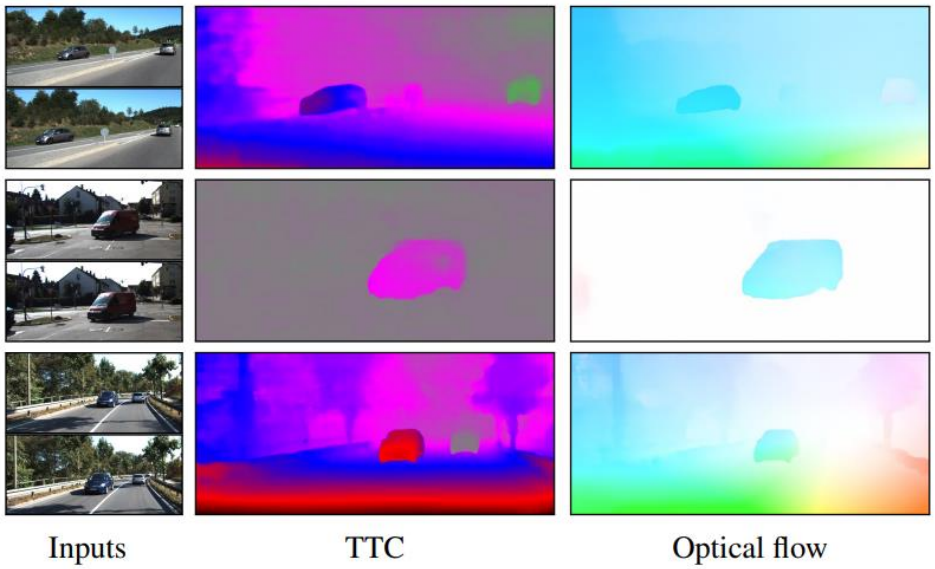


Figure 5: **Optical flow as an auxiliary task.** We estimate optical flow as an auxiliary task to improve our TTC estimation. Like for continuous TTC, we estimate optical flow via a series of binary classifications. While optical flow is not a goal for us, this figure shows that our prediction is reasonable.

Figure 5: Optical flow as an auxiliary task. 圖 5：作為輔助任務的光流。

We estimate optical flow as an auxiliary task to improve our TTC estimation. Like for continuous TTC, we estimate optical flow via a series of binary classifications.
我們將光流估計為輔助任務以改進我們的 TTC 估計。 與連續 TTC 一樣，我們通過一系列二元分類來估計光流。

While optical flow is not a goal for us, this figure shows that our prediction is reasonable.
雖然光流不是我們的目標，但這個數字表明我們的預測是合理的。

4. Implementation Details 實施細則

4.1. TTC Data Generation 數據生成

A dataset providing per-pixel TTC ground truth does not exist.
不存在提供每像素 TTC 地面實況的數據集。

However, we can use ratio of the depth in different frames to compute the TTC with Equation 7.
但是，我們可以使用不同幀中的深度比率來計算 TTC，公式 7。

This ratio can be computed from datasets offering per-pixel scene flow [X(t0), Y (t0), Z(t0)] → [X(t1), Y (t1), Z(t1)].
這個比率可以從提供每像素場景流 [X(t0), Y (t0), Z(t0)] → [X(t1), Y (t1), Z(t1)] 的數據集計算出來。

To train for TTC estimation we use the Driving dataset from the SceneFlowDatasets [29].
為了訓練 TTC 估計，我們使用來自 SceneFlowDatasets [29] 的駕駛數據集。

We also use the KITTI15 [30] dataset to train for TTC estimation.
我們還使用 KITTI15 [30] 數據集來訓練 TTC 估計。

We split the training dataset of KITTI15 into train and validation split, and show analysis on the validation part of the dataset.
我們將 KITTI15 的訓練數據集拆分為訓練和驗證拆分，並對數據集的驗證部分進行分析。

To pretrain our network for the task of binary optical flow, we use the FlyingChairs2 [7, 19] and the FlyingThings3D [29] datasets.
為了對我們的網絡進行二進制光流任務的預訓練，我們使用了 FlyingChairs2 [7, 19] 和 FlyingThings3D [29] 數據集。

We also use optical flow data available from Driving [29] and KITTI15 [30], and train our network for both binary optical flow and TTC estimation.
我們還使用來自驅動 [29] 和 KITTI15 [30] 的光流數據，並訓練我們的網絡進行二元光流和 TTC 估計。

4.2.Working in the Inverse TTC Domain . 在逆 TTC 域中工作

We are interested in objects predicted to collide within i seconds.
我們對預測在 i 秒內發生碰撞的物體感興趣。

However, τ∈ (-∞, ∞), with positive and negative values indicating objects moving towards and away from the camera, respectively.
然而，$\tau \in (-\infty, \infty)$，正值和負值分別表示物體朝向和遠離相機移動。

That is, some of the pixels whose TTC is smaller than τi will never collide with the camera plane.
也就是說，一些 TTC 小於 τ i 的像素永遠不會與相機平面發生碰撞。

In practice, then, we seek to identify the pixels such that (f(x)…(14)), which introduces an additional complication.
在實踐中，我們試圖識別像素 (f(x)…(14))，這會引入額外的複雜性。

$$(\tau(x,y) \leq \tau_i) \ \cap \ (\tau(x,y) > 0), \qquad (14)$$

Moreover, the effect of TTC in the image space in not linear.
此外，TTC 在圖像空間中的影響不是線性的。

A simple solution is to work with the ratio-of-depths, the inverse of the TTC domain:
一個簡單的解決方案是使用深度比，即 TTC 域的倒數：

$$\eta = \frac{Z(t_1)}{Z(t_0)} = 1 - \frac{t_1 - t_0}{\tau}. \qquad (15)$$

Yang and Ramanan termed it motion-in-depth [43].
Yang 和 Ramanan 將其稱為深度運動 [43]。

Thanks to Equation 15, the effect of TTC is linear in image space and Equation 14 simply reduces to
由於方程 15，TTC 的影響在圖像空間中是線性的，方程 14 簡單地簡化為

$$\eta(x,y) \leq \eta_i. \qquad (16)$$

For binary TTC, then, we simply scale the features of the source image by a factor α i = η i.
對於二值 TTC，我們只需按因子 α i = η i 縮放源圖像的特徵。

Similarly, for continuous estimation, we sample planes uniformly in the inverse TTC domain.
類似地，對於連續估計，我們在逆 TTC 域中均勻地採樣平面。

We apply uniformly spaced scale factors to the features of the source image:
我們將均勻間隔的比例因子應用於源圖像的特徵：

$$\{\alpha_i\}_{i=1:N} = \{\alpha_0 + i\Delta\alpha\}_{i=1:N}.$$

The AUC of the resulting segmentation gives us the map η (x, y), which we then covert to a TTC map.
結果分割的 AUC 為我們提供了映射 η (x, y)，然後我們將其轉換為 TTC 映射。

4.3. Architecture 架構

Figure 4 shows our architecture.
圖 4 顯示了我們的架構。

The feature extraction module uses spatial pyramid pooling layers as PSMNet [4].
特徵提取模塊使用空間金字塔池化層作為 PSMNet [4]。

The resulting 32-channel feature maps are at one-third the input image resolution.
生成的 32 通道特徵圖是輸入圖像分辨率的三分之一。

We then apply a task-dependent operator, *, parametrized by φ, to f1 and generate fφ1.
然後，我們將一個由 φ 參數化的任務相關運算符 * 應用於 f1 並生成 fφ1 。

For instance, when estimating the horizontal component of binary optical flow, * shifts the features by a horizontal shift ui, and for binary TTC, it scales them by αi.
例如，在估計二元光流的水平分量時，* 將特徵平移一個水平平移 ui，對於二元 TTC，它將它們縮放 αi。

The full list for each task is in the table in Figure 4.
每個任務的完整列表在圖 4 的表格中。

To avoid cropping out features, we zero-pad f0 and fφ1 to 1.5x the feature map resolution.
為了避免裁剪特徵，我們將 f0 和 fφ1 零填充到特徵圖分辨率的 1.5 倍。

The concatenation of f0 and fφ1 is fed to a 2D encoder-decoder network with skip connections.
f0 和 fφ1 的串聯被饋送到具有跳過連接的 2D 編碼器-解碼器網絡。

This module has three heads—one for binary TTC, one for binary horizontal optical flow, and one binary vertical optical flow.
該模塊具有三個頭，一個用於二進制 TTC，一個用於二進制水平光流，一個用於二進制垂直光流。

We apply a sigmoid to each output to obtain the respective probability maps.
我們對每個輸出應用 sigmoid 以獲得相應的概率圖。

Intuitively, this network tells us if the features in fφ1 are scaling up/down, shifting right/left, or shifting up/down with respect to the corresponding features in f0.
直觀地說，這個網絡告訴我們 fφ1 中的特徵是否相對於 f0 中的相應特徵向上/向下縮放、向右/向左移動或向上/向下移動。

4.4. Training 訓練

We pre-train our network for the binary optical flow task on the FlyingChairs2 dataset and train it further on the FlyingThings3D dataset [29] using a binary cross-entropy (BCE) loss with respect to a thresholded version of the ground truth.

我們在 FlyingChairs2 數據集上為二元光流任務預訓練我們的網絡，並在 FlyingThings3D 數據集 [29] 上使用二元交叉熵 (BCE) 損失對地面實況的閾值版本進行進一步訓練。

The TTC head is left unsupervised in this stage.
在此階段，TTC 負責人不受監督。

We then fine-tune our network for estimating binary TTC first on the Driving [29] and then on the KITTI15 [30] datasets.
然後我們微調我們的網絡，首先在驅動 [29] 和 KITTI15 [30] 數據集上估計二進制 TTC。

Since both datasets also offer optical flow data, in this second stage we train for both binary optical flow and binary TCC.
由於兩個數據集也提供光流數據，因此在第二階段我們訓練二元光流和二元 TCC。

In Section 5 we discuss the impact of this choice on the quality of the results.
在第 5 節中，我們討論了這種選擇對結果質量的影響。

We use relative weights of 0.8 and 0.2 for binary TTC and binary optical flow tasks respectively.
我們分別對二進制 TTC 和二進制光流任務使用 0.8 和 0.2 的相對權重。

For training on the KITTI15 dataset, we use the same split as Yang and Ramanan [43].
對於 KITTI15 數據集的訓練，我們使用與 Yang 和 Ramanan [43] 相同的分割。

For the continuous version, we pre-train the network as before, and fine-tune it on FlyingThings3D for continuous optical flow.
對於連續版本，我們像以前一樣預訓練網絡，並在 FlyingThings3D 上對其進行微調以獲得連續光流。

For each training image pair we uniformly sample horizontal and vertical shifts and stack the maps corresponding to each shift.
對於每個訓練圖像對，我們統一採樣水平和垂直位移，並堆疊對應於每個位移的地圖。

We use the resulting volumes to compute continuous optical flow via the AUC operation described in Section 3.2.4.
我們使用結果體積通過第 3.2.4 節中描述的 AUC 操作計算連續光流。

We can then continue training the network using a BCE loss on the individual probability maps, and a Smooth-L1 (SL1) [12] regression loss on the output of the AUC module.
然後，我們可以繼續使用單個概率圖上的 BCE 損失和 AUC 模塊輸出上的 Smooth-L1 (SL1) [12] 回歸損失來訓練網絡。

We use relative weights of 0.1 and 0.9 respectively.

我們分別使用 0.1 和 0.9 的相對權重。

To fine-tune our network for continuous TTC estimation, we follow a similar strategy as for the binary segmentation training.
為了微調我們的網絡以進行連續 TTC 估計，我們遵循與二元分割訓練類似的策略。

We continue training the network for the task of continuous optical flow and continuous TTC estimation on the Driving and the KITTI15 datasets.
我們繼續訓練網絡，以完成在駕駛和 KITTI15 數據集上進行連續光流和連續 TTC 估計的任務。

However, this time we form three volumes, two corresponding to optical flow and one to TTC.
不過這次我們形成了三卷，兩卷對應光流，一卷對應 TTC。

As discussed in Section 4.2, we work in the inverse TTC domain and uniformly sample scale factors.
如第 4.2 節所述，我們在逆 TTC 域中工作並統一採樣比例因子。

The network trained on the entire KITTI15 dataset is used for scene flow estimation on KITTI15 benchmark images, as explained in Section 5.
在整個 KITTI15 數據集上訓練的網絡用於對 KITTI15 基準圖像進行場景流估計，如第 5 節所述。

We refer the reader to our Supplementary for the additional training details.
我們將讀者推薦給我們的補充以獲取額外的培訓細節。

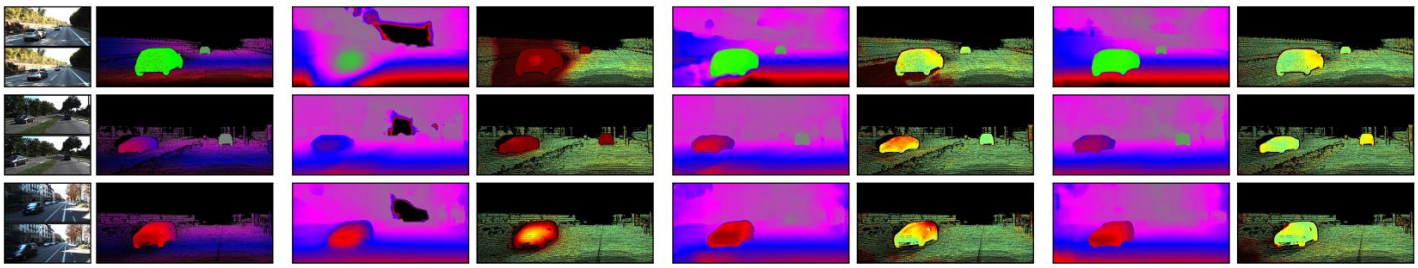


Figure 6: **Comparison on continuous TTC estimation.** We show the predicted TTC and the error map (red and green indicate high and low error, respectively) for all methods. Our method and Yang and Ramanan's method explicitly train for TTC estimation and provide better results than the monocular scene flow method of Hur and Roth. Note the lower error in the TTC estimation for our method.

Figure 6: Comparison on continuous TTC estimation.
圖 6：連續 TTC 估計的比較。

We show the predicted TTC and the error map (red and green indicate high and low error, respectively) for all methods.
我們展示了所有方法的預測 TTC 和誤差圖（紅色和綠色分別表示高低誤差）。

Our method and Yang and Ramanan's method explicitly train for TTC estimation and provide better results

than the monocular scene flow method of Hur and Roth.
我們的方法以及 Yang 和 Ramanan 的方法明確訓練了 TTC 估計，並提供了比 Hur 和 Roth 的單目場景流方法更好的結果。

Note the lower error in the TTC estimation for our method.
請注意我們方法的 TTC 估計誤差較低。

5. Evaluation and Results 評估和結果

In this section we discuss qualitative and quantitative results for our method.
在本節中，我們將討論我們方法的定性和定量結果。 .

First, to validate the importance of the training strategy described in Section 3.3, we compare three training strategies:
首先，為了驗證 3.3 節中描述的訓練策略的重要性，我們比較了三種訓練策略：

(1) we train our network only to estimate binary TTC,
(1) 我們訓練我們的網絡只是為了估計二進制 TTC，

(2) we pre-train it to estimate binary optical flow (OF) and then binary TTC, and
(2) 我們預訓練它來估計二進制光流 (OF)，然後是二進制 TTC，和

(3) we pre-train with binary OF, and then continue training with both binary OF and binary TTC (our method).
(3) 我們使用二進制 OF 進行預訓練，然後繼續使用二進制 OF 和二進制 TTC（我們的方法）進行訓練。

As shown in Table 1, our final method achieves a percentage error of 1.013 and a mean intersection-over-union (mIOU) of 0.9525.
如表 1 所示，我們的最終方法實現了 1.013 的百分比誤差和 0.9525 的平均交集交叉（mIOU）。

If we only train for TTC estimation after pre-training for OF, we observe an increase in percentage error to 1.174 and a drop in mIOU to 0.9453.
如果我們在對 OF 進行預訓練後只訓練 TTC 估計，我們會觀察到百分比誤差增加到 1.174，而 mIOU 下降到 0.9453。

Removing the binary OF estimation altogether,leads to an additional increase in percentage error to 2.670 and an additional drop in mIOU to 0.8808.
完全去除二元 OF 估計會導致百分比誤差額外增加至 2.670，而 mIOU 額外下降至 0.8808。

We also thoroughly validate our approach numerically on the KITTI15 validation set.
我們還在 KITTI15 驗證集上徹底驗證了我們的方法。

We compare to Yang and Ramanan [43], who proposed the only existing approach that computes per-pixel TTC from a monocular camera, under practical assumptions.
我們與 Yang 和 Ramanan [43] 進行了比較，他們提出了在實際假設下從單目相機計算每像素 TTC 的唯一現有方法。

They too look at how the size of objects changes, but they estimate it explicitly (and locally) from the optical flow between frames.
他們也觀察物體的大小是如何變化的，但他們從幀之間的光流中明確地（和局部地）估計它。

We also measure against PRSM [41] and OSF [30], both of which perform 3D scene flow estimation using stereo cameras.
我們還針對 PRSM [41] 和 OSF [30] 進行了測量，它們都使用立體相機執行 3D 場景流估計。

They represent images as a collection of planar super-pixels and jointly solve for geometry and 3D motion, which informs about the change of depth over time (Section 4.1).
它們將圖像表示為平面超像素的集合，並聯合求解幾何和 3D 運動，從而了解深度隨時間的變化（第 4.1 節）。

This can be used to estimate motion-in-depth for each pixel, η(x, y), using Equation 15, which can then be thresholded.
這可用於估計每個像素的深度運動，η (x, y)，使用等式 15，然後可以對其進行閾值處理。

Our final comparison is against Hur and Roth [18], a state-of-the-art monocular scene flow estimation approach.
我們最後的比較是針對 Hur 和 Roth [18]，這是一種最先進的單目場景流估計方法。

We compare against their best model, which uses a combination of selfsupervised learning on the KITTI15 raw dataset and supervised learning on the entire KITTI15 training dataset.
我們與他們最好的模型進行了比較，該模型結合了 KITTI 15 原始數據集上的自監督學習和整個 KITTI15 訓練數據集上的監督學習。

Note that this approach is already fine-tuned on our validation set.
請注意，這種方法已經在我們的驗證集上進行了微調。

Table 1 shows that our continuous TTC estimation, yields the lowest motion-in-depth error:
表 1 顯示我們的連續 TTC 估計產生最低的深度運動誤差：

$$\text{MiD} = ||\log(\eta) - \log(\eta_{GT})||_1 \cdot 10^4. \qquad (17)$$

However, our fundamental innovation is the ability to define a temporal geofence, i.e., detecting pixels with a TTC smaller than a given τi without the need to estimate the full TTC first.
然而，我們的基本創新是定義時間地理圍欄的能力，即檢測具有小於給定 τ i 的 TTC 的像素，而無需首先估計完整的 TTC。

On this task we perform on par with Yang and Ramanan, but our binary TTC is around 26 faster.
在這個任務上，我們的表現與 Yang 和 Ramanan 相當，但我們的二進制 TTC 快了大約 26。

OSF performs better than our approach in terms of binary TTC, though, again, it uses a richer input.
OSF 在二進制 TTC 方面比我們的方法表現更好，但同樣，它使用更豐富的輸入。

Moreover, OSF and PRSM, implemented on a CPU, take 390 s and 300 s respectively.
此外，在 CPU 上實現的 OSF 和 PRSM 分別需要 390 秒和 300 秒。

The approach by Hur and Roth struggles despite fine-tuning the model on the validation set.
儘管在驗證集上對模型進行了微調，但 Hur 和 Roth 的方法仍然很掙扎。

However, unlike ours and Yang and Ramanan's approaches, they do not use a synthetic dataset to pre-train, nor train for a betterposed TTC estimation task.
然而，與我們以及 Yang 和 Ramanan 的方法不同的是，他們不使用合成數據集進行預訓練，也不使用更好的 TTC 估計任務進行訓練。

Instead, they focus on the more difficult task of monocular scene flow estimation.
相反，他們專注於更困難的單目場景流估計任務。

We further evaluate our method on the KITTI15 benchmark [31] with the same strategy as Yang and Ramanan [43]:
我們使用與 Yang 和 Ramanan [43] 相同的策略，在 KITTI15 基準測試 [31] 上進一步評估我們的方法：

we use our motion-in-depth estimation to compute scene flow, the 3D motion [X(t0), Y (t0), Z(t0)] → [X(t1), Y (t1),Z(t1)] corresponding to each pixel.
我們使用深度運動估計來計算場景流，即 3D 運動 [X(t0), Y (t0), Z(t0)] → [X(t1), Y (t1),Z(t1)] 對應每個像素。

The first two scene flow components can be estimated by back-projecting the optical flow for each pixel.
前兩個場景流分量可以通過對每個像素的光流進行反投影來估計。

For the Z component, traditional approaches use stereo information.
對於 Z 分量，傳統方法使用立體資訊。

We can estimate the third component from the motion-in-depth ratio if we are given the depth in one frame.
如果給定一幀中的深度，我們可以從深度運動比率中估計第三個分量。

After computing the depth at t0 with GANet [46] (evaluated by D1 in Table 2) and combining it with our TTC estimate, we can compute the depth for each pixel at time t1.
在使用 GANet [46]（由表 2 中的 D1 評估）計算 t0 處的深度並將其與我們的 TTC 估計值結合後，我們可以計算 t1 時刻每個像素的深度。

Therefore, D2 in Table 2 effectively measures the quality of our results.
因此，表 2 中的 D2 有效地衡量了我們結果的質量。

The other numbers in the table evaluate other components and are reported for completeness.
表中的其他數字評估其他組件並報告完整性。

While our method is not designed to estimate scene flow, it performs on par, or slightly better than methods specifically optimized for it.
雖然我們的方法不是為了估計場景流而設計的，但它的性能與專門針對它優化的方法相當，或者略好一些。

In certain scenarios, binary TTC can be more informative than depth.
在某些情況下，二進制 TTC 可以提供比深度更多的資訊。

For instance, the top row of Figure 1 shows a car driving away from the camera, and one that is farther, but driving towards the camera.
例如，圖 1 的最上面一行顯示了一輛遠離攝像頭的汽車，還有一輛更遠但駛向攝像頭的汽車。

The latter, detected by our binary TTC, is potentially more impactful—in the true sense of the word!—for the ego vehicle path planning.
後者由我們的二元 TTC 檢測到，對於自我車輛路徑規劃可能更具影響力——在這個詞的真正意義上！

The last row of Figure 3 shows that, as discussed in Section 3.2.2, our method can compensate for camera rotation, even if that breaks some of our assumptions.
圖 3 的最後一行顯示，如第 3.2.2 節所述，我們的方法可以補償相機旋轉，即使這打破了我們的一些假設。

This is also visible for continuous TTC in the first row of Figure 5.
這對於圖 5 第一行中的連續 TTC 也是可見的。

We show quantized TTC estimation results in Figure 1 Q .
我們在圖 1 Q 中顯示了量化的 TTC 估計結果。

Note that even just 9 TTC quantization levels (8 binary classifications) provide a meaningful representation of the scene.
請注意，即使只有 9 個 TTC 量化級別（8 個二進制分類）也能提供有意義的場景表示。

Moreover, the underlying binary classifications can be run in parallel as they are independent of each other.
此外，底層的二進制分類可以並行運行，因為它們彼此獨立。

Therefore, quantized TTC can be run at roughly the same frame-rate as the binary TTC.
因此，量化 TTC 可以以與二進制 TTC 大致相同的幀速率運行。

We show a visual comparison with Hur and Roth [18], and Yang and Ramanan [43] on continuous TTC for KITTI15 images in Figure 6.
我們在圖 6 中的 KITTI15 圖像的連續 TTC 上展示了與 Hur 和 Roth [18] 以及 Yang 和 Ramanan [43] 的視覺比較。

Note the lower for our approach.
請注意我們的方法較低。

In Figure 7 we show qualitative result of our approach on the Citiscapes dataset [6].
在圖 7 中，我們展示了我們的方法在 Citiscapes 數據集 [6] 上的定性結果。

We show two failure cases on this dataset, one due to the sudden vertical motion caused by a road bump and another due to an object rotating significantly.
我們在這個數據集上展示了兩個失敗案例，一個是由於道路顛簸引起的突然垂直運動，另一個是由於物體顯著旋轉。

Broadly, our method struggles for motions under-represented in the training dataset.
從廣義上講，我們的方法難以解決訓練數據集中代表性不足的運動。

		Binary (200 ms – 2 s)		Continuous
		mIOU (↑)	% error (↓)	MiD (↓)
Stereo	PRSM [41]	0.9365	1.339	124.0
	OSF [30]	**0.9556**	**0.941**	115.0
Mono	Hur & Roth [18]	0.9418	1.233	115.13
	Yang & Ramanan [43]	0.9525	1.012	75.00
	Ours	0.9525	1.013	**73.55**

Table 1: Comparison on the validation set of KITTI15 for both binary TTC (averaged over a set $\{\alpha_i\}$ uniformly sampled in the interval $\tau \in [0.02s, 2s]$) and continuous TTC. Note that PRSM and OSF both use richer input data (stereo vs mono). MiD, motion-in-depth, directly evaluates TTC.

Table 1: Comparison on the validation set of KITTI15 for both binary TTC (averaged over a set fig uniformly sampled in the interval 2 [0:02s; 2s]) and continuous TTC.
表 1：KITTI15 驗證集對二進制 TTC（在間隔 2 [0:02s; 2s] 中均勻採樣的一組圖的平均值）和連續 TTC 的比較。

Note that PRSM and OSF both use richer input data (stereo vs mono).
請注意，PRSM 和 OSF 都使用更豐富的輸入數據（立體聲與單聲道）。

MiD, motion-indepth, directly evaluates TTC.
MiD，motion-indepth，直接評估 TTC。

	D1-all	D2-bg	D2-fg	D2-all	Fl-all	SF-all
UberATG-DRISF [28]	2.55	2.90	9.73	4.04	4.73	6.31
ACOSF [24]	3.58	3.82	12.74	5.31	5.79	7.90
ISF [2]	4.46	4.88	11.34	5.95	6.22	8.08
Yang&Ramanan [43]	1.81	3.39	8.54	4.25	6.30	8.12
Ours	**1.81**	**3.84**	**9.39**	**4.76**	**6.31**	**8.50**

Table 2: Top 5 published methods on the KITTI scene flow benchmark. Our method performs reasonably well, despite not being designed for scene flow, see text.

Table 2: Top 5 published methods on the KITTI scene flow benchmark.
表 2：在 KITTI 場景流基准上發布的前 5 種方法。

Our method performs reasonably well, despite not being designed for scene flow, see text.
儘管我們的方法不是為場景流而設計的，但我們的方法表現相當不錯，請參閱文本。

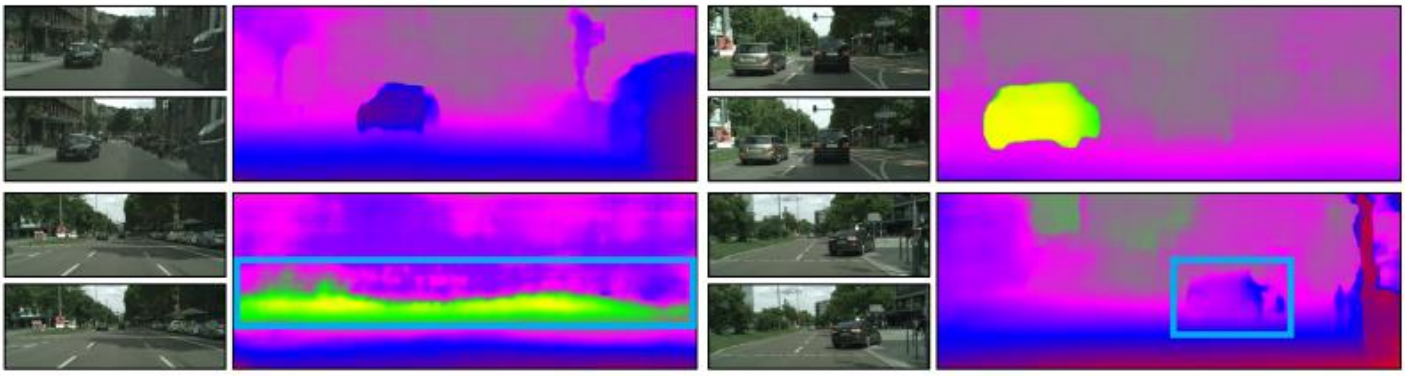


Figure 7: **Unseen dataset.** Our predicted TTC map on Citiscapes [6]. The first row shows an example of a car moving towards us and another where the car in the adjacent lane is speeding away. The bottom two rows show failures due to a road bump and a drastic rotation, respectively.

Figure 7: Unseen dataset. Our predicted TTC map on Citiscapes [6].
圖 7：看不見的數據集。 我們在 Citiscapes 上預測的 TTC 地圖 [6]。

The first row shows an example of a car moving towards us and another where the car in the adjacent lane is speeding away.
第一行顯示了一個汽車向我們駛來的例子，另一個例子是相鄰車道上的汽車正在加速駛離。

The bottom two rows show failures due to a road bump and a drastic rotation, respectively.
底部兩行分別顯示由於路面顛簸和劇烈旋轉而導致的故障。

6. Conclusions 結果

In certain scenarios, time-to-contact (TTC) information can be more useful than depth.
在某些情況下，聯繫時間 (TTC) 資訊可能比深度更有用。

However, existing TTC estimation methods either make impractical assumptions, or cannot be run in real time.
然而，現有的 TTC 估計方法要么做出不切實際的假設，要么無法實時運行。

We presented a framework to estimate time-to-contact (TTC) from a monocular input.
我們提出了一個框架來從單眼輸入估計接觸時間（TTC）。

In just 6.4 ms, our approach computes a temporal geofence to detect objects predicted to collide with the camera plane within a given TTC.
在短短 6.4 毫秒內，我們的方法計算了時間地理圍欄，以檢測在給定 TTC 內預測會與相機平面發生碰撞的物體。

By computing a number of such geofences, it can also estimate TTC with arbitrary quantization, including continuous TTC.
通過計算多個這樣的地理圍欄，它還可以估計任意量化的 TTC，包括連續 TTC。

We show that our method achieves competitive performance for TTC estimation— even when other methods use richer input data.
我們表明我們的方法在 TTC 估計方面取得了有競爭力的性能——即使其他方法使用更豐富的輸入數據。