

**Algorithm 5:** NERF(ScN, C, u)

---

```

 $\text{dir}_c \leftarrow \left[ \frac{u_x - c_x}{f_x}, \frac{u_y - c_y}{f_y}, 1 \right];$ 
 $\text{dir}_w \leftarrow \frac{\text{dir}_c \times R^\top}{\|\text{dir}_c\|};$ 
for  $\forall i \in \text{range}(N)$  do
     $x[i] \leftarrow T + \frac{(2i-1) \cdot L \cdot \text{dir}_w}{2N};$ 
     $c[i] \leftarrow F_c(x_i, \text{dir}_w);$ 
     $o[i] \leftarrow F_o(x[i]);$ 
     $a[i] \leftarrow 1 - \text{Exp}\left(-\frac{o[i] \cdot L}{N}\right);$ 
     $oc[i] \leftarrow \prod_{j=1}^{i-1} (1 - a[j]);$ 
 $\text{pc} \leftarrow \sum_{i=1}^N (oc[i] \cdot a[i] \cdot c[i]);$ 
return pc;

```

---

483 **NeRF rendering** Algorithm 5 takes as input a neural radiance field ScN, a camera C, and a pixel  
484 coordinate u, and outputs the rendered RGB value pc at that pixel. The rendering proceeds in three  
485 steps: (1) it computes the normalized direction of the camera ray  $\text{dir}_w$  corresponding to u and samples  
486 N points  $x$  along the ray within the maximum range L (Line 1-4); (2) it evaluates the opacity and  
487 color of each sampled point using the networks  $F_o$  and  $F_c$  (Line 5-6); (3) it aggregates the colors  
488 along the ray by computing a weighted sum based on the opacity and depth ordering of the samples  
489 (Line 7-9).

## 490 B Fundamental Operations Table

Fundamental operations for GAUSSIANSPLOT and NERF are shown in Table 4

Table 4: Basic Operation Table. Conditional  $A ? B : C$  returns  $B$  if  $A$  is true and otherwise  $C$ . Permuting  $x$  based on  $y$  means reordering the elements of  $x$  according to the indices that sort  $y$  in ascending order. E.g., permuting (9, 3, 7) based on (5, 13, 8) results in (9, 7, 3).

Operation	Inputs	Output	Math Representation
Element-wise add (Add)	$x, y \in \mathbb{R}^{n \times m}$	$z \in \mathbb{R}^{n \times m}$	$z_{ij} = x_{ij} + y_{ij}$
Element-wise multiply (Mul)	$x, y \in \mathbb{R}^{n \times m}$	$z \in \mathbb{R}^{n \times m}$	$z_{ij} = x_{ij} * y_{ij}$
Division (Div)	$x \in \mathbb{R}_{>0}^{n \times m}$	$z \in \mathbb{R}_{>0}^{n \times m}$	$z_{ij} = \frac{1}{x_{ij}}$
Matrix multiplication (Mmul)	$x \in \mathbb{R}^{n \times m}, y \in \mathbb{R}^{m \times k}$	$z \in \mathbb{R}^{n \times k}$	$z = x \times y$
Matrix inverse (Inv)	$x \in \mathbb{R}^{n \times n}, \det(x) \neq 0$	$z \in \mathbb{R}^{n \times n}$	$z = x^{-1}$
Matrix power (Pow)	$x \in \mathbb{R}^{n \times n}, k \in \mathbb{N}$	$z \in \mathbb{R}^{n \times n}$	$z = x \times x \times \dots \times x$
Summation (Sum)	$x \in \mathbb{R}^n, k \in \mathbb{N}$	$z \in \mathbb{R}$	$z = \sum_{i=1}^k x_i$
Product (Prod)	$x \in \mathbb{R}^n, k \in \mathbb{N}$	$z \in \mathbb{R}$	$z = \prod_{i=1}^k x_i$
Matrix transpose (T)	$x \in \mathbb{R}^{n \times m}$	$z \in \mathbb{R}^{m \times n}$	$z_{ij} = x_{ji}$
Element-wise exponential (Exp)	$x \in \mathbb{R}^{n \times m}$	$z \in \mathbb{R}^{n \times m}$	$z_{ij} = e^{x_{ij}}$
Frobenius norm (Norm)	$x \in \mathbb{R}^{n \times m}$	$z \in \mathbb{R}$	$z = \ x\ $
Element-wise indicator (Ind)	$x \in \mathbb{R}^{n \times m}$	$z \in \mathbb{R}^{n \times m}$	$z_{ij} = (x_{ij} > 0) ? 1 : 0$
Sorting (Sort)	$z \in \mathbb{R}^n$	$x \in \mathbb{R}^n, y \in \mathbb{R}^n$	permute $x$ based on $y$

491

## 492 C Supplementary Experiment Results

## 493 C.1 Detailed Scenario Information

494 Table 5 summarizes scene configurations and rendering quality metrics for the scenarios discussed in  
495 Section 5. Figure 6 depicts representative GAUSSIANSPLOT-rendered images for each scene.



Figure 6: Example rendered image for scenario (left to right, top to bottom) Lego, Pinetree, Airport, Garden, Plane, Truck, Car by GAUSSIANSPLAT

Table 5: Reconstruction Quality of scene models is evaluated by the following metrics: Number of Gauss used for scene representation (#GAUSS, applicable only to Gaussian Splatting); Peak Signal-to-Noise Ratio (PSNR); Structural Similarity Index Measure (SSIM); Learned Perceptual Image Patch Similarity (LPIPS).

	Gaussian Splatting				NeRF		
	#GAUSS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
Lego	43543	25.60	0.95	0.07	21.38	0.81	0.19
Chair	N/A	N/A	N/A	N/A	22.56	0.89	0.11
Drums	N/A	N/A	N/A	N/A	19.46	0.81	0.23
Pinetree	113368	31.06	0.97	0.06	22.41	0.79	0.20
Airport	617371	18.83	0.84	0.33	N/A	N/A	N/A
Garden	524407	18.74	0.37	0.32	N/A	N/A	N/A
Plane	51316	28.05	0.95	0.11	22.89	0.70	0.26
Truck	47895	24.70	0.94	0.09	23.53	0.75	0.19
Car	34699	26.98	0.93	0.10	20.75	0.69	0.27

## 496 C.2 Supplementary Results for Certified Classification

497 To further validate our certified classification results in Section 5 we empirically compare them  
 498 against sampled classifications. Specifically, we render an image at the center of each partition and  
 499 check whether it can be correctly classified. The result is shown in Table 6. The sampled correct  
 500 classification rate exceeds the verified rate from our method, suggesting potential over-approximation  
 501 in certification. We visualize this relationship in Figure 7 revealing that all verified partitions are  
 502 subsets of correctly classified sampled partitions. This demonstrates that our verification method,  
 503 while conservative, reliably under-approximates the set of robustly classifiable inputs, which ensures  
 504 soundness.

## 505 C.3 Supplementary Results for Certified Pose Estimation

506 To validate our certified pose estimation results (Section 5), we compare them against empirical  
 507 pose estimations. For each partition, we render an image at its center and evaluate whether the  
 508 pose estimation error falls below the predefined threshold (Table 7). While the sampled success  
 509 rate exceeds the certified rate, which suggests conservatism, the gap remains bounded: in the worst  
 510 case, the rate increases from 23.3% (certified) to 52.4% (sampled), while in the best case, it rises  
 511 only modestly from 70.0% to 75.1%. This indicates that our method avoids excessive conservatism  
 512 without compromising reliability. Figure 8 visualizes this relationship, confirming that all certified

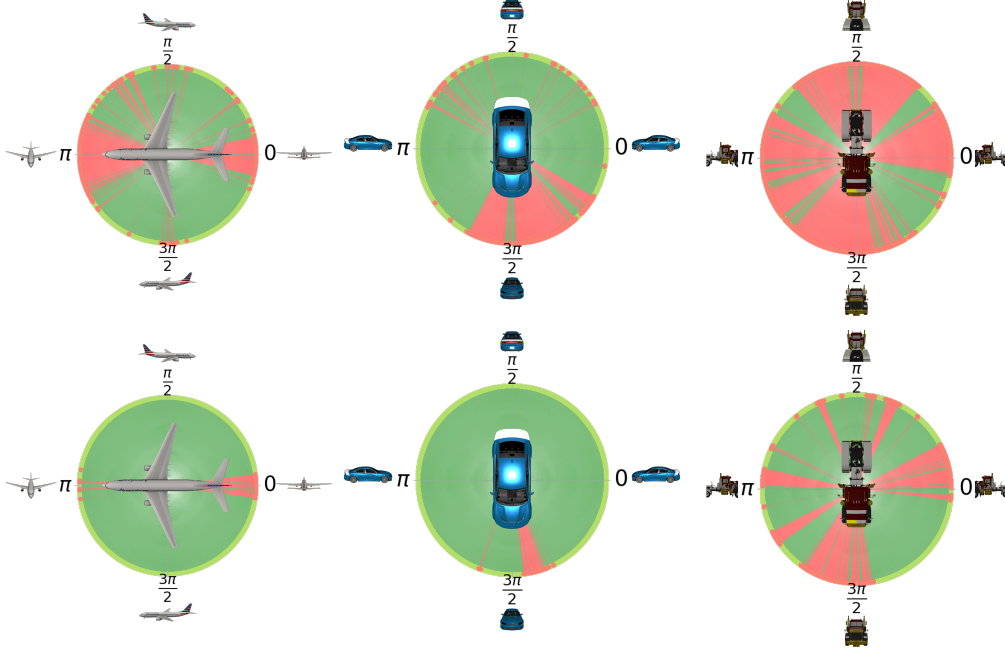


Figure 7: Classification result for PLANE, CAR, and TRUCK by GAUSSIANSPLAT across 0-360° camera rotation. Top row: Certified classification result. Bottom row: classification result obtained via sampling. **Green**: Provably robust classifications. **Red**: Classifications not provably robust to camera-angle changes.

Table 6: Certifiable Classification Results. The following factors are included in the evaluation: object class (obj); method used for scene representation and rendering (method, GS for GAUSSIANSPLAT); distance from camera poses to object centroid ( $r$ ); camera pose height ( $h$ ); number of partitions (#part); percentage of correct pose partitions by sampling (perS); percentage of verified pose partitions by our method (perC); and runtime by our method (Rt in minutes).

obj	method	$r(m)$	$h(m)$	#part	perS	perC	Rt
Plane	GS	40	10	62832	96.0%	74.5%	1140.5
Car	GS	8	2.5	62832	96.3%	76.7%	879.6
Truck	GS	4	1.2	62832	71.8%	31.9%	963.4
Plane	NeRF	40	36	3142	25.6%	22.8%	94.3
Car	NeRF	4	0.5	3142	75.9%	42.1%	118.6

pose ranges are subsets of empirically valid ranges, thereby further demonstrating the soundness of our verification approach.

#### C.4 Supplementary Results for Certified Object Detection

To validate our certified object detection results (Section 5), we empirically compare them against sampled detections. For each partition, we render an image at its center and assess whether at least one valid bounding box candidate exists. We compare the result with the certified object detection and the results are summarized in Table 8. The sampled rate exceeds the certified detection rate, indicating conservatism in our analysis. Further visualizations in Figure 9 confirm that all certified regions are subsets of empirically detected regions, demonstrating soundness.

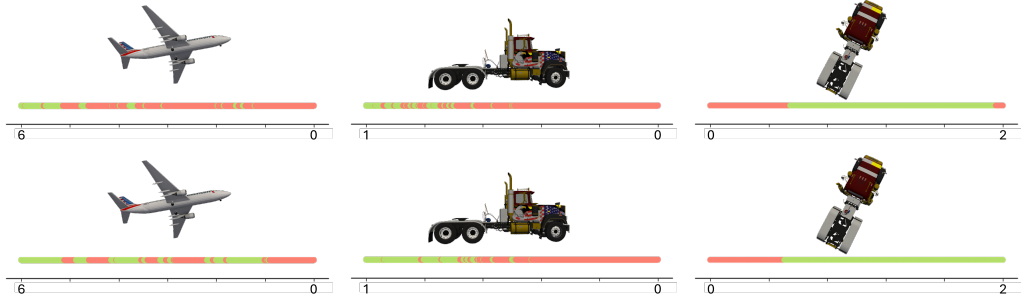


Figure 8: Certified pose estimation results for plane, truck by GAUSSIANSPAT and truck by NERF. Top row: Certified pose estimation result. Bottom row: Pose estimation result obtained via sampling. **Green**: Provably robust pose estimation. **Red**: Not provably robust.

Table 7: Certifiable pose estimation and object detection results. Object class: obj, Method use for scene representation and rendering: method, Distance from camera pose to object centroid  $r$ , Camera pose perturbation range:  $\epsilon$ , Percentage of verified pose partitions by sampling: perS, Percentage of verified pose partitions by our method: perC, and Runtime (min) by our method: Rt.

Setup		Certified Pose Estimation				
obj	method	$r$	$\epsilon$	perS	perC	Rt
Plane	GS	174.4	6 m	52.4%	23.3%	362.2
Truck	GS	7.8	1 m	37.6%	22.2%	311.3
Plane	NERF	40	20 m	46.7%	37.2%	82.4
Truck	NERF	8	2 m	75.1%	70.0%	53.4

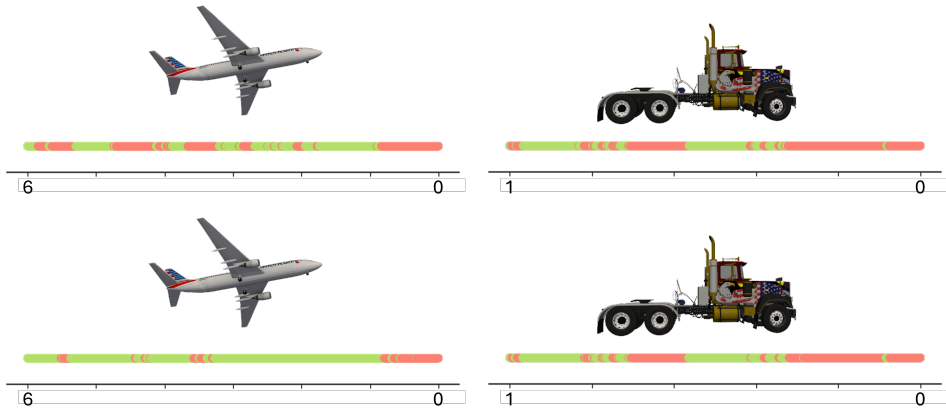


Figure 9: Certified object detection results for plane and truck by GAUSSIANSPAT. Top row: Certified object detection result. Bottom row: Object detection result obtained via sampling. **Green**: Provably robust pose estimation. **Red**: Not provably robust.



Table 8: Certifiable pose estimation and object detection results. Object class: obj, Method use for scene representation and rendering: method, Confidence threshold(applicable only to objection task): threshold, Camera pose perturbation range:  $\epsilon$ , Percentage of verified pose partitions by sampling: perS, Percentage of verified pose partitions by our method: perC, and Runtime (min) by our method: Rt.

Setup		Certified Object Detection				
obj	method	threshold	$\epsilon$	perS	perC	Rt
Plane	GS	0.45	6 m	81.5%	51.4%	422.2
Truck	GS	0.46	1 m	47.2%	42.4%	367.3
Plane	NERF	0.12	3.5 m	92.2%	80.4%	121.3
Truck	NERF	0.12	1.5 m	100.0%	71.0%	66.5

## 522 D Proofs

523 **Proposition 2.** *Any continuous function is linearly over-approximable.*

524 *Proof.* We present a simple proof for scalar  $f$  and this can be extended to higher dimensions. We fix  
525  $x_0 \in \mathbb{R}$  and a finite slope  $k \in \mathbb{R}$ . Since  $f(x) - k(x - x_0)$  is continuous, for any  $\varepsilon > 0$ , there must  
526 exist an open neighborhood  $U(x_0)$  around  $x_0$ , such that  $\forall x \in U(x_0)$ ,

$$|f(x) - f(x_0) - k(x - x_0)| < \frac{\varepsilon}{3}. \quad (1)$$

527 Equivalently,

$$f(x_0) + k(x - x_0) - \frac{\varepsilon}{3} < f(x) < f(x_0) + k(x - x_0) + \frac{\varepsilon}{3}. \quad (2)$$

528 We construct candidate linear upper and lower bounds over  $U$  as:

$$l_U(x) = k(x - x_0) + f(x_0) - \frac{\varepsilon}{3}, \quad u_U(x) = k(x - x_0) + f(x_0) + \frac{\varepsilon}{3}. \quad (3)$$

529 We can check that these linear functions  $l_U(x)$  and  $u_U(x)$  are valid lower and upper bounds for  $f(x)$   
530 over  $U$ , because  $\forall x \in U(x_0)$ :

$$f(x) - l_U(x) = f(x) - (k(x - x_0) + f(x_0) - \frac{\varepsilon}{3}) > -\frac{\varepsilon}{3} + \frac{\varepsilon}{3} = 0, \quad (4)$$

$$f(x) - u_U(x) = f(x) - (k(x - x_0) + f(x_0) + \frac{\varepsilon}{3}) < \frac{\varepsilon}{3} - \frac{\varepsilon}{3} = 0. \quad (5)$$

531 Additionally, difference between  $l_U(x)$  and  $u_U(x)$  is tightly bounded, as  $\forall x \in U(x_0)$ :

$$|u_U(x) - l_U(x)| = (k(x - x_0) + f(x_0) + \frac{\varepsilon}{3}) - (k(x - x_0) + f(x_0) - \frac{\varepsilon}{3}) = \frac{2\varepsilon}{3} < \varepsilon.$$

532 Now, for any compact set  $B$ , the collection  $\mathcal{U} = \{U(x) \mid x \in B\}$  forms an open cover of  $B$ . By  
533 the definition of compactness, there exists a finite subcover of  $B$ , denoted by  $\{U_i = U(x_i) \mid i =$   
534  $1, \dots, s\}$ . We can now define piecewise linear functions on  $B$  as follows:

$$\begin{aligned} u_B(x) &= \min_{j \in I} u_{U_j}(x), \\ l_B(x) &= \max_{j \in I} l_{U_j}(x), \quad \text{if } x \in \{U_j \mid j \in I\} \end{aligned} \quad (6)$$

535 where  $I$  is the index set of  $U_i$  that cover  $x$ . Consequently, we have

$$u_B(x) = \min_{j \in I} u_{U_j}(x) \geq f(x) \geq \max_{j \in I} l_{U_j}(x) = l_B(x), \quad (7)$$

536 and

$$|u_B(x) - l_B(x)| \leq u_{U_j}(x) - l_{U_j}(x) \leq \varepsilon, \quad \forall j \in I. \quad (8)$$

537 Thus,  $f$  is linearly over-approximable.  $\square$

538 **Lemma 4.** *Given any non-singular matrix  $X$ , the output of Algorithm MATRIXINV  $\langle lX_{\text{inv}}, uX_{\text{inv}} \rangle$*   
539 *satisfies that:*

$$lX_{\text{inv}} \leq \text{Inv}(X) \leq uX_{\text{inv}}$$

540 *Proof.* For any non-singular  $n \times n$  matrices  $X$  and  $X_{\text{ref}}$ , denote their difference as  $\Delta X = X - X_{\text{ref}}$ .  
 541 The  $k^{\text{th}}$  order Taylor Polynomial  $X_p$  and remainder  $R$  of matrix inverse of  $X$ , estimated at  $X_{\text{ref}}$ , can  
 542 be written as follows:

$$X_p = X_{\text{ref}}^{-1} \sum_{i=0}^k (\Delta X \cdot X_{\text{ref}}^{-1})^i \quad (9)$$

$$R = X_{\text{ref}}^{-1} \sum_{i=k+1}^{\infty} (\Delta X \cdot X_{\text{ref}}^{-1})^i \quad (10)$$

543 Assuming that  $\|\Delta X \cdot X_{\text{ref}}^{-1}\| < 1$ , the remainder  $R$  can be bounded by:

$$\text{Eps} = \|R\| \leq \|X_{\text{ref}}^{-1}\| \cdot \sum_{i=k+1}^{\infty} \|\Delta X \cdot X_{\text{ref}}^{-1}\|^i = \|X_{\text{ref}}^{-1}\| \cdot \frac{\|\Delta X \cdot X_{\text{ref}}^{-1}\|^{k+1}}{1 - \|\Delta X \cdot X_{\text{ref}}^{-1}\|} \quad (11)$$

544 Using the bound in inequality 11, we obtain bounds on  $X^{-1}$ :

$$X_p - \text{Eps} \leq X^{-1} \leq X_p + \text{Eps} \quad (12)$$

545 Thus, we establish lower and upper bound for  $X^{-1}$  as:

$$lX_{\text{inv}} = X_p - \text{Eps}, \quad uX_{\text{inv}} = X_p + \text{Eps}. \quad (13)$$

546 Finally, by replacing  $X_{\text{ref}}^{-1}$  with  $X_0$  in the expression for  $X_p$  (in Equation 9),  $\text{Eps}$  (in Equation 11),  
 547  $lX_{\text{inv}}$  and  $uX_{\text{inv}}$  (in Equation 13), we obtain the same expressions as those given in Algorithm  
 548 MATRIXINV.  $\square$   $\square$

549 **Lemma 5.** For any given inputs — effective opacities  $a$ , colors  $c$  and depth  $d$ , VR-IND outputs the  
 550 same value as computation from Line 12 to Line 16 in GAUSSIANSPAT.

551 *Proof.* We first derive the math expression of pixel color  $p_c$  from GAUSSIANSPAT. By denoting  
 552  $\text{arg sort}(d)$  as a mapping  $\sigma : l \rightarrow l$ , where  $l$  refers to the index set of Gaussians, as at Line 12,  $c_s$  at  
 553 Line 13 and  $o_c$  at Line 15 can be written as follows.

$$a_{s_i} = a_{\sigma(i)} \quad (14)$$

$$c_{s_i} = c_{\sigma(i)} \quad (15)$$

$$o_{c_i} = \prod_{j=1}^{i-1} (1 - a_{\sigma(j)}) \quad (16)$$

554 According to the definition of Ind operation, we have:

$$\text{Ind}(d_i - d_j) = \begin{cases} 1 & \text{if } d_i > d_j \\ 0 & \text{if } d_i \leq d_j. \end{cases} \quad (17)$$

555 Then by replacing  $i, j$  with  $\sigma(i), \sigma(j)$ , and multiplying  $a_{\sigma(i)}$ , (17) becomes:

$$a_{\sigma(i)} \cdot \text{Ind}(d_{\sigma(i)} - d_{\sigma(j)}) = \begin{cases} a_{\sigma(i)} & \text{if } d_{\sigma(i)} > d_{\sigma(j)} \\ 0 & \text{if } d_{\sigma(i)} \leq d_{\sigma(j)} \end{cases} \quad (18)$$

556 For any fixed index  $i$ , by multiplying all the case in the second branch of (18), it follows:

$$\prod_{j=i}^N (1 - a_{\sigma(i)} \cdot \text{Ind}(d_{\sigma(i)} - d_{\sigma(j)})) = 1. \quad (19)$$

557 By applying Equation 16, Equation 18 and Equation 19 into Line 15  $o_{c_i}$  can be written as follows:

$$\begin{aligned} o_{c_i} &= \prod_{j=1}^{i-1} (1 - a_{\sigma(i)}) \\ &= \prod_{j=1}^{i-1} (1 - a_{\sigma(i)} \cdot \text{Ind}(d_{\sigma(i)} - d_{\sigma(j)})) \\ &= \prod_{j=1}^N (1 - a_{\sigma(i)} \cdot \text{Ind}(d_{\sigma(i)} - d_{\sigma(j)})) \end{aligned} \quad (20)$$

558 Then by applying Equation 20 into Line 16, the pixel color  $pc$  computed via Algorithm BLENDSORT  
 559 can be expressed as:

$$pc = \sum_{i=1}^N \left( \prod_{j=1}^N (1 - a_{\sigma(i)} \cdot \text{Ind}(d_{\sigma(i)} - d_{\sigma(j)})) a_{\sigma(i)} c_{\sigma(i)} \right) \quad (21)$$

560 Since the summation and product operations are invariant to the reordering of input elements, and  
 561 both indices  $i$  and  $j$  iterate over the entire index set  $I$ , the reordering mapping  $\sigma$  in Equation 21 can be  
 562 eliminated, resulting in:

$$pc = \sum_{i=1}^N \left( \prod_{j=1}^N (1 - a_i \cdot \text{Ind}(d_i - d_j)) a_i c_i \right) \quad (22)$$

563 The last equation comes from 17

564 Next, we derive the math expression for  $pc$  in Algorithm VR-IND.  $oc_i$  at Line 2 can be written as:

$$oc_i = \prod_{j=1}^N (1 - a_i \cdot \text{Ind}(d_i - d_j)) \quad (23)$$

565 Then, by applying Equation 23 into Line 3 of Algorithm VR-IND, the expression for  $pc$  becomes:

$$pc = \sum_{i=1}^N \left( \prod_{j=1}^N (1 - a_i \cdot \text{Ind}(d_i - d_j)) a_i c_i \right) \quad (24)$$

566 Note that the expressions for  $pc$  are identical in both Equation 22 and Equation 24. This demonstrates  
 567 that Algorithms BLENDSORT and VR-IND yield the same input-output relationship while applying  
 568 different operations, thus completing this proof.

569 □