Hyperplane Culling for Stochastic Rasterization

J. Munkberg[1] and T. Akenine-Möller[1,2]

[1]Intel Corporation [2]Lund University

Abstract

We present two novel culling tests for rasterization of simultaneous depth of field and motion blur. These tests efficiently reduce the set of xyuvt samples that need to be coverage tested within a screen space tile. The first test finds linear bounds in ut- and vt-space using a separating line algorithm. We also derive a hyperplane in xyuvt-space for each triangle edge, and all samples outside of these planes are culled in our second test. Based on these tests, we present an efficient stochastic rasterizer, which has substantially higher sample test efficiency and lower arithmetic cost than previous tile-based stochastic rasterizers.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms

1. Introduction

Realistic camera effects, such as motion blur and depth of field, rendered with accurate visibility, may provide a major leap forward for image quality in real-time rendering. As a result, the research activity in the field of stochastic rasterization [CCC87, AMMH07] has increased over the last few years [FLB*09, MESL10, MCH*11, LAKL11, AMTMH12].

We note that the rasterizer in a contemporary graphics processor is heavily optimized and implemented in power-efficient, fixed-function hardware. In this paper, we focus only on the visibility problem of stochastic rasterization with the target being a new fixed-function unit. As a consequence, we do not implement our work on top of current graphics processors, e.g., in a compute shader. Also, we do not study shading here, since stochastic rasterization does not increase the shading rate substantially [CCC87, RKLC*11].

Below, we briefly summarize the most related work on *tile-based* stochastic rasterization techniques. From recent rasterization research [MCH*11, LAKL11, AMTMH12], it is clear that there are many benefits to using a hierarchical traversal with a test that culls samples on a per-tile basis. For motion blur rasterization, Laine et al. [LAKL11] developed a tight axis-aligned test in a dual space. Munkberg et al. [MCH*11] used a moving triangle versus tile frustum test, combined with a conservative edge test, with higher culling efficiency as a result. Efficient depth of field (DOF) culling tests have been developed using separating planes between the tile and the triangle [AMTMH12]. Laine et al. [LAKL11] used a similar test with a subset of the separating planes in their DOF algorithm.

Laine et al. [LAKL11] presented the first tile-based rasterization technique that handles *simultaneous* motion blur and depth of field. We consider this a major step forward. Culling is achieved by clipping down the extents of u, v, and t *individually*. Consider Figure 1, where a small out-of-focus triangle moves across the screen. With individual bounds for u and t, all (u,t) samples within the red box will be tested, while it is only the samples in the blue diagonal strip that can possibly hit the triangle. In a recent technical report, developed independently of our work, Laine & Karras [LK11] addressed this problem by extending the dual space motion blur test to a trilinear equation that takes correlations between u&t and v&t into account. They formulated a culling test where four bilinear patches are evaluated *for each sample*. While achieving very high culling rates, it is not clear if this test is beneficial from an arithmetic cost perspective, due to the significant per-sample cost.

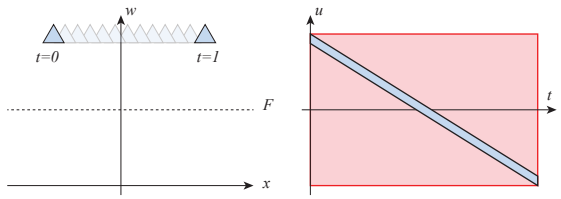


Figure 1: *A small out-of-focus triangle moves across the screen (left). The axis-aligned bounds (right) for this triangle in ut-space for a tile around $x = 0$, are very coarse, as only samples in the diagonal strip can hit the triangle.*

In this paper, we first extend previous work on depth of field rasterization by developing a novel test of a moving *and* defocused triangle against a tile. This tests exploits correlations in ut- and vt-space for tighter bounds and has a lower cost than Laine & Karras' trilinear test. For both mo-

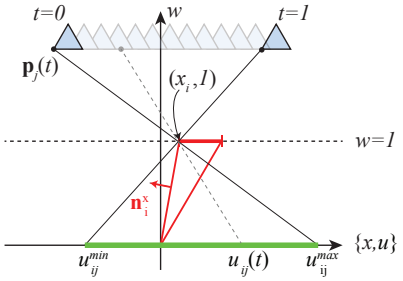


Figure 2: *A potential separating line, passing through a moving triangle vertex $\mathbf{p}_j(t)$ and a tile side (at $x = x_i$) in screen space, intersects the camera lens at $u_{ij}(t)$.*

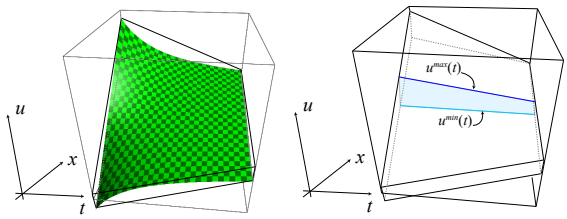


Figure 3: *Left: $u(x,t)$ is a rational function in t and linear in x. We derive a lower and upper bilinear bound for the surface. Right: given the bilinear bounds, we can, for a certain screen space position x, derive a lower ($u^{min}(t)$) and upper ($u^{max}(t)$) linear bound in ut-space.*

tion blur [MCH*11] and depth of field [AMTMH12], culling using planes aligned to triangle *edges* can increase the rasterization performance substantially in scenes with large or sliver triangles. Our second contribution is a triangle-edge culling test, which builds upon a conservative bounding of the five-dimensional edge equation. To our knowledge, there is no prior tile against triangle edge test that works for simultaneous motion and defocus blur. Our culling tests support arbitrary sampling patterns and provide trivial rejection tests of all samples within a screen space tile.

2. Tile-Based Culling

In this section, we introduce two culling tests for stochastic rasterization of *simultaneous* motion blur and depth of field. A clip-space vertex of a triangle is denoted $\mathbf{p} = (p_x, p_y, p_w)$. We assume linear vertex motion, $\mathbf{p}(t) = (1-t)\mathbf{q} + t\mathbf{r}$. The signed clip space circle of confusion radius of a moving vertex $\mathbf{p}_j(t)$ is a linear function $c_j(t) = (1-t)c_j^0 + t\, c_j^1$.

2.1. Culling with Linear Bounds in *ut* and *vt* Space

Previous work on depth of field rasterization [AMTMH12] derives separating plane tests between a screen space tile edge and a defocused triangle. Here, we generalize these tests to also take motion into account. We further use the result to derive linear bounds in ut- and vt-space.

In Figure 2, a potential separating line, passing through a moving triangle vertex $\mathbf{p}_j(t)$ and a tile corner (x_i, y_i), in screen space, intersects the camera lens at:

$$(u_{ij}(t)\ ,\ v_{ij}(t)\ ) \ = \ \left(\frac{\mathbf{n}_i^x \cdot \mathbf{p}_j(t)}{c_j(t)} \ , \ \frac{\mathbf{n}_i^y \cdot \mathbf{p}_j(t)}{k\, c_j(t)} \right). \qquad (1)$$

$\mathbf{n}_i^x = (-1, 0, x_i)$ and $\mathbf{n}_i^y = (0, -1, y_i)$ are normal vectors for the tile-frustum planes (planes containing the origin and a tile side) and $k > 0$ is a scalar that compensates for non-square aspect ratios. For a given screen space tile, the potential visible interval in the u-dimension denoted $\hat{u}(t)$, on the lens for a moving triangle is given by: $\hat{u}(t) = [\min_{i,j} u_{ij}(t), \max_{i,j} u_{ij}(t)]$. Note that by moving the denominator in Equation 1 to the left-hand side and collecting terms, this is the trilinear (dual space) equation [LK11].

To design an efficient culling test, we search for a conservative lower bound of all six (three vertices $\times$ two tile edges) $u_{ij}(t)$ functions. We represent this bound as a lower bilinear patch as shown in Figure 3. Appendix A includes the derivation of the patch. For each tile, we evaluate the bilinear patch, which has been padded with the tile extents, at the tile center and find a conservative bounding line in ut space. Individual uvt samples below this line are culled by a test on the form $u < at + b$. The upper linear bound is derived similarly and gives a second line above which we can cull samples. Applying the same technique to $v_{ij}(t)$ in Equation 1, we also derive two bounding lines in vt space. If the triangle intersects the focal plane, e.g., $c_j(t) = 0$, for some t, the culling test is disabled. In total, we cull samples against four bounding lines. The cost is eight linear interpolations per tile and four linear interpolations per sample. Additionally, by evaluating the four bounding lines at $t = 0$ and $t = 1$, we get bounds for the u and v ranges per tile.

2.2. Culling using Linearized Edge Equations

In this section, we conservatively bound the edge equation for a triangle with motion blur and depth of field using a hyperplane, and use this hyperplane to cull samples.

An edge equation for a triangle with linear vertex motion in clip space and depth of field is given by:

$$e(x,y,u,v,t) = (\mathbf{n}(t) + \mathbf{m}(t) \times (u, kv, 0)) \cdot (x, y, 1), \qquad (2)$$

where $\mathbf{n}(t)$ and $\mathbf{m}(t)$ are quadratic vectors in t. For each screen space tile R, this equation can be bounded by a hyperplane $\mathbb{P} : \mathbf{a} \cdot \mathbf{x} = 0$ in $xyuvt$ space, where $\mathbf{x} = (1, x, y, u, v, t)$, such that:

$$e(x,y,u,v,t) > \mathbf{a} \cdot \mathbf{x}, \ \ \forall (x,y) \in R. \qquad (3)$$

The full derivation is included in Appendix B.

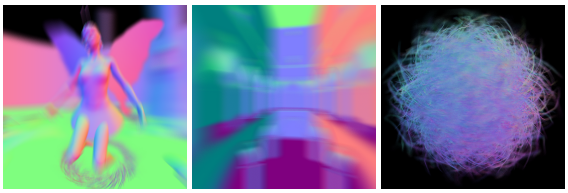
We recompute the hyperplane for each triangle edge and tile, and use the planes to cull samples within the valid $xyuvt$ domain. A tile can be trivially rejected if the cube of valid samples is entirely outside one hyperplane. This is done by picking the corner $\mathbf{c}$ of the hypercube farthest in the negative direction of the hyperplane normal. If $\mathbf{a} \cdot \mathbf{c} > 0$, the tile can be safely culled. Furthermore, the axis-aligned uvt bounds can be refined by the hyperplanes. In general: $\mathbf{a} \cdot \mathbf{x} > 0 \Leftrightarrow x_k > -\frac{1}{a_k} \sum_{i \neq k} a_i x_i$, which gives bounds for each coordinate

Scene	BBox	La	Lb	Lc	New
Fairy	28	28	28	28	67 (67)
blur	2.5	13	26	26	35 (44)
blur ×2	0.5	5.4	26	26	20 (23)
Sponza	7.4	7.4	7.4	7.4	78 (79)
blur	2.9	6.2	7.2	7.2	32 (50)
blur ×2	1.4	5.2	7.2	7.2	20 (30)
Hairball	6.7	6.7	6.7	6.7	20 (20)
blur	1.2	4.1	6.6	6.6	8.6 (14)
blur ×2	0.45	2.8	6.5	6.5	6.8 (8.8)

Table 1: *Sample test efficiency (higher is better) results using* 1×1 *pixel tiles. The first line for each scene shows the STE of a static rendering,* blur *has DOF and MB and* blur ×2 *has double shutter time and lens radius. The STE scores with hyperplane tests per sample are in the parenthesis.*

Scene	BBox	La	Lb	Lc	New
Fairy	1	1.3	1.4	1.6	0.8
blur	11	4.6	6.0	4.7	3.9
blur ×2	52	13	22	12	11
Sponza	1	1.2	1.4	1.5	0.2
blur	2.6	1.6	2.1	1.9	0.7
blur ×2	5.7	2.3	3.3	2.5	1.5
Hairball	1	1.5	1.7	1.8	1.0
blur	5.5	3.7	4.1	3.9	3.4
blur ×2	14	7.2	8.1	7.1	6.6

Table 2: *Execution times (lower is better) for a CPU implementation running on one core, with* 2×2 *pixel tiles and 16 samples per pixel, including triangle setup, rasterization, Gouraud shading, and framebuffer updates. All numbers are normalized against* BBox *on the static scene.*



Fairy (23) Sponza (92) Hairball (3)

Figure 4: *Test scenes (courtesy of Ingo Wald, Marko Dabrovic and Samuli Laine) with the* blur ×2 *setting. The average triangle area in pixels is indicated in parentheses.*

axis. For example, using the hypercube corner **c** from above, the temporal bounds can be refined by an inequality: $t > -(a_{\text{const}} + a_x c_x + a_y c_y + a_u c_u + a_v c_v)/a_t$, for each edge.

In our implementation, we pad the hyperplanes with half of a tile's screen space extents in the triangle setup, and evaluate the inner product $a_{\text{const}} + a_x c_x + a_y c_y$ once at the tile center. Inspired by previous work [AMTMH12], we selectively enable the hyperplane tests per triangle, by comparing the screen space AABB of the triangle at $(u,v,t) = (0,0,0)$ with the screen space AABB of the blurred triangle. If the former area is significantly smaller, the efficiency of the edges will be low, and the test is disabled.

3. Results

For our practical stochastic rasterizer (New), we use three culling tests, namely Laine et al.'s motion blur test in dual space [LAKL11], the bilinear test from Section 2.1, and the hyperplane test (Section 2.2). The triangle's screen-space AABB is computed using a previous technique [LK11]. We compare against Laine et al.'s algorithm [LAKL11] (La), Laine & Karras' algorithm [LK11] (Lb) and the (unpublished) combination of the two (Lc). For reference, we also include a brute-force rasterizer (BBox), which tests all samples within the triangle's bounding box. For a comparison against other stochastic rasterization algorithms (e.g. Interleave and Pixar [FLB*09]), we refer to the analysis by Laine et al. Our test scenes are presented in Figure 4. Sample test efficiency (STE) results, i.e., the fraction of the coverage tested samples that hits the triangle are summarized in Table 1. We analyze the general case of simultaneous motion blur and depth of field below. With *only* motion blur or DOF, New has STE on par with state-of-the-art algorithms [MCH*11, AMTMH12], yet at a higher cost.

The cost of the per-primitive setup, per-tile, and per-sample culling are summarized in Table 3, where we assume that tests on the form $u < u_{min}$ can be carried out by the hardware rasterizer at negligible cost. To give a rough estimate of the arithmetic cost, we report execution times of software implementations of all algorithms in Table 2. We use a tile size of 2×2 pixels in this measurement, as this was reported to give better performance than 1×1 pixel tiles in terms of arithmetic cost in Laine et al's paper [LAKL11]. Our algorithm scales favorably in all scenes. Although Laine & Karras' algorithm (Lb) has higher STE than La, the fine-grained culling operations per sample do not pay off in our software implementation. With modest or no blur, the overhead of the per-sample culling makes the algorithms Lb and Lc more expensive than the brute-force algorithm BBox. Note that a fixed-function hardware implementation may show different behavior. We leave that study for future work.

Discussion We do not cull against the hyperplanes per sample, which would add 9 fused multiply-add (FMA) operations to each sample test. In comparison, a single inside test costs about 25 FMA operations [LAKL11]. In our software implementation, these per-sample tests do not pay off. However, in a fixed-point hardware implementation it may be worthwhile. For reference, the STE scores with per-sample culling against the hyperplanes are included in Table 1 (numbers in parentheses). Our bilinear test is significantly less expensive than Laine & Karras' trilinear test and still allows for fine-grained culling in *ut*- and *vt*-space. A hard case for

	La	Lb	Lc	New
Per-triangle	500	400	700	900
Per-tile	40	16	56	50/130*
Per-sample	-	**12**	**12**	**4**

Table 3: *Cost estimates (arithmetic ops) of culling. In practice, the per-sample cost dominates. With selective enabling of the hyperplanes, the per-tile cost of* New *varies.*

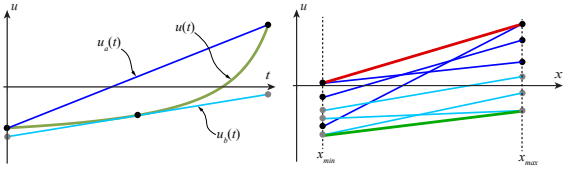


Figure 5: *Left: the green rational curve is bounded with an upper and a lower linear segment:* $u_a(t) = u(0) + t(u(1) - u(0))$ *and* $u_b(t) = u(\frac{1}{2}) - \frac{1}{2}\frac{\partial u}{\partial t}(\frac{1}{2}) + t\frac{1}{2}\frac{\partial u}{\partial t}(\frac{1}{2})$. *Right: a set of lines are bounded by a lower and upper linear approximation.*

our coarser approximation is triangles with large perspective motion. However, in most practical scenarios, the results indicate that our test is a better option in terms of total rasterization cost. The hyperplane test is expensive per tile, but efficient in scenes with modest blur, larger or sliver triangles.

Acknowledgements We thank the Advanced Rendering Technology team at Intel for valuable feedback, and particularly Aaron Lefohn and Charles Lingle for supporting this research.

References

[AMMH07] AKENINE-MÖLLER T., MUNKBERG J., HASSELGREN J.: Stochastic Rasterization using Time-Continuous Triangles. In *Graphics Hardware* (2007), pp. 7–16. 1, 4

[AMTMH12] AKENINE-MÖLLER T., TOTH R., MUNKBERG J., HASSELGREN J.: Efficient Depth of Field Rasterization using a Tile Test based on Half-Space Culling. *Computer Graphics Forum, 31*, 1 (2012), 3–18. 1, 2, 3

[CCC87] COOK R. L., CARPENTER L., CATMULL E.: The Reyes Image Rendering Architecture. In *Computer Graphics (Proceedings of SIGGRAPH 87)* (1987), vol. 21, pp. 95–102. 1

[FLB*09] FATAHALIAN K., LUONG E., BOULOS S., AKELEY K., MARK W. R., HANRAHAN P.: Data-Parallel Rasterization of Micropolygons with Defocus and Motion Blur. In *High-Performance Graphics* (2009), pp. 59–68. 1, 3

[LAKL11] LAINE S., AILA T., KARRAS T., LEHTINEN J.: Clipless Dual-Space Bounds for Faster Stochastic Rasterization. *ACM Transactions on Graphics, 30*, 4 (2011), 106:1–106:6. 1, 3

[LK11] LAINE S., KARRAS T.: *Improved Dual-Space Bounds for Simultaneous Motion and Defocus Blur*. Technical Report NVR-2011-004, NVIDIA, Nov. 2011. 1, 2, 3

[MCH*11] MUNKBERG J., CLARBERG P., HASSELGREN J., TOTH R., SUGIHARA M., AKENINE-MÖLLER T.: Hierarchical Stochastic Motion Blur Rasterization. In *High-Performance Graphics* (2011), pp. 107–118. 1, 2, 3, 4

[MESL10] MCGUIRE M., ENDERTON E., SHIRLEY P., LUEBKE D.: Real-Time Stochastic Rasterization on Conventional GPU Architectures. In *High Performance Graphics* (2010), pp. 173–182. 1

[RKLC*11] RAGAN-KELLEY J., LEHTINEN J., CHEN J., DOGGETT M., DURAND F.: Decoupled Sampling for Graphics Pipelines. *ACM Transactions on Graphics, 30*, 3 (2011), 17:1–17:17. 1

Appendix A: Bilinear Bounds in *xut* space

In Figure 3, we visualize one $u_{ij}(t)$ function for varying x, as a surface: $u_j(x,t) = \mathbf{n}^x \cdot \mathbf{p}_j(t)/c_j(t) = (x\, p_{j_w}(t) - p_{j_x}(t))/c_j(t)$. We want to bound this surface from below by a bilinear patch. We exploit that $u_j(x,t)$ is linear in x and for a given $x = x_0$, $u_j(x_0,t)$ is a rational function in t. If $c_j(t) \neq 0$, $\forall t \in [0,1]$, the rational function is monotone and can be bounded by two linear functions $u_a(t)$ and $u_b(t)$, as shown in Figure 5.

The bilinear patch must be conservative for all $t \in [t_0,t_1] \subseteq [0,1]$ and within the screen space horizontal extents $[x_0,x_1]$ of the moving and defocused triangle. To find a lower bilinear bound, we proceed as follows: first, fix $x = x_0$ and evaluate $\min(u_a(t),u_b(t))$, the minimum of the upper and lower linear bounds for the rational function $u_j(x_0,t)$, at $t = t_0$ and $t = t_1$. Repeat the procedure for $x = x_1$. This gives us four points (one for each (x_i,t_i) corner) that define a bilinear patch bounding $u^j(x,t)$ over $[x_0,x_1] \times [t_0,t_1]$. The procedure is repeated for all three $u_j(x,t)$ surfaces, and at each corner we keep the minimum value. The four points define a bilinear patch which is a lower bound of all three $u_j(x,t)$ functions. The points can be computed in the triangle setup, where we also pad the bounds with the half the tile size times the x-slope of the patch, so that we can conservatively evaluate the patch once at the tile-center.

Appendix B: Linearized Edge Equations

In this section, we show that an edge equation from a triangle with linear vertex motion in 2D homogeneous clip space and depth of field can be written as:

$$e(x,y,u,v,t) = \underbrace{\mathbf{n}(t)\cdot(x,y,1)}_{e_n} + \underbrace{(\mathbf{m}(t)\times(u,kv,0))\cdot(x,y,1)}_{e_m} \quad (4)$$

where $\mathbf{n}(t)$ and $\mathbf{m}(t)$ are quadratic vectors in t. Furthermore, we bound the equation by a hyperplane in *xyuvt* space.

The clip space vertex position is given by: $\mathbf{p}'(u,v,t) = \mathbf{p}(t) + c(t)(u,k\,v,0)$, where $c(t)$ is the signed clip space circle of confusion radius of vertex $\mathbf{p}$, and $k > 0$ is a scalar that compensates for non-square aspect ratios. The edge equation's normal is defined by:

$$\mathbf{p}'_0 \times \mathbf{p}'_1 = \mathbf{p}_0 \times \mathbf{p}_1 + (c_1\mathbf{p}_0 - c_0\mathbf{p}_1) \times (u,k\,v,0), \quad (5)$$

where the first cross product can be expressed as:

$$\mathbf{p}_0(t) \times \mathbf{p}_1(t) = (\mathbf{a}t^2 + \mathbf{b}t + \mathbf{c}) = \mathbf{n}(t). \quad (6)$$

This is the normal of the time-dependent edge equation [AMMH07]. To obtain Equation 4, we introduce $\mathbf{m}(t) = c_1(t)\mathbf{p}_0(t) - c_0(t)\mathbf{p}_1(t)$.

For efficient culling, we bound Equation 4 within a screen space region R by a hyperplane: $\mathbf{a}\cdot\mathbf{x} = 0$, where $\mathbf{x} = (1,x,y,u,v,t)$, such that $e(x,y,u,v,t) > \mathbf{a}\cdot\mathbf{x}, \quad \forall (x,y) \in R$.

We follow previous work on motion blur rasterization [MCH*11] to linearize the first term (e_n) of Equation 4.

$$e_n = \mathbf{n}(t)\cdot(x,y,1) > \mathbf{o}\cdot(x,y,1) + \gamma t, \quad \forall (x,y) \in R. \quad (7)$$

To bound $e_m = (\mathbf{m}(t)\times(u,kv,0))\cdot(x,y,1) = (u,kv,0)\cdot((x,y,1)\times\mathbf{m}(t))$, we first bound each component of $\mathbf{m}(t)$ and express $\hat{\mathbf{d}} = (\hat{x},\hat{y},1)\times\hat{\mathbf{m}}$ as a vector of intervals: $\hat{\mathbf{d}} = (\hat{d_x},\hat{d_y},\hat{d_z})$, where an interval $\hat{d} = [\underline{d},\overline{d}]$. A lower bound of e_m can then be written as:

$$e_m > \min(\hat{d_x}u + \hat{d_y}kv) > \alpha u + \beta v + \xi, \quad \forall (x,y) \in R. \quad (8)$$

The coefficients are given by $\alpha = \text{avg}(\hat{d_x})$, $\beta = k\,\text{avg}(\hat{d_y})$ and $\xi = -0.5(|\hat{d_x}| + k|\hat{d_y}|)$, where $\text{avg}(\hat{x}) = 0.5(\overline{x} + \underline{x})$ and $|\hat{x}| = \overline{x} - \underline{x}$.

To simplify the per-tile derivation of α,β and ξ, we express the average and width of $\hat{d_x}$ and $\hat{d_y}$ as linear functions of the tile center coordinates x_c and y_c respectively (in the triangle setup), padded to take the half-tile extents into account.

Combining Equation 7 and 8, we have shown that:

$$e(x,y,u,v,t) = e_n + e_m > \mathbf{a}\cdot\mathbf{x}, \quad (x,y) \in R, \quad (9)$$

where $\mathbf{x} = (1,x,y,u,v,t)$ and $\mathbf{a} = (\xi + o_z, o_x, o_y, \alpha, \beta, \gamma)$. □