Automotive Soiling Simulation Based On Massive Particle Tracing

Stefan Roettger [†], Martin Schulz [‡], Wolf Bartelheimer [*], Thomas Ertl [†]

[†] Visualization and Interactive Systems Group, IfI, University of Stuttgart
`http://wwwvis.informatik.uni-stuttgart.de`
[‡] Science+Computing GmbH, Tübingen
[*] BMW AG, München

Abstract. In the automotive industry Lattice-Boltzmann type flow solvers like PowerFlow from Exa Corporation are becoming increasingly important. In contrast to the traditional finite volume approach PowerFlow utilizes a hierachical cartesian grid for flow simulation. In this case study we show how to take advantage of these hierarchical grids in order to extend an existing Lattice-Boltzmann CFD environment with an automotive soiling simulation system. To achieve this, we chose to constantly generate a huge number of massive particles in user manipulable particle emitters. The process of tracing these particles step by step thus creates evolving particle streams, which can be displayed interactively by our visualization system. Each particle is created with stochastically varying diameter, specific mass and initial velocity, whereas already existing particles may decay because of aging, when leaving the simulation domain or when colliding with the vehicle's surface. On the one hand the display of these animated particles is a very natural and intuitive way to explore a CFD data set. On the other hand animated massive particles can be easily utilized for driving an automotive soiling simulation just by coloring the particles' hit points on the vehicle's surface.

1 Introduction and Motivation

In the flow visualization community many CFD simulation and visualization environments have been presented in the past [6, 5, 3, 1, 2, 10]. In order to solve the Navier-Stokes equations most of todays available simulation applications are based on the finite volume approach, which is well known and established in the automotive industry. More recent approaches such as PowerFlow from Exa Corporation [4] use a Lattice-Boltzmann simulation algorithm based on hierarchically refined cartesian grids (see Figure 1). With PowerFlow now being used as a standard flow solver in many development projects especially in automotive aerodynamics, there had been the demand for a visualization system that could take advantage of the special properties of the hierarchical grids. In cooperation with the BMW Group, Munich, such a visualization system was developed leading to a visualization application that allows interactive and intuitive exploration of a CFD data set [7]. A variety of well known flow visualization techniques like particle probes and cutting planes have been incorporated into the visualization system (see Figure 2), which can be configured also for immersive environments like the PowerWall or the CAVE.

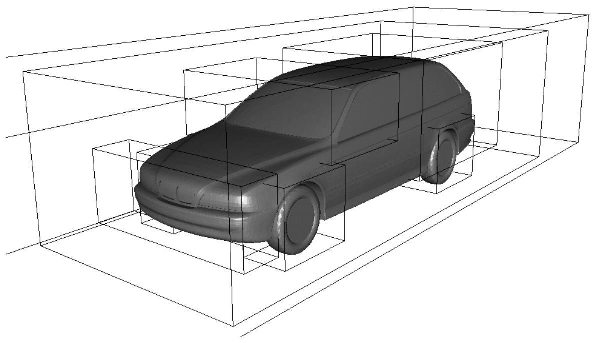


Fig. 1. The wireframes show the boundaries of a hierarchical cartesian grid, which is refined at the most interesting regions of a BMW 5 series for a Lattice-Boltzmann type CFD simulation.

Starting with this approach, the following observations lead to the development and inclusion of a soiling simulation module into the visualization system: In the traditional development process a vehicle cannot be checked for its soiling behaviour until an operable full-scale model is available. Since this is the case only at the end of the design process, our goal was to extend the existing Lattice-Boltzmann flow simulation and visualization environment in such a way that the flow engineer would be able to predict and understand the soiling behaviour at a much earlier design stage. Therefore, soiling problems could be detected before the expensive full-scale prototypes are being built. In order to achieve this we developed a particle animation system, which is able to interactively simulate the evolution of particle streams consisting of a large number of dust particles. The sustained creation and simulation of massive particles with stochastically varying diameter, initial velocity, initial position and specific mass produces a growing and evolving particle stream, which mimics the properties of dust. Each particle of the stream eventually decays either because its age is exceeding the allowed life time, or when a particle leaves the domain of the simulation, or because a collision with the vehicle's surface occured. Then the accumulation of dust on a car can be visualized simply by means of coloring each particle's hit point on the vehicle surface.

2 Physical Properties of Dust Particles

The first and most important step towards a realistic soiling simulation is to model the physical properties of dust correctly. We choose to restrict our model to dry driving conditions, because there are numerous less understood effects that influence soiling in a wet environment. Without great loss of accuracy dust particles can be idealized as spheres with a diameter D_p and a specific mass ρ_p. The drag coefficient c_{d_p}, which characterizes the forces induced on a dust particle by the fluid flow, basically depends on the particle's diameter and its velocity Δv_p relative to the flow. In general, the drag coefficient has to be measured experimentally, but for the low *Reynolds* numbers (with typical values of R_e up to 10) we en-

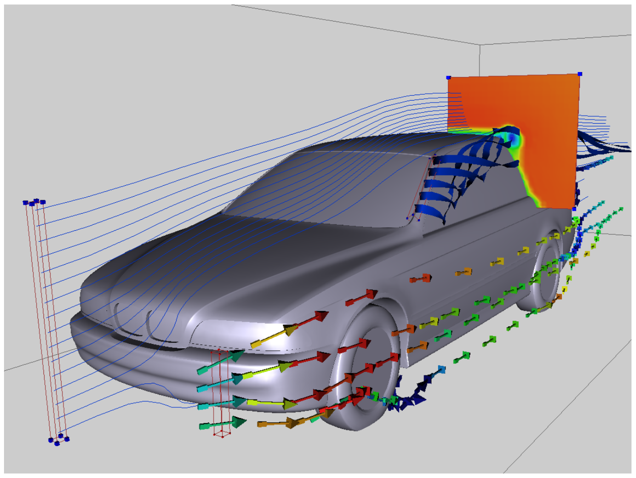


Fig. 2. An example of traditional flow visualization techniques: A cutting plane showing pressure values and several unanimated particles probes like stream lines, stream ribbons and glyphs.

counter in automotive CFD simulations the drag coefficient can be approximated with sufficient accuracy by the formula of *O'Seen*:

$$c_{d_p} = \frac{24}{R_e}\left(1 + \frac{3}{16}R_e\right), \quad R_e < 10 \quad (O'Seen)$$

Rewriting the formula of *O'Seen*, so that R_e is eliminated, we arrive at the following equation for the drag coefficient c_{d_p} in an air flow with density $\rho_{air} = 1.3\ \mathrm{kg/m}^3$ and viscosity $\nu_{air} = 10^{-5}\ \mathrm{m}^2/\mathrm{s}$:

$$R_e = \frac{D_p \cdot \Delta v_p}{\nu_{air}} \quad \rightarrow \quad c_{d_p} = \frac{24 \cdot \nu_{air}}{D_p \cdot \Delta v_p} + \frac{9}{2}$$

Next we substitute the drag coefficient c_{d_p} in the formula, which describes the forces induced on the particles by the fluid flow and we derive the following equation for a force F_p that drives a particle of diameter D_p, flow-effective area $A_p = \frac{1}{4}\pi {D_p}^2$, mass $m_p = \frac{1}{6}\rho_p \pi {D_p}^3$ and relative velocity Δv_p:

$$F_p = \frac{1}{2}\rho_{air} A_p c_{d_p} {\Delta v_p}^2 = \frac{1}{2}\rho_{air} A_p \Delta v_p \left(\frac{24 \cdot \nu_{air}}{D_p} + \frac{9}{2}\Delta v_p\right)$$

3 Massive Particle Tracing on Cartesian Grids

Now each dust particle that was emitted with stochastically varying diameter D_p, specific mass ρ_p, initial velocity v_{p_0} and initial position x_{p_0} is treated as a point mass. According to the following second order differential equation the integration of a particle's path is accomplished by an adaptive, embedded Runge-Kutta particle tracer, which integrates the acceleration $a_p(t)$ of a dust particle to compute the particle's position after a determined period of time [9].

$$x_p(t) = \int \left(\int a_p(t)dt + v_{p_0} \right) dt + x_{p_0}$$

Both the influence of the fluid flow and gravity must be accounted for, thus the total acceleration adds up to $a_p(t) = \frac{F_p(t)}{m_p} + g$ with $g = -9.81\ \mathrm{m/s}^2$. Adaptive embedded Runge-Kutta tracers have been evaluated in depth in numerical analysis. As the domain of the integration is basically a trilinearly interpolated and hierarchically refined cartesian grid it can be shown that an embedded Runge-Kutta integrator of order 4(3) is fully sufficient with respect to integration accuracy. Taking this into account we reimplemented a particle tracer as suggested in [8]. The integration step size, however, tends to be rather small for particle diameters well below $3\mu\mathrm{m}$. Fortunately, the average diameter of the relevant dust particles lies in the range of 5 to several $100\mu\mathrm{m}$.

4 Near-Surface Effects

For soiling simulations the correct near-surface behaviour of the simulated dust particles plays an important role with respect to simulation accuracy. The adhesion probability of a particle depends mainly on the speed by which it is approaching the surface. Faster particles are more likely to hit the surface than slower ones. Since flow velocity is zero on the vehicle's surface, the adhesion of each particle is driven by its momentum and by electrostatic effects (see Figure 3). At the time of writing the latter effect is not yet completely understood and will be subject to further research activities. For now we assume that the electrostatic forces are of inverse quadratic order ($F_e = c_e \cdot \frac{1}{r^2}$ with c_e being an electrostatic field coefficient and r being the Hausdorff distance to the vehicle's surface).

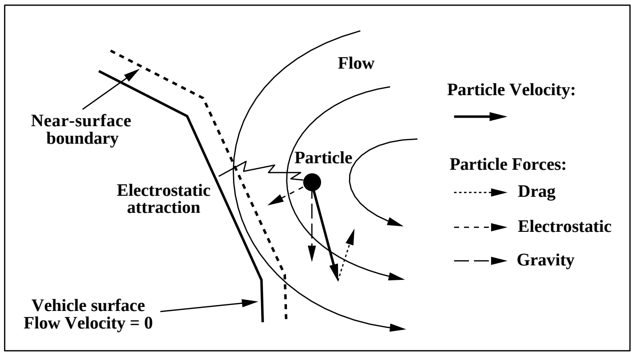


Fig. 3. A dust particle that is accelerated by its movement relative to the direction of the flow and by electrostatic and gravitational forces.

The cartesian representation of the flow field implies several restrictions. In principle, the vehicle geometry has to be stored and handled explicitly, since the simulation grid is derived by voxelization. For the same reason special care has to be taken when modeling any near-surface effect. In a correct physical setting the flow velocity, for example, is zero on the vehicle's surface as said before. This condition is not guaranteed when using a cartesian flow representation, because of the finite resolution of the grid. To compensate for that we define a so called near-surface zone. Its thickness equals the size of the smallest voxels in the grid (see Figure 4). Inside this near-surface zone the flow velocity is attenuated by the square root of the normalized Hausdorff distance to the vehicle's surface. These distances are calculated on-the-fly by utilizing an octree representation of the explicit set of surface triangles.

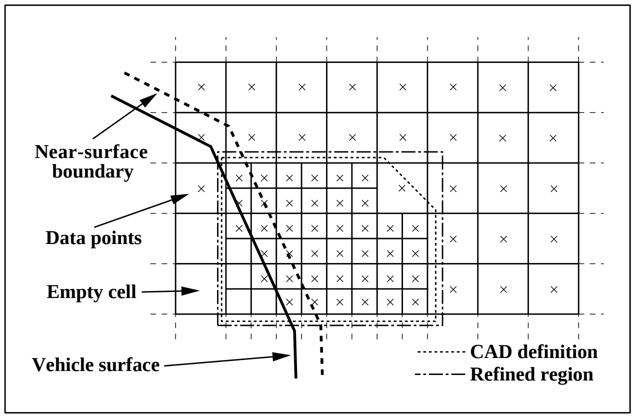


Fig. 4. Hierarchical grid representation with explicit vehicle surface and near-surface zone.

5 Massive Particle Visualization

With a particle tracer as described above a cloud of dust particles can be animated by continuously proceeding from keyframe k_n to keyframe k_{n+1}. During that task new particles are being created continuously in the extent of the particle emitters employing a stochastical process as mentioned before. Likewise, a fraction of the particles is decaying constantly because of their age, when colliding with the surface of the vehicle, or when leaving the domain of the simulation. The emerging stream of dust particles is visualized by a virtual camera with constant exposure time, such that particles with higher velocity result in longer traces on each image of the animation than slower ones (see Figure 5). This provides the viewer with an improved physical and three-dimensional insight into the flow properties, since the velocity of each particle is implicitly visible. By utilizing SGI's *Cosmo3D/OpenGL-Optimizer* the particle animation is embedded into an immersive environment, which provides the flow engineer with the ability to navigate and manipulate the particle emitters in an intuitive way. This procedure is very similar to the handling of smoke probes in a real-world wind tunnel, hence the acceptance of such a visualization technique is high.

One the one hand we mentioned that we wanted to trace the particles at real-time, but for an approaching flow with a typical velocity of $u_\infty = 20\ \mathrm{m/s}$ a particle is passing the vehicle in circa 0.3 seconds on the average, which is far too fast for a human viewer to catch any details. The solution is to interactively trace the particle cloud in slow motion. On the other hand a particle is very likely to leave the simulation domain before it has reached a particle age of typically 2 to 4 seconds, which mainly depends on the mass of the particles. Under normal conditions the total number of traced particles then stabilizes on a level that is determined by the frequency of particles being created and the average time the particles spend in the domain of the simulation. In our case, however, we have found it necessary to set the maximum life time of a particle to about 5 seconds to reliably prevent the rare case of particles rotating endlessly due to simulation or integration errors. Otherwise a constantly increasing number of simulated particles would slow down the simulation.

The continuous step by step integration of the entire particle stream can be parallelized efficiently. Therefore, the simulation can be sped up dramatically on symmetrical multi-processing systems. Nevertheless, the tracing of massive animated particles at high framerates requires an enormous amount of flow field evaluations and interpolations per second. By taking advantage of the hierarchical cartesian grids, cell localization and trilinear interpolation can be performed very efficiently, so that a high number of animated particles can be displayed simultaneously.

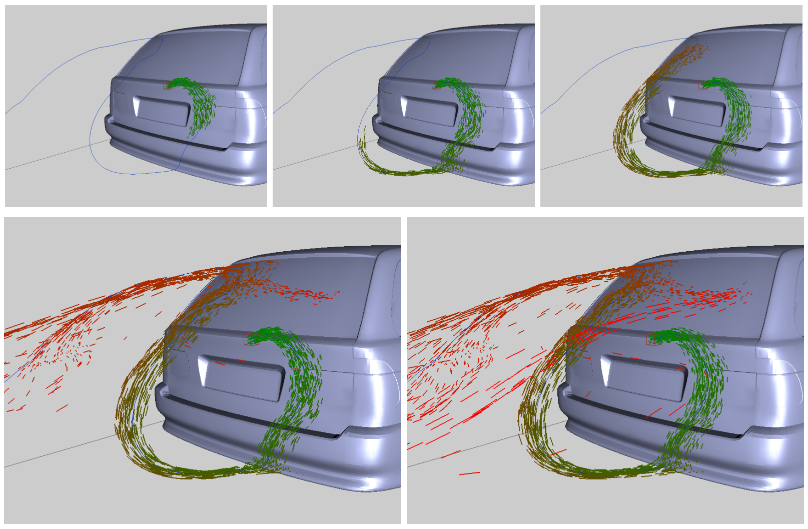


Fig. 5. Steps 1 to 5 of an interactive particle animation. Dust particles are emitted from a small user definable wireframe box behind the car. The emerging massive particle stream is first moving down and then up the back window. Eventually, it is diverging at the top of the car due to shearing forces.

Now the accumulation of dirt on a vehicle's surface can be visualized easily by coloring the hit points on the surface of the car. In our case the nearest vertices on the car surface are searched. The color of each of the vertices is determined by colorcoding the number of encountered hits. For that purpose hardware-accelerated 1D-textures are employed, which also allow smooth, Gouraud-shaded transitions between the vertices. In order to accelerate the detection of the hit points we are taking advantage of an octree representation of the explicit car surface, which reduces the total number of visited triangles from 70.000 of the original CAD model to an average of 8 to 15 triangles per octree search. By storing a single bit per voxel, which denotes the presence of geometry, we can speed up collision detection even further.

6 Results and Conclusion

A series of screenshots from a soiling simulation employing the techniques mentioned above is shown in Figure 6. The total run time was approximately 24 hours of continuous particle animation with screenshots taken after 15 minutes, 45 minutes, 2 hours, 12 hours and finally 24 hours. During the simulation run our particle animation system was simultaneously tracing an average number of about 930 dust particles on an SGI Octane MXI. With its dual 250 MHz MIPS R10K processors the refresh rate of the multi-threaded animation was 7 Hz, which corresponds to a speedup of about 85% in comparison to a single processor system. Under normal dry driving conditions the probability for a single dust particle in the air to hit the car is fairly low. When using time-averaged flow fields, which smooth out the transient flow components, the probability by which a particle hits the surface is reduced even further. Therefore the main problem of a soiling simulation as discussed above is the huge number of emitted particles that must be traced in order to achieve a sufficiently smooth dust distribution on the vehicle's surface. To improve this we are planning to use time-dependent flow fields in the future. Besides this, a variety of less understood physical effects and environmental influences makes it very difficult to compare the result of a soiling simulation to the real-world evaluations conducted by BMW. Nevertheless, we have been able to verify that our physical model of the dust behaviour is sufficiently exact. In order to improve our knowledge about the involved electrostatic effects near the vehicle's surface further research activities are needed. In the mean time our efforts will serve as a basis for further convergence towards the complex real-world soiling situation and as an accompanying visualization tool for the flow engineers.

7 Acknowledgments

This work has been financed by FORTWIHR, the Bavarian Consortium for High Performance Scientific Computing.

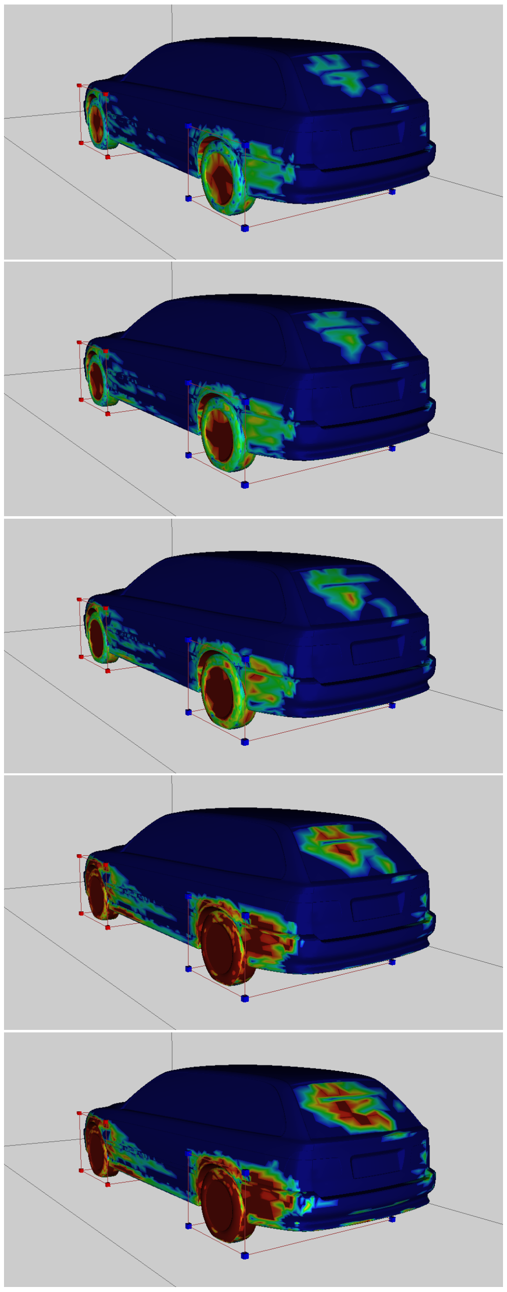
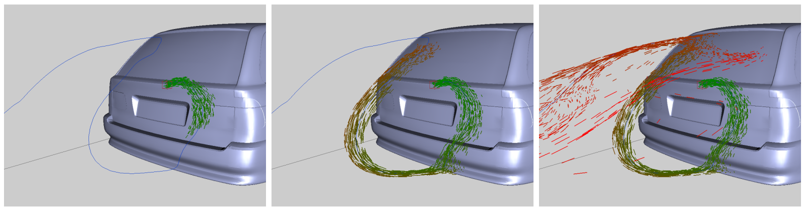


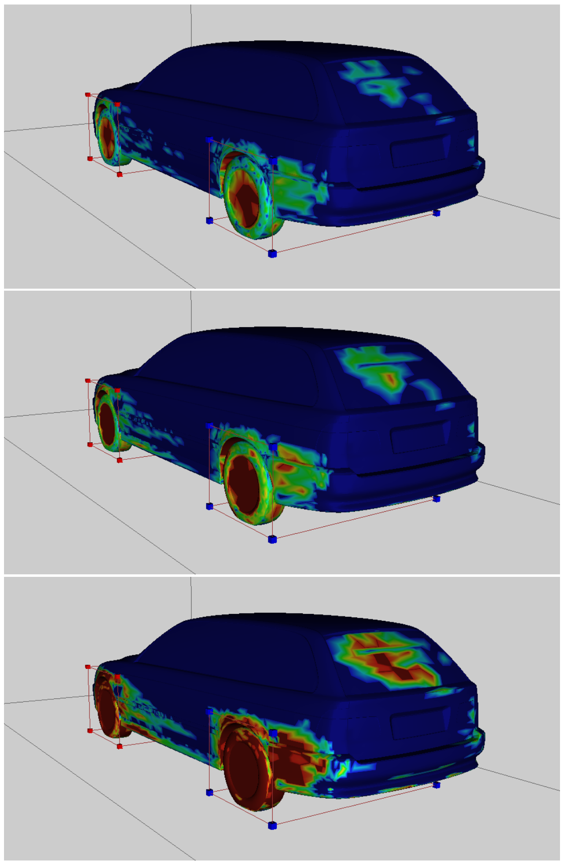
Fig. 6. Steps 1 to 5 of a soiling simulation running for approximately 24 hours on a dual processor SGI Octane MXI with 2x250MHz MIPS R10K. The colorcoding shows the hit rate of particles colliding with the car body. Red hot spots indicate areas with a high degree of soiling.

References

1. S. Bryson and C. Levit. The Virtual Windtunnel. In *IEEE Computer Graphics and Applications, 12(4):25-34*, 1992.
2. Steve Bryson. Approaches to the Successful Design and Implementation of VR Applications. In *Proc. SIGGRAPH'94*, 1994.
3. Steve Bryson and Steven Feiner. Virtual Environments in Scientific Visualization. In *Virtual Reality for Visualization, Course Notes of Tutorial 5 at Visualization 95*, 1995.
4. Exa Corporation. *http://www.exa.com*.
5. D. A. Lane. Scientific Visualization of Large-Scale Unsteady Fluid Flows. In G. Nielson, H. Hagen, and H. Mueller, editors, *Scientific Visualization*, pages 125–145. IEEE Computer Society, 1997.
6. F. Post and T. van Walsum. Fluid Flow Visualization. In H. Hagen, H. Mueller, and G. Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer Berlin, 1997.
7. M. Schulz, F. Reck, W. Bartelheimer, and Th. Ertl. Interactive Visualization of Fluid Dynamics Simulations in Locally Refined Cartesian Grids. In *Proc. Visualization '99*. IEEE, 1999.
8. C. Teitzel, R. Grosso, and T. Ertl. Efficient and Reliable Integration Methods for Particle Tracing in Unsteady Flows on Discrete Meshes. In W. Lefer and M. Grave, editors, *Visualization in Scientific Computing '97*, pages 31–41, Wien, 1997. Springer.
9. S.K. Ueng, C. Sikorski, and K.L. Ma. Efficient Construction of Streamlines, Streamribbons and Streamtubes on Unstructured Grids. *IEEE Transactions on Visualization and Graphics*, 2(2):100–109, June 1996.
10. S. P. Uselton. exVis: Developing A Wind Tunnel Data Visualization Tool. In R. Yagel and H. Hagen, editors, *Proc. Visualization '97*. IEEE, 1997.



Steps 1 to 3 of an interactive particle animation. Dust particles are emitted from a small user definable wireframe box behind the car. The emerging massive particle stream is first moving down and then up the back window. Eventually, it is diverging at the top of the car due to shearing forces.



Steps 1 to 3 of a soiling simulation running for approximately 24 hours on a dual processor SGI Octane MXI with 2x250MHz MIPS R10K. The colorcoding shows the hit rate of particles colliding with the car body. Red hot spots indicate areas with a high degree of soiling.